CONF-710302--10

IDAHO NUCLEAR CODE AUTOMATION:
A Standardized and Modularized Code Structure

by

C. W. Solbrig
L. J. Ybarrondo
and
R. J. Wagner

Nuclear Safety Development Branch
Idaho Nuclear Corporation

March 1971

ABSTRACT

This paper presents the Idaho Nuclear Code Automation (INCA) system to be implemented during the course of the Loss-of-Coolant Accident Analysis Program being conducted for the Atomic Energy Commission as part of the Loss-of-Fluid Test (LOFT) program.

Most of the knowledge, both analytical and experimental, concerning nuclear safety analyses will ultimately reside in computer codes. The value and reliability of this knowledge depends upon the accessibility and ease of use of these codes. Some of the problems encountered in the present generation of nuclear safety codes are (1) difficulty in interfacing codes, (2) inconsistency between codes, (3) long running time to ensure stability, and (4) overemphasis on empirical correlations. This document describes a comprehensive plan for developing the third generation of nuclear safety codes with an integrated code automation system that will alleviate these difficulties.

IDAHO NUCLEAR CODE AUTOMATION (INCA) SYSTEM

I. INTRODUCTION

The purpose of this paper is to present the Idaho Nuclear Code Automation (INCA) System which is to be incorporated into the Loss-of-Coolant Accident Analysis Program[1] being conducted for the Atomic Energy Commission as part of the Loss-of-Fluid Test (LOFT). Most of the knowledge, both analytical and experimental, of nuclear safety analysis will ultimately reside in computer codes. The value of this knowledge for nuclear safety analyses depends upon the accessibility and ease of use of these codes. This vital knowledge represents a considerable investment. In order to maximize the return on this investment, nuclear safety codes must be developed within an overall code structure.

The present nuclear safety computer codes developed by Idaho Nuclear Corporation and other organizations have been developed on the basis that the entire nuclear plant can be divided into subsystems and that each subsystem can be analyzed independently of the others. For example, the thermal and hydraulic analysis of the primary loop of a pressurized water reactor (PWR) reactor is separated in two parts: (1) the flow loop with a very simple representation of the core and (2) the reactor core. The heat transfer in the core has been found to influence strongly the flow in the loop during accident conditions so that analytical decoupling is not justified. Since, in general, these codes were not designed to be run together, considerable effort is required to interface them. In fact, because they were not designed to be run together, difficulties were experienced in determining whether significant coupling occurs between the core and the loop. The investigation of the coupling was completed by upgrading the model of the reactor core which was used in the loop code and noting that significantly different results were obtained between the modified and unmodified codes for the same physical problem. An overall code structure would allow the interaction between components to be investigated easily yet each of the components could be run separately if the interaction was found negligible.

Some of the general problems which exist in the present manner of conducting a nuclear safety analysis are:

(1) Many codes which presently exist represent duplication of effort and yet contain contradictory models.

(2) Codes which do represent duplication of effort and should produce the same results are difficult to compare because of different models as well as input and output differences.

(3) Some codes produce input for other codes or require results from the running of other codes either sequentially or iteratively. These codes do not interface easily and must either be interfaced manually or considerable effort must be expended in combining the codes. For example, a thermal and hydraulic loop code and thermal and hydraulic core code should have compatible friction factor models.

(4) A slight change in the physical system such as a change in the cooling fluid quite often requires a code to be completely rewritten.

(5) Certain computations which should be the same in different codes are not. For example, the computation of the friction factor is required by most thermal and hydraulic codes but is often different in each one.

(6) Certain computations which were originally the same and should have remained the same in different codes have not. If a particular computation is upgraded in one code, it will probably not be ugraded in the second code.

All of these difficulties can be circumvented by use of an overall code structure with properly constructed codes and subcodes. The development of modular structures similar to that presented in this paper are described in the literature[2,3,4,5,6,7]. These modular code systems have been designed primarily for nuclear calculations under conditions that do not apply to a loss-of-coolant accident; however, the general schemes referenced apply to a more comprehensive system.

The referenced systems have been more concerned with linking existing codes that may or may not be consistent. The description presented here also utilizes this concept but goes one step further to present a method of code development that will provide consistency between codes and a simple method to keep all codes up-to-date and coincident with the "state of the art" and the latest experimental results. This paper, then describes how the loss-of-coolant accident will be separated into component parts and analyzed in terms of a unified code structure replete with modular codes and subcodes.

II. OBJECTIVES

The specific objectives of developing codes in this structure are to:

(1) Provide a uniform structure in which codes can be developed and tested.

(2) Eliminate duplication of effort, where possible.

(3) Provide a method of easily comparing different versions of codes where duplication of effort is required.

(4) Provide a method of easily interfacing codes either sequentially or iteractively so that strong coupling effects can be investigated.

(5) Provide a structure that will allow the physical system to be changed easily.

(6) Provide consistent calculations in different codes.

(7) Provide simultaneous updating of similar calculations in different codes.

(8) Provide a structure which will allow portions of the system which were not previously analyzed or cannot be analyzed at this time to be included in the analysis.

(9) Provide a uniform and sound basis for comparing the results of analytical models with those of scoping experiments.

III. GENERAL DESCRIPTION OF THE INCA STRUCTURE

A code system which includes both characteristics of standardization and modularization is an automated code system. The two characteristics are illustrated in Figure 1. The overlapping circles represent computer codes which have common computational procedures. Standardization requires that the overlapping areas be exactly the same in each code. Modularization requires that each of these overlapping areas be modules. Modules are defined as codes or subcodes designed to be completely interchangeable. Modules serve the same purpose and interface exactly the same in the systems that use them.

For example, Circle A could represent the computer code CONTEMPT-PS[8], Circle B the computer code RELAP3[9], and the common area "x" a friction factor subcode common to both CONTEMPT-PS and RELAP3. Standardization requires that the friction factor calculation be the same in both codes. Modularization requires that a different version of the friction factor calculation be interchangeable.

The preceding discussion illustrates the basic concepts of an automated code system. The remainder of this section describes the code automation system that will be implemented by Idaho Nuclear Corporation. The three-level INCA system will be described first and the characteristics of the modules included in each level will be discussed later.

