Implementation of a Reckoner Facility on the Lincoln Laboratory IBM 360/67

PETER B. HILL

AUERBACH Corporation, Philadelphia, Pennsylvania

ARTHUR N. STOWE

Lincoln Laboratory,[1] *Lexington, Massachusetts*

I. Introduction

Another paper in this volume [1] has presented the concept of coherent programming as it is embodied in the Lincoln Reckoner [2]. The term "coherent" implies a set of programs which may be independently developed, perhaps written in different languages, but which nevertheless operate on each other's results and call each other. This concept is being used to build a Reckoner facility on the IBM System 360 Model 67 at Lincoln Laboratory.

Coherent programs will be supported by a base of services which we have called the Mediator. The Mediator, although outside the set of coherent programs, provides the coherent programs with services, including storing and retrieving files, and running other coherent programs.

The primary mechanism for locating data files and programs is the stratified directory. A general rule of the Mediator is that all files and programs are located by name, using Mediator data services. Under this procedure, file addresses are passed to the coherent programs. Once located by the Mediator data services, each file is operated upon directly by a coherent program.

The status of the coherent programs which have been interrupted, or have suspended themselves when calling on other coherent programs, is maintained by the Mediator in a kind of push-down stack called the Return List.

[1] Operated with the support of the U.S. Air Force.

Both core and peripheral storage are managed by a set of services which, like the directory and return list services, are provided by the Mediator to fulfil one of the requirements of coherent programs, namely, that they and their results do not get in each other's way.

II. Stratified Directory

A major functional requirement of the Mediator is that the user be able to employ any set of coherent programs to solve his problem, regardless of the languages required for its execution. For example, all programs designed to operate on English text should be available to a user who is working on text.

The problem, of course, is that each language has its own syntax and naming rules which are likely to be in at least partial conflict with those of another language. It is the Mediator's role to provide the means by which these conflicts can be resolved. One of the mechanisms used is the stratified directory shown in Fig. 1.

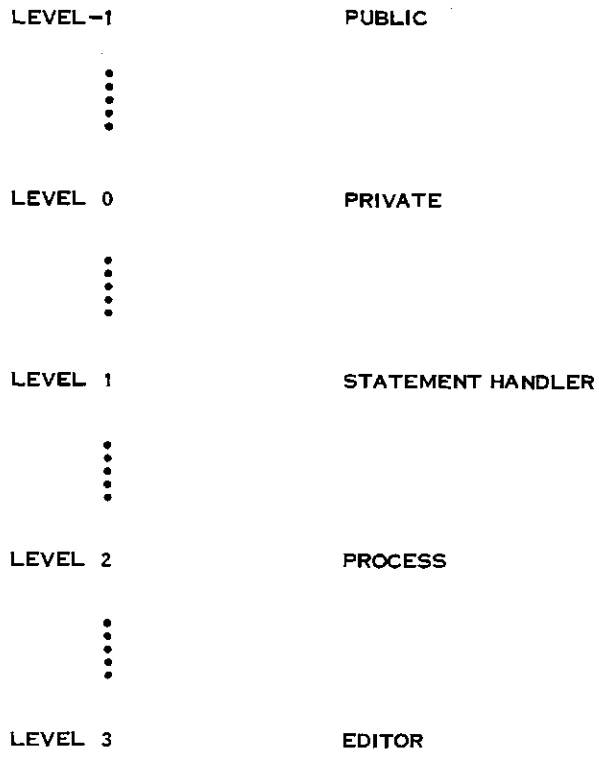

FIG. 1. The stratified directory.

The public directory, Level − 1, contains the names of public coherent programs, i.e., Reckoner routines, System Editor, etc., which are available to every user. Level 0 contains entries for files which have been accumulated by a user in past console sessions. This level is created when a user is first introduced to the system.

Level 1, normally containing the statement handler, is created each time the user initiates a console session. At the end of the session, this level is merged into the cumulative private level.

The next two levels would be constructed if a user called upon a process which, in turn, called on an Editor service. The levels are established automatically, without the user being aware of it. Whenever a routine which crosses a language barrier is initiated, a new level is created; as each barrier is recrossed upon a return, the just completed level is merged back into its predecessor.

The significance of the stratified directory is that each level becomes a *linguistic context* in which the meanings of names may be different from the meanings they have in other levels, preceding or following.

III. Temporary Names and Parameters

Two other mechanisms for manipulating data within levels and across levels involve the use of temporary names, and the use of parameters (see Fig. 2). This figure shows a Reckoner process for computing the hypotenuse C, given the two inputs A and B. The line labeled Parameter contains the internal names by which the variables are known. The next line contains the names of ephemeral entities which will disappear when the process is completed. All of this is quite familiar to any programmer.