1. THE THREE-LEVEL INCA SYSTEM

Figure 2 illustrates the three-level modular structure of the INCA system. The three levels include (1) the executive level, (2) the modular code level, and (3) the modular subcode level. Figure 2 illustrates how some of the codes that have been developed or used for analysis by Idaho Nuclear Corporation would function in the overall code structure. For example, the analysis of a containment vessel requires knowledge of the flow rate and energy out of the broken section of the primary coolant reactor loop of a large water reactor because this is the input to the containment analysis. The flow rate from the primary loop in turn is dependent upon the heat generation rate within the reactor core. Figure 2 shows that the executive level code sequence would control the selection of the codes required to complete this computation. This code sequence would select the desired nuclear kinetics code to supply the heat generation in the core. This result would then be used as an input

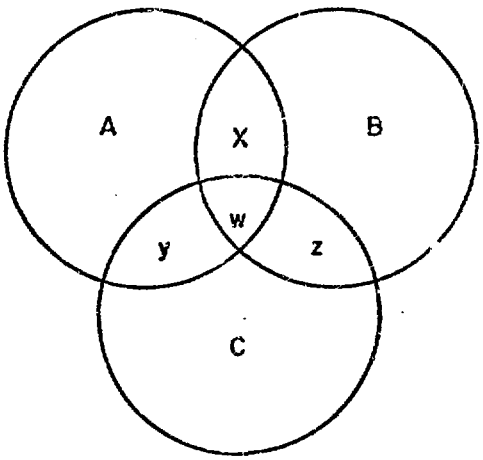


FIGURE 1 Two-Level Modular System

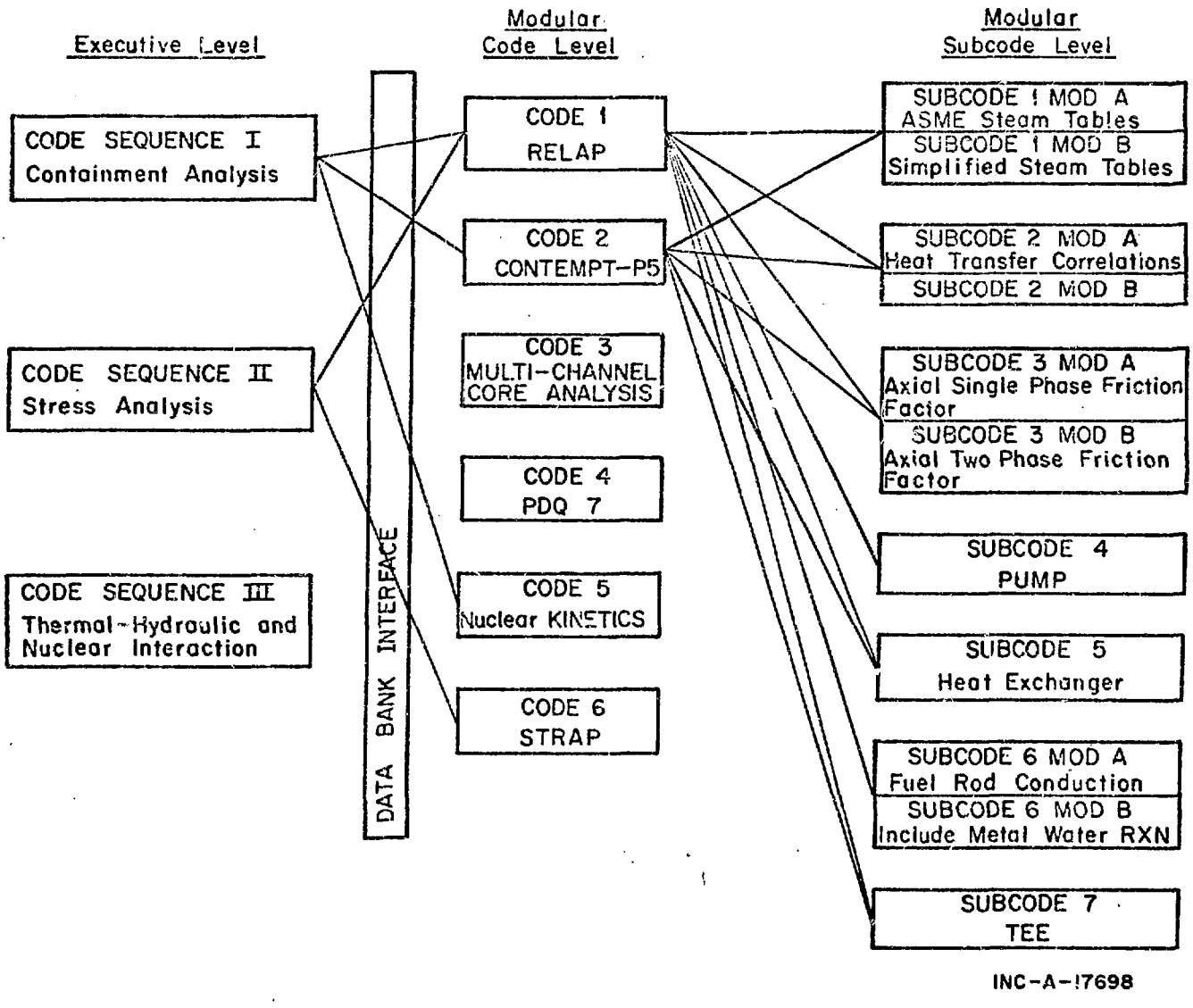


FIGURE 2 Three-Level Modular System

to the RELAP code which performs the primary loop computations to determine the flow rate and energy out of the broken loop. This code sequence would then control the iterations between these two codes. The computations in these two codes do not depend upon the computations in the CONTEMPT-PS code. This code performs the computations in the containment vessel after the flow rate and energy from the break are computed from the previous two codes. The Code Sequence I controls the transfer of this flow rate and energy information into CONTEMPT-PS. After the computations in CONTEMPT-PS are completed, control is transferred back to Code Sequence I and the calculation is terminated.

Several modules from the subcode level are represented in Figure 2. For simplicity, subcodes required by the nuclear codes are not illustrated. Different versions of the subcode modules will be available in the system. These different versions will, in general, represent different levels of exactness. For example, two versions of the steam tables are shown. If extreme accuracy is required, the ASME steam table may be used. If reduced computer time is desired, the simplified steam table will be selected.

CONTEMPT-PS requires some of the same subcode modules which RELAP3 requires. Consistency of the subcode models used by these two codes is desirable and will be insured by the code automation system. If one of these subcodes is updated, the updated version will be used by both RELAP3 and CONTEMPT-PS with no modification in either RELAP3 or CONTEMPT-PS.

Figure 2 refers to only a few code sequences, codes, and subcodes which will be included in the INCA system. For example, an alternate code sequence containment analysis consists of utilizing the energy and flow rate data stored in the data bank from previous accident analyses The flow rate would be used as direct input to the CONTEMPT-PS module. No other modular code would be used in this code sequence. A third containment analysis consists of utilizing a standard heat generation distribution in RELAP3. After RELAP3 calculates the flow rate and energy out of the break, the CONTEMPT-PS code would be called and the remainder of the run would proceed as in the first code sequence described.

In general, the executive level controls the order in which modular codes are executed as well as the specification of the data bank allocations. It also specifies or prepares the data which are to be in the immediate core. The executive level codes will be written in a combination of IBM job control language (JCL) and FORTAN statements. Input and output will usually be through the data bank. The input data may have been generated by other modular codes and the output intended for use in other modular codes. The data bank will be used to store information for later computer runs as well as information on design basis accidents. The data bank will also contain conversion codes that will edit and prepare the output or one or more codes as input for another code. Modular codes usually require that the numbers that they use and generate be in immediate core storage although there are exceptions (such as PDQ-7[10]). Modular subcodes will usually not require any storage of numbers. Modular codes and subcodes are written in standard FORTRAN language.

This section has described the general structure of the INCA system and how the varous components function. The next section will describe the general structure and characteristics of these components.

2. STRUCTURE OF THE INCA COMPONENTS

Although some of the characteristics of each level of the INCA system will be different, certain characteristics will be common to all three levels. For example, the resulting codes and code structure will be as independent of a particular computer as possible. Each of the components will be as standardized and as modular as possible. Such standardization requires that guidelines be established prior to code development. Thus, the information to be required by and obtained from a module must be specified independently of the model used to construct the module.

The following rules are to be adhered to in the development of all components in the code structure. Some of the rules in the following sections will appear to be so obvious as to be unnecessary, but experience indicates that particular attention must be given to them if the INCA system is to be successful.

(a) A standard nomenclature for FORTRAN variables will be utilized to allow the user to become quickly familiar with a code and also to aid in the diagnosis of programming errors.

(b) In order to preserve the replacement features of modular codes and subcodes, all versions of the same module must have the same name. The only method of distinguishing between different versions of the same code will be an identifying date on a comment card immediately following the FORTRAN subroutine or function statement in a module. Selection between versions will be made by specifying the computer library in which the various versions are stored.

(c) The intent and function of a modular code or subcode will be decided prior to code development. After the code or subcode has been inserted in the INCA structure, this intent and function should not be expanded or contracted. If splitting a module into two parts becomes necessary, both parts will be renamed and the original name used to form a subprogram which calls both of the newly formed modules.

(d) Only FORTRAN IV will be used so that the INCA structure will be as independent of the particular computer as possible.

(e) Clear and understandable FORTRAN programming will be used at all times.

For example, long algebraic expressions can be very difficult to interpret unless the statements are continued at logical places. For example, the FORTRAN statement

```
      RUV = 0.5* (RHO(I+1) + ROH(I))*
     1      0.5* (U(I+1) + U(I))*
     2      0.5* (V(I+1) + V(I))
```

is much more understandable than the equivalent

```
      RUV = 0.125* (RHO(I+1) + RHO(I))* (U(I
     +1) + U(I))* (V(I+1) + V(I))
```

Several different symbols will not be used for the same quantity because of resulting difficulties in understanding a program.

(f) No attempt will be made to save a small amount of computer time at the expense of complicating the programming. For example, transfer of information to a modular subcode through a FORTRAN common statement is slightly faster than through an argument list. However, use of the common statement would make the subcode nonmodular and time consuming to change. In general, if 20% of the running time of an often used code can be saved, the programming time required to increase the efficiency of the code is justified. The increase in efficiency is often obtained by the use of the symbolic language. Obviously the symbolic language of one computer is not useable on another. Thus, if the computational efficiency of a code can be increased only by converting it to a code that does not fit into the automated code structure, this may only be done if the original automated version of the code is retained and always updated before the nonautomated version is updated.

(g) The most efficient programming consistent with rules (e) and (f) will be used. For example, defining a constant within a FORTRAN DO LOOP requires much more computer time than defining the constant outside the loop. Thus, the latter procedure will be followed.

(h) Nothing will be done in the module that could adversely affect any other module in the machine. For example, a general clearing of a portion of the core should not be done because some information being used in another module could be destroyed. All zeros required by a code will be set to zero by rigorous coding.

(i) The statement numbers in each completed module will be arranged numerically and incremented by fives to allow statements to be located quickly. Each card will be numbered in columns 73 to 80 with an identification which consists of the first three letters of the subprogram name and five numbers which should increase by ten for each card to allow the insertion of other cards. A small program will be available which produces this type of deck from an unordered and unnumbered deck.

(j) A code manual will be produced for each executive code sequence, modular code, and modular subcode. One of the components of this writeup will be a flow chart for each module which describes the significant blocks in the module. Comment cards will be included in the source deck which corresponds to this flow chart.

This introduction has described the rules which will be followed in the development of all the components of the INCA system. The following sections describe the rules which will be adhered to for a particular code level.

2.1 Executive Codes

An executive code (or a code sequence) is made up of a FORTRAN main program and job control language (JCL) statements. It is used to control the sequence in which modular codes are executed and to control the flow of information from one code to another. In general, an executive code is modular in nature but is specific to the problem being solved and the computer being used. However, an executive code in FORTRAN is required for each modular code to execute that code on all computers which can accept FORTRAN.

The following rules will be adhered to in developing executive codes.

(a) The FORTRAN portion of the executive code will be small in size. Its primary function will be to specify the order in which the selected modular codes are to be run. It may include an iteration procedure and appropriate convergence testing statements to determine when the output of two codes are consistent.

(b) Conversion routines will be called by the executive code to convert output from one code to input to another code.

(c) One of the primary functions of the JCL portion of the executive code will be to select the units used for reading information from disk, tapes, and punched cards and writing information on disks, tapes, and paper.

(d) The other primary function of the JCL portion of the executive code will be to specify the overlay structure. This overlay structure is designed to optimize computer time and space and will depend upon the particular computer being used.

(e) The JCL portion of the executive code, by specifying the library source, also selects the version of a modular code or subcode that is to be used.

2.2 Modular Codes

A modular code is made up of a principal FORTRAN subroutine and associated FORTRAN subroutines and function subprograms. Modular codes behave in the same manner and have the same characteristics as a FORTRAN main program does except for the fact that the location of the input and output information of a modular code is controlled by an executive code. The modular code activates modular subcodes by an appropriate "call" statement. All of

the FORTRAN subroutines and function subprograms called need not be modular subcodes because they may be pertinent only to one modular code.

The following rules will be adhered to in developing modular codes.

(a) Modular codes will be large and could, in fact, be run completely independently of the code system. A modular code will be large enough that to define a rigid interface between it and all other modular codes in the system will be impractical. Instead, input and output will be defined as if the modual code existed alone. Conversion routines will exist in the data bank which will convert the output of one code to the input of another code. The code will be modular in nature, however, because different versions of the same modular code will require the same input and produce the same output.

(b) A modular code will store information when it is in the computer. Numbers which are generated in the code will be stored in the data bank if these numbers are required by another code or if these numbers are to be used by the same code after it has been recalled to the computer by the executive. A modular code will also supply absolute dimensions (FORTRAN dimension statements) for the subcodes which it calls.

(c) Variable dimensioning will be used in modular codes in the following manner. A modular code will be made up of at least two FORTRAN subroutines. The first subroutine is referred to as the dimension subroutine. It will supply the absolute dimensions to the principal FORTRAN subroutine. For example, variable dimensioning could be accomplished in the code ANDAR(a) with the following FORTRAN statements. (ANDARD is the dimension subroutine of ANDAR.)

```
      SUBROUTINE ANDARD
      DIMENSION TROD (1000(, RHOR (1000), TR (100)
      READ (5, 1) NA, NR .
    1 FORMAT (215)
      CALL ANDAR (NA, NR, TROD, RHOR, TW)
      RETURN
      END
      SUBROUTINE ANDAR (NA, NR, TOD, RHOR, TW)
      DIMENSION TORD (NA, NR), RHOR (NA, NR), TW(1)
      o
      o
      o
      o
      RETURN
      END
```

(a) This code is not to be included in the INCA structure but is a fictitious code used for illustration only.