The primary difference is the manner in which parameters and temporaries are handled by the Mediator. Both are handled by name, not by address, as they are in ALGOL or FORTRAN. Another difference is that in the management of both, the Mediator must deal with the problems of crossing from one linguistic context to another.

IV. Implementation

At present, the implementation of the Reckoner and Mediator is proceeding in two stages: the first stage will be completed in January 1968, and the second stage will be completed in the Spring of 1968.

```
HYPOTENUSE

● PARAMETER  A B C

● TEMPORARY AA BB CC

● MUL A A AA

● MUL B B BB

● ADD AA BB CC

● SQRT CC C

● FINIS
```

FIG. 2. Computing HYPOTENUSE.

The first phase items which comprise the basic Mediator are shown in Fig. 3. All essential services will be included; some less vital services will be postponed to the second phase.

The set of coherent programs comprises the bulk of the first phase. Most of the Reckoner computational services will be written in FORTRAN IV. These services include basic scalar and array arithmetic, matrix algebra and routines to construct arrays from scalars, vectors, and smaller arrays. A set of error routines will also be built to print out computational error messages such as dividing by zero, or logical errors such as requesting a file that does not exist. Also tentatively planned for the first phase is a stripped down Reckoner process builder and runner. These facilities will allow the user to write named processes which can be stored away for later operation. The Hypotenuse shown in Fig. 2, is an example of such a process.

The second phase items, shown in Fig. 4, will be expanded to include a broader range of services. The user will have more options that will allow him to field errors automatically. The Reckoner process builder and runner will be augmented to provide complete services.

In addition, we plan to write a set of programs which will allow the user to express his computations and logical operations in an algebraic language.

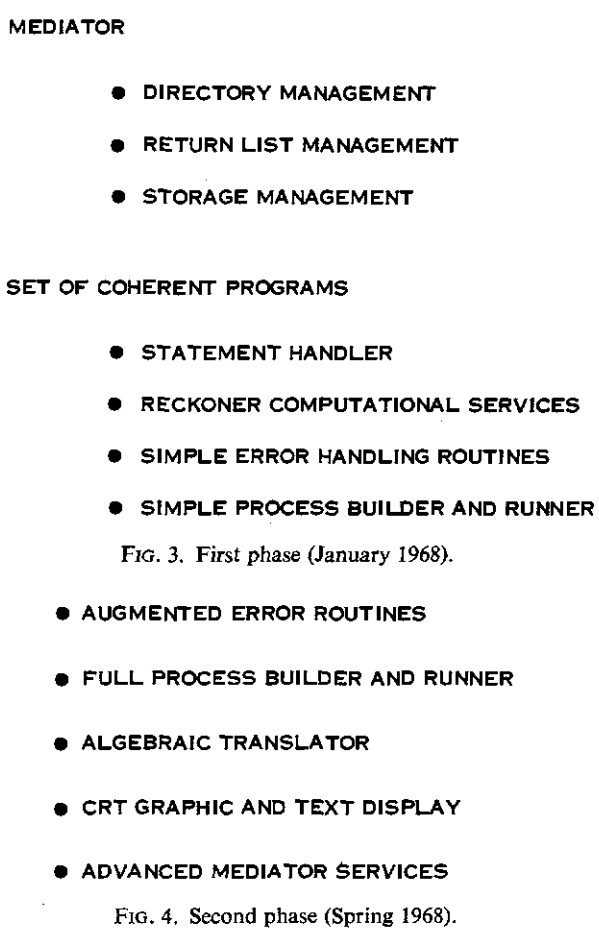

FIG. 3. First phase (January 1968).

FIG. 4. Second phase (Spring 1968).

A cathode ray tube facility will be added to allow the user to display his data in either graphical form or as tabulated data.

The completion of this second phase of the Mediator system will give the user a highly versatile on-line computational system and, at the same time, will not require that he be a professional programmer.

REFERENCES

1. WIESEN, R. A., YNTEMA, D. B., FORGIE, J. W., and STOWE, A. N., Coherent Programming in the Lincoln Reckoner, this volume.
2. STOWE, A. N., WIESEN, R. A., YNTEMA, D. B., and FORGIE, J. W., The Lincoln Reckoner: An Operation-Oriented, On-Line Facility with Distributed Control, *Proc. Fall Joint Comput. Conf.* **29**, 433 (1966).