In certain instances, the use of dynamic dimensioning will be advantageous in addition to variable dimensioning. Dynamic dimensioning is advantageous to increase the size of one or more arrays and decrease the size of one or more other arrays. Dynamic dimensioning could be combined with variable dimensioning in the ANDAR code with the following FORTRAN statements

```
      SUBROUTINE ANDARD
      DIMENSION POOL (2100)
      READ (5.1) NA, NR
    1 FORMAT (215)
      NTROD = 1
      NRHOR = NTROD + NA*NR
      NTW = NRHOD + NA*NR
      CALL ANDAR (NA, NR, POOL (NTROD), POOL (NRHOR), POOL (NTW))
      RETURN
      END
      SUBROUTINE ANDAR (NA, NR, TROD, RHOR, TW)
      DIMENSION TORD (NA, NR), RHOR (NA, NR), TW(1)
      o
      o
      o
      o
      RETURN
      END
```

In this example, a single array is used for allocation of computer storage. An additional array would be required if arrays of fixed point numbers are used. The arguments in the "CALL" to ANDAR include array names in a very simple and meaningful manner. For example, a name, NRHOR, is given to the initial location of the array RHOR. This naming of locations clearly identifies the array which is desired in the "call" statement. The subroutine ANDAR has been left unchanged in the transition from a variable-dimensioned program to a variable and dynamic dimensioned code. These examples illustrate the manner in which variable and dynamic dimensioning will be implemented.

The manpower required to convert an existing code to variable and dynamic dimensioning using the INCA technique will be small, few mistakes will be made, and only the dimension subroutine need be recompiled if the absolute dimensions are changed.

(d) The FORTRAN "common" statement will be used to transfer information within a given modular code but not to a modular subcode.

(e) A modular code will be capable of operating on any computer which accepts FORTRAN without any change in the coding of the module. The only exception to this rule is the dimension statement in the dimension routine which may have to be decreased for a machine with a small storage capability. Variable dimensioning makes decreasing the dimension statement a simple change and one without likelihood of errors.

(f) A standard list of test problems will be set up for each modular code. This standard set of problems will be run each time a change is made to the modular code to determine what effect the change will have upon the results.

2.3 Modular Subcodes

A modular subcode is either a single FORTRAN subroutine or a single FORTRAN function subprogram which is activated by an appropriate "call" statement from a modular code or subcode. A modular subcode is intended to perform a computational procedure which is required by several codes and which should be exactly the same in each of these codes. A good example is the computation of the friction factor in thermal and hydraulic codes.

The following rules will be adhered to in developing modular subcodes.

(a) Information will not be transferred to or from subcodes by the FORTRAN "common" statement. This rule avoids conflicts in the names of variables in the calling code and the called subcode. Any difficulty associated with long argument lists can be circumvented by formulating subcodes as FORTRAN function subprograms and using a defining FORTRAN function statement in the calling code.

Subcodes are small enough in size and specific enough in nature that such a rigid interface (the augument list) between a given subcode and all other codes is practical to define.

(b) Numbers which are calculated by the modular subcode will not be stored in the subcode. Thus a subcode may be called by any code without destroying information which may be needed at a later time. A calling modular code, a tape, or a disk may be used for storage. For example, a rod conduction modular subcode would calculate the temperature distribution at a new time from a temperature distribution at a previous time, the heat generation rate, and the boundary conditions. The new distribution will be stored in the modular code so that the same modular subcode can be called to calculate the new temperature distribution in another rod without destroying any stored information.

(c) Variable dimensioning will be used in subcodes so that arrays of different sizes can be used with the same module. Any dynamic dimensioning which is required will be taken care of in the calling modular code and will be automatically included in the modular subcode.

(d) All modular subcodes will be completely investigated off-line with a simple calling program. The desired range of all the arguments will be checked and the limitations included within the module.

(e) A standard list of test problems will be set up for each modular subcode. This standard set of problems will be run each time a change is made in the subcode. The results will be used to determine what effect the change will have upon the results.

(f) The argument list will include arguments which might be required in future versions of a subcode. Inclusion of these arguments will eliminate the difficulty of changing the FORTRAN "call" statement when a different version of a subcode is produced.

(g) A modular subcode will be capable of operating on any computer which accepts FORTRAN IV without any change in the coding of the module.

(h) Input parameters should never be used on the left hand side of an equal sign. This could easily cause a constant to be redefined in a modular code which is an extremely difficult mistake to find.

These preceding sections have described the modular concept, the specific three level structure proposed for the INCA system, and some of the detailed characteristics of the components of this system. The following section will describe the actual code sequences, codes, and subcodes which will be included in the INCA system.

IV. THE GENERAL CORPORATE CODE STRUCTURE

The development of an automated code structure requires that the intents and purposes of the components within the structure be defined to make certain that all the components fit together and that expansion can take place without disrupting the system. The first part of this section describes an example of the interrelationship between the various components of the system. For brevity, the example is confined to the development from the executive structure through the modular subcode level for the thermal-hydraulic codes.

The second part of this section provides an example of a method for classifying existing codes by Idaho Nuclear Corporation and other organizations so that comparisons can be made between appropriate codes and the assets and shortcomings of each can be ascertained. Again, for brevity, the example is confined to the thermal-hydraulic codes.

1. MODULES IN THE INCA SYSTEM

An organization chart of the modules in the INCA system is presented in Figure 3. The development of the modules in the system is consistent with the overall code structure presented in Figure 2.

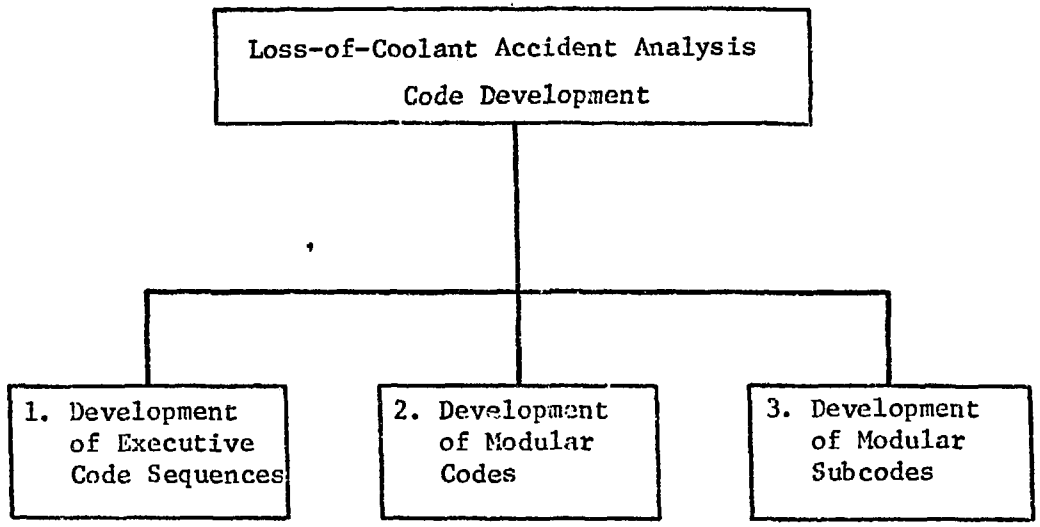


FIGURE 3 Relationship of the Modules in the INCA System

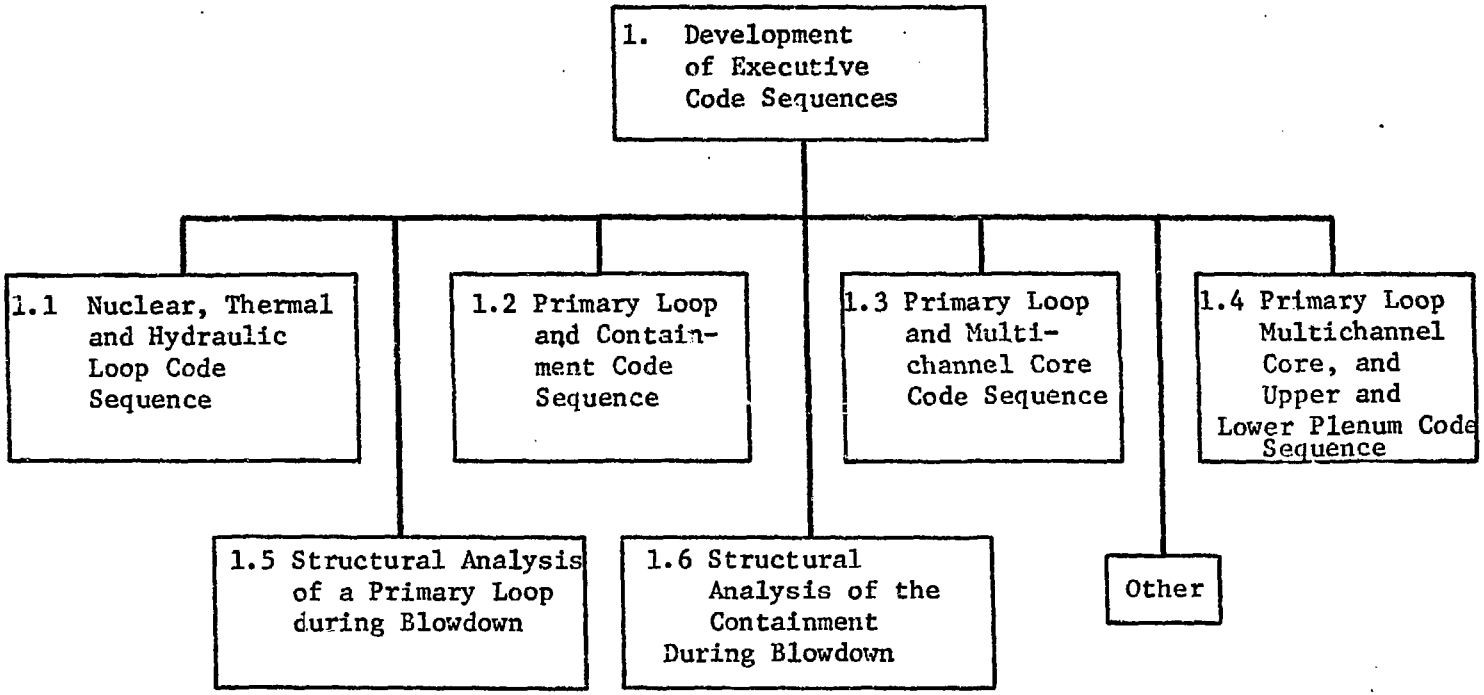


FIGURE 3 (cont.)

Plate 1

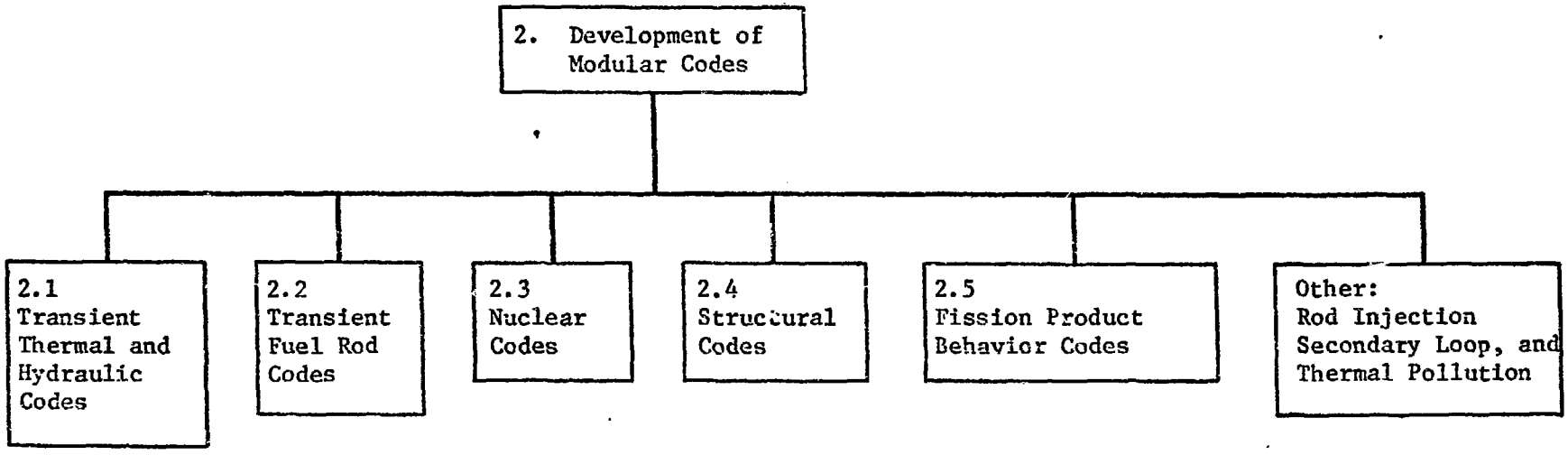


FIGURE 3 (cont.)

Plate 2

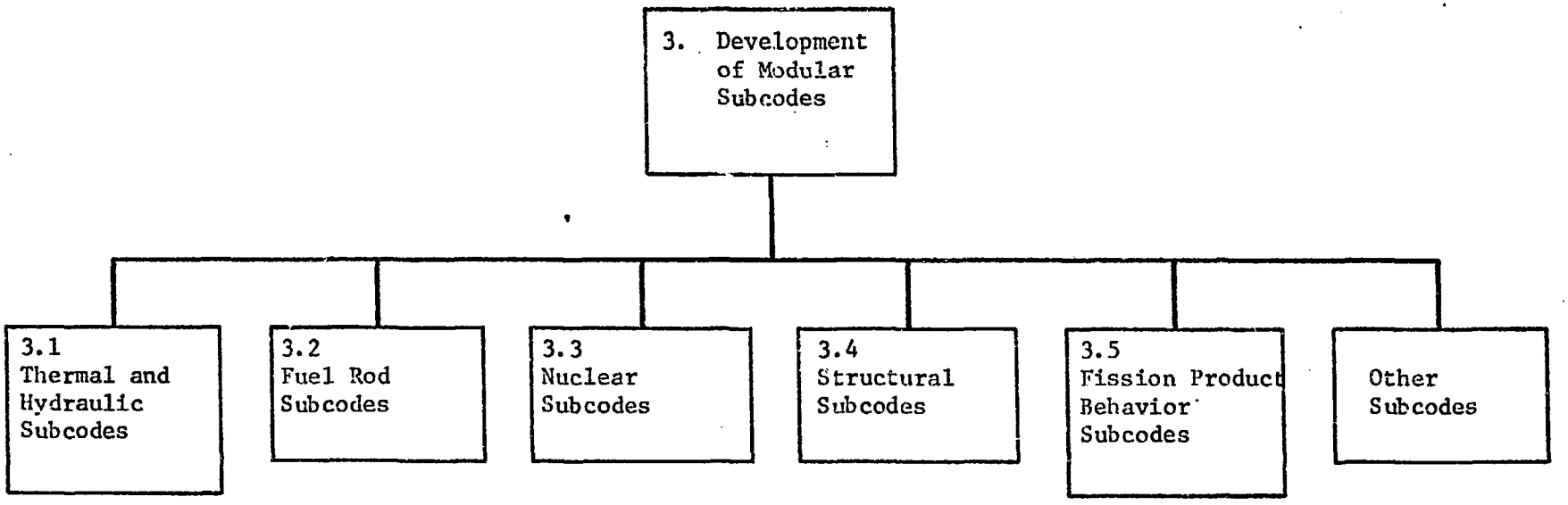
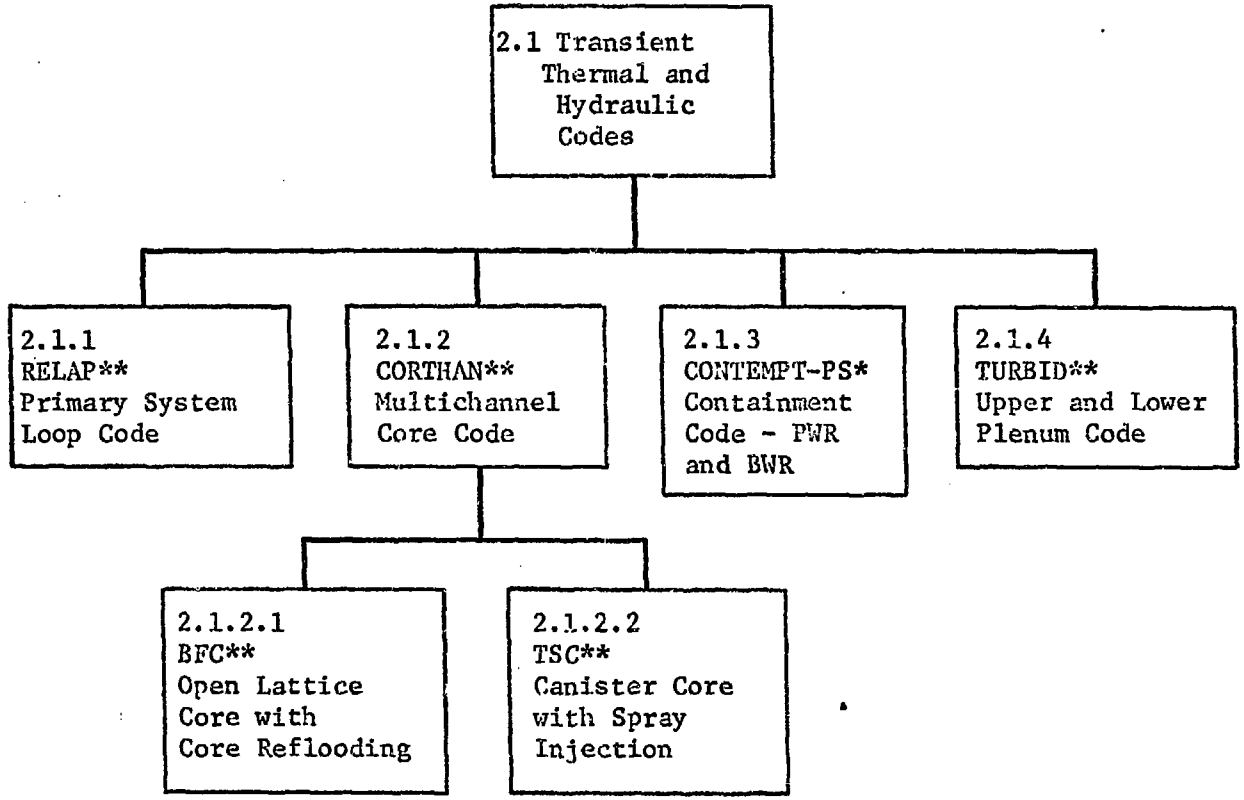


FIGURE 3 (cont.)

Plate 3

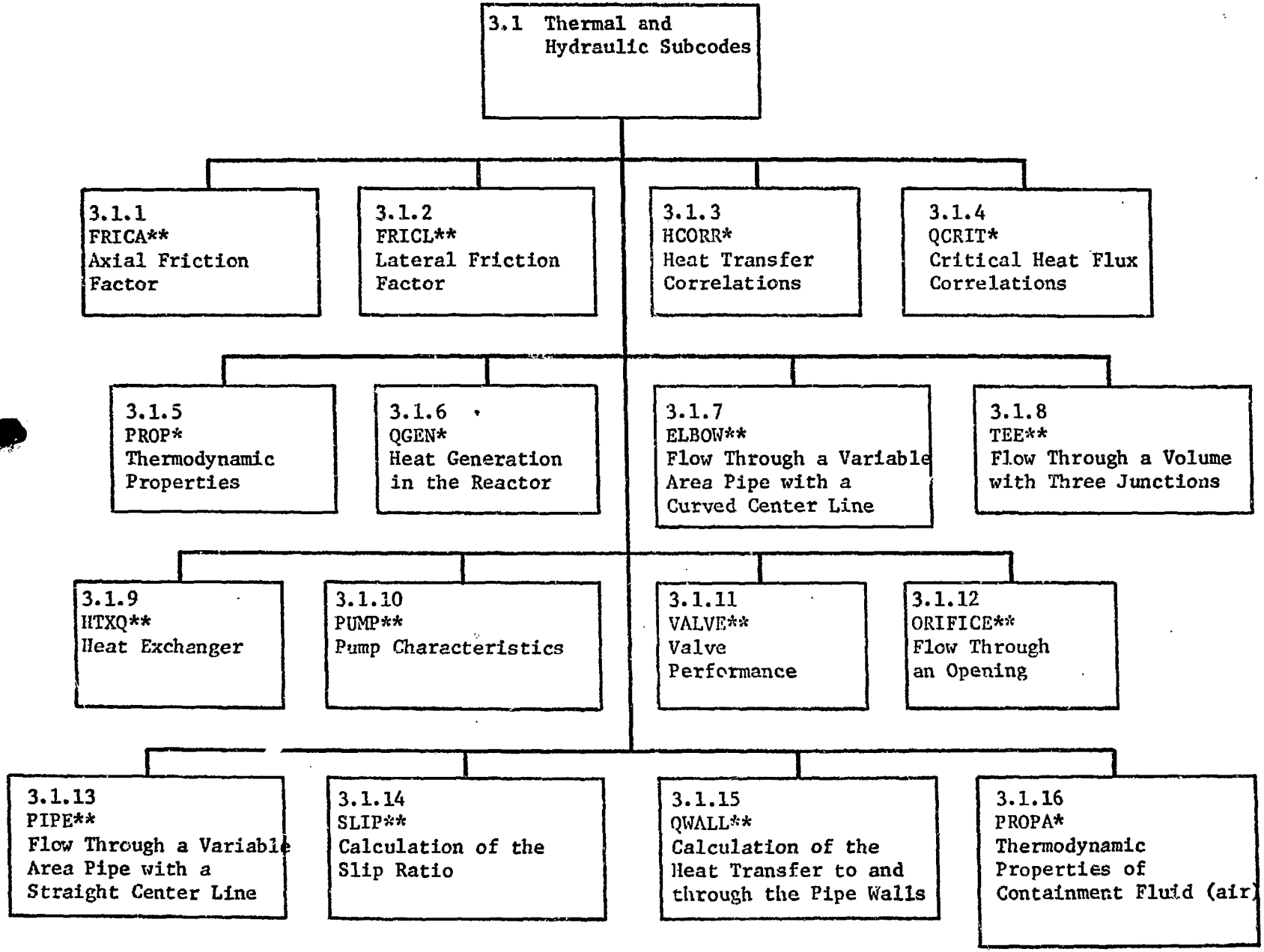


* Indicates codes which can be obtained from codes which are presently available at Idaho Nuclear Corporation.

** Indicates codes which will be developed.

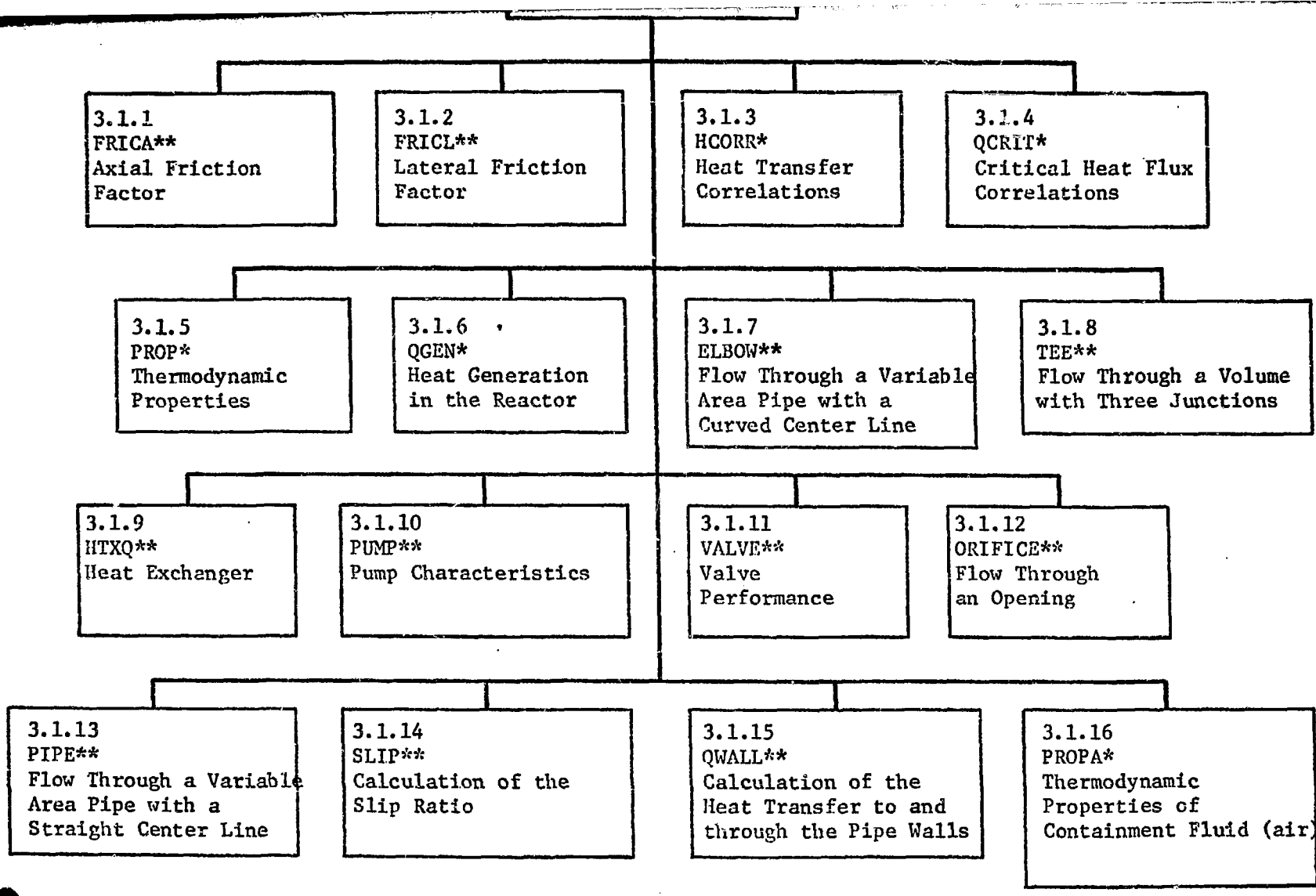
FIGURE 3 (cont.)

Plate 2.1



* Indicates subcodes which will be developed.

** Indicates subcodes which can be obtained from codes presently available at Idaho Nuclear Corporation.



2

* Indicates subcodes which will be developed.

** Indicates subcodes which can be obtained from codes presently available at Idaho Nuclear Corporation.

FIGURE 3 (cont.)

Plate 3.1

That is, development of modules will proceed along three levels (1) the executive code sequence level, (2) the modular code level, and (3) the modual subcode level. Codes which may be obtained from codes which are presently available at Idaho Nuclear Corporation are designated with an asterisk. The present names of these codes have been retained where possible to preserve the identification of well accepted codes. Codes which will be developed are identified with a double asterisk.

Some of the executive code sequences which will be developed are shown in Figure 3, Plate 1. The fact that these executive codes will be quite simple to develop is one of the principal advantages of the modular system. Thus, any code sequence which may be required in the future can be constructed quite easily from the modular codes and subcodes because the executive code merely provides storage allocation on peripheral devices and communication between codes.

The principal classification of modular codes is shown in Figure 3, Plate 2. The areas of immediate interest are the transient thermal and hydraulic codes, transient fuel rod codes, nuclear codes, structural codes, and fission product behavior codes. Item 2.1 is described in more detail in Figure 3, Plates 2.1.

The modular subcodes which are to be included in the INCA system are also classified into several areas which correspond to the modular code classification (Figure 3, Plate 3). Since any code may use any subcode, this classification is somewhat superficial. For example, a thermal and hydraulic code may call a fuel rod subcode. The subcodes, which are to be developed as thermal and hydraulic subcodes, are illustrated in Figure 3, Plate 3.1. The other subcodes indicated in Plate 3 refer to miscellaneous subcodes such as IBM scientific subroutines (for example, matrix inversion routines).

The thermal and hydraulic subcodes are illustrated in Figure 3, Plate 3.1. Some of these subcodes will be developed in the course of conducting the loop code development (RELAP). The other subcodes will be developed in a separate project. All of these subcodes are needed at the present time.

This section has described briefly the organization of the executive codes, the modular subcodes which are part of the INCA system. Modular codes in this system are comparable in certain instances to codes which have been and are being developed at other organizations. The following section provides a basis for comparing different codes.

2. CODE CLASSIFICATION STRUCTURE

Since many of the codes which have been produced by the nuclear industry are intended to serve the same objectives, the ability to compare these codes is necessary to determine which are the better codes and the better techniques. This section presents an example of a method for classifying codes to provide a basis for comparing them. For brevity, the example is confined to the thermal-hydraulic codes. This method of evaluating codes will also prove useful to the Atomic Energy Commission in evaluating the codes used by vendors in licensing reactors.

Each code will be catalogued with identifying numbers which represent its classification. For example, the thermal and hydraulic codes are described by eleven categories. The collection of numbers wich represents the class a given code falls into will be referred to as its classification set. All thermal and hydraulic codes which describe slip flow should have a number 2 for its fourth number in its classification set. Thus codes which describe slip flow can be selected very easily. In addition, slip flow codes which are also loop codes would have in addition a 3 in the sixth number of its classification set.

Pertinent categories are tabulated in this section only for the thermal hydraulic modular code development area. These categories are subdivided into classes. The number of categories and classes will be expanded in the future if required.

2.1 Classification of Thermal and Hydraulic Codes

The categories which are to be used to classify thermal and hydraulic codes are:

(a) Time state

(b) Equations of change solved

(c) Equation of state

(d) Flow state

(e) Thermal state

(f) Geometry

(g) Dimensions

(h) Boundary conditions

(i) Method of solution

(j) Restrictions necessary to insure stability

(k) Accuracy limitation

Each of these categories has several classes. If a class must be split, all previously classified codes will be listed in both classes until the class to which it belongs can be determined.

Although a thermal and hydraulic code will contain many special components, the basic nature of the code can be exhibited by explaining how this code describes the flow through a straight section of pipe of constant area. This pipe should be divided into several subvolumes or increments. This analysis should be appended to the code classification.

A code is classified according to the claims of the developer independent of whether the claims are justified or not. The code is then reviewed using these claims as a basis for comparison.

The following is a detailed break down of the previous categories.

(a) Time State

(1) Transient

(2) Steady state

(b) Equations of change solved

(1) Continuity and momentum (usually incompressible)

(2) Energy and the steady state continuity

(3) Continuity, momentum, and energy

(4) Energy and continuity

(c) Equation of state

(1) ρ = constant

(2) $\rho = \rho(H)$ or $H = H(\rho)$

(3) $\rho = \rho(P, H)$ or $H = H(P,\rho)$

(4) $P = P(\rho, H)$

(5) $dP = \left(\frac{\partial P}{\partial \rho}\right)_s d\rho$

where ρ = density, H = enthalpy, and P = pressure

(d) Flow state

(1) No slip

(2) Slip allowed

(e) Thermal state

(1) Equal phase temperatures (one temperature or enthalpy used to represent the entire temperature profile in a channel)

(2) Nonequal phase temperatures (two temperatures or enthalpies used; one represents the average vapor enthalpy, the other represents the average liquid enthalpy)

(3) Complete enthalpy or temperature distribution in a channel or pipe considered

(f) Geometry

(1) Core code

(2) Upper and lower plenum code

(3) Primary loop code

(4) Containment code

(') Secondary loop code

(g) Dimensions

(1) 1D cartesian

(2) 2D cartesian

(3) 3D cartesian

(4) Cylindrical (r, ϕ)

(5) Cylindrical (r, z)

(6) Volumetric

(h) Boundary conditions

(1) Nonloop code requiring information at inflow boundaries. For example, a core code which only requires boundary conditions at the inlet.

(2) Nonloop code requiring information at inflow and outflow boundaries. For example, a core code which requires pressure specification at the inlet and outlet of the core.

(3) Loop code which describes only a closed loop and required no boundary conditions.

(4) Loop code which can describe breaks in the line as well as TEE joints in the line.

(i) Method of solution

(1) Green's function or integral equation

(2) Completely explicit, numerical

(3) Completely implicit, numerical

(4) Combined explicit and implicit numerical; that is, some of the equations are solved explicitly and some are solved implicitly

(5) Characteristic solution

(6) Quasi-steady state

(7) Orthogonal series solution

(8) LOS ALAMOS methods, such as MAC[11] and PIC[12]

(9) Scheme not reported

(j) Restrictions necessary to insure stability

(1) Limited by the speed of sound $\Delta t/\Delta z \leq k/v_s$ where k is a constant close to 1; that is, $1/2 \leq k \leq 2$

(2) Limited by the local fluid velocity

(3) None reported

(k) Accuracy limitation

(1) Pressure calculation requires a very accurate enthalpy calculation

(2) Calculation is limited by truncation error term in one or more of the equations

(3) Depends upon evaluating derivatives in the steam tables

(4) None reported

IV. CONCLUSIONS

The system discussed provides two fundamentally attractive advantages to the AEC and the nuclear industry: (1) it will alleviate many of the past problems and errors in design and safety analyses and prevent their recurrence in present and future analyses, and (2) it will provide sound technical rationale for answering the ever present questions raised by the industry, the AEC, and the public on the conservatism built into analyses.

REFERENCES

1. Curet, H. D., et al., Loss-of-Coolant Accident Analysis Program, IN-1382 (June 1970).

2. Buccani, G., Lattori, G., Mongini-Tamagnini, C., "Caronte - The Euratom System for Automatic Control of Linked Calculations," Published in the Proceedings of the The Effective Use of Computers in the Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee, p 297.

3. Bayard, J. P., Boudet, R., "The CODNUC System A Modular System of Subroutine for Reactor Calculation," Published in the Proceedings of The Effective Use of Computers in the Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee.

4. Watanabe, T., Arai, K., Noda, T., "Hatachi Nuclear Codes Control System, NCCS," Published in the Proceedings of The Effective Use of Computers in the Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee, p 313.

5. Honeck, H. C., et al., "JOSHUA - A Reactor Physics Computation System," Published in the Proceedings of The Effective Use of Computers in the Nuclear Industry," April 21-23, 1969, Knoxville, Tennessee, p 324.

6. Just, L. C., Walker, P., Toppel, B. J., "Recent Development and Capabilities in the ARC System," Published in the Proceedings of The Effective Use of Computers in the Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee, p 337.

7. Crowther, R. L., Petrick, W. P., Weitzberg, A., "Three Dimensional BWR Simulation," Published in the Proceedings of The Effective Use of Computers in the Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee, p 344.

8. Carmichael, C. F. and Marko, S. A., CONTEMPT-PS, A Digital Computer Code for Predicting the Pressure-Temperature History within a Pressure Suppression Containment Vessel in Response to a Loss-of-Coolant Accident, IDO-17252 (March 1969).

9. Rettig, W. H., Moore, K. V., Uptmor, M. L., RELAP3 -- A Computer Program for Reactor Blowdown Analysis, IN-1321 (June 1970).

10. Cadwell, W. R., PDQ-7 Reference Manual, Bettis Atomic Power Laboratory, WAPD-TM-678 (January 1967).

11. Welch, J. E., et al., "The MAC Method, A Computing Technique for Solving Viscous Incompressible, Transient Fluid-Flow Problems Involving Free Surfaces," Los Alamos Scientific Laboratory Report, LA-3425 (November 1965).

12. Amsden, A. A., "The Particle-in-Cell Method for the Calculation of the Dynamics of Compressible Fluids," Los Alamos Scientific Laboratory Report, LA-3466 (February 1966).