Conceptual Design of Object Oriented Program (OOP) for Pilot Code of Two-Fluid, Three-field Model with C++ 6.0

Bub Dong Chung, and Young Jin Lee
Korea Atomic Energy Research Institute, 150 Dukjin-dong, Yuseong, Daejeon, 305-353,Korea.
bdchung@kaeri.re.kr, yjlee1@kaeri.re.kr

1. Introduction

Engineering software for design purpose in nuclear industries have been developed since early 1970s, and well established in 1980s. The most popular and common language for the software development has been FORTRAN series, until the more sophisticated GUI and software coupling is needed. The advanced computer language, such as C++[1], C# has been developed to help the programming for the easy GUI need and reuse of well developed routines, with adopting the objective oriented program. A recent trend of programming becomes objective-oriented since the results are often more intuitive and easier to maintain than procedure program.

The main motivation of this work is to capture objective oriented concepts for conventional safety analysis programs which consist of many functions and procedure oriented structures. In this work, the new objective programming with C++ 6.0 language has been tried for the PILOT[2],[3] code written in FORTRAN[4],[5] language, and conceptual OOP design of the system safety analysis code has been done.

2. Method and Results

The PILOT code is a solver for one-dimensional two-fluid three field thermal hydraulics. It was designed as a procedure oriented program, and consists of many functions. Object-oriented programs can be designed by laying out the network of objects with their behaviors and patterns of interaction and by arranging the hierarchy of classes. Thus there is structure both in the program's activity and in its definition.

PILOT code has a basic solution structure to solve the finite difference equations derived from three field hydraulic equations. The whole system can be considered as a network of volume and junction objects. The general safety analysis code may be considered as an assembly of more objects, such as heat structures, reactor kinetics, and control. On the other side, the solution scheme needs a strict order of functional process. The top function has been designed as procedure oriented method for the realization of the whole solution scheme. The elements which can be classified as individual behavior have been expressed as objects. The details of members and functions of each element (or class) will be designed later.

2.1 Top functional procedures

The conventional functional processes are file-open[FileOpen()] and input processing[InputProcess()]. After the initialization [Init()] , transient process will be followed and repeated until the end of time. The transient process consists of functions for time step control [DtStepControl()], heat structure calculation [HeatStrAdvance()], hydraulic solver [HydroSolve()], reactor kinetics [ReactorKin()], and control and trip [Control()].

2.2 Design of Classes

Reactor thermal system is considered as an assembly of hydraulic loop and heat structure objects. Each hydraulic loop consists of basic objects, such as volume and junctions. The necessary components can be derived from volume or junction. To get whole system, the reactor kinetic and control& trip objects should be added. Class is blue print of each object, and described in the follows.

2.2.1 Volume Class

Volume class is a basic class of system. It contains volume geometry data and properties as member variables. Some members such as velocity, hydraulic diameter should be described as array variables to use for multi dimensional code. The design of member functions is relatively subjective. One should decide whether specific function will be realized in volume class or not. Good example is the functions for steam properties. Table 1 shows the member functions design for PILOT code.

Table1. Member function in volume class (PILOT)

Member	Function
void init()	Volume class initialization
void vavg()	Calculation volume average velocity
double Hiff()	Liquid side interfacial heat transfer
double Higg()	Gas side interfacial heat transfer
void cell_matrix()	5x5 volume cell matrix coefficient
void cell_press()	5x5 volume matrix inverson, system pressure matrix
void eq_back()	$5(\delta U_v, \delta U_f, \delta\alpha_v, \delta\alpha_d, \delta P)$ calculation
void update()	new variable update of old variable

void correct()	Convervative form correction

2.2.2 Junction Class

Junction class is another basic class of hydraulic system. It contains the junction geometry data and properties. The necessary function to solve the momentum equation also contains in junction class as shown in Table 2.

Table 2. Member functions in Junction Class (PILOT)

Member	Functions
void init()	Junction class initialization
void Properties()	Junction dornoed property calc.
double fwall_d()	Droplet flow wall drag coefficient
double fwall_v()	Vapor flow wall drag coefficient
double fwall_l()	Liquid flow wall drag coefficient
double fvl()	Vapor-liquid interfacial drag
double fvd()	Vapor-droplet interfacial drag
void cell_matrix()	explicit velocity with 3x3 cell matrix
void eq_back()	Final velocity calc. from volume P
void update()	Old variable Update

2.2.3 Component Class

Component class is important feature of hydraulic system, since all special functions required for system code should be realized from special component model. The design of these component can be varied depends on the subjective decision. We can design Pump class inherited from basic Volume class and Valve class inherited from basic Junction class. Once the special component class has been derived, the special component function could be added or override the base class function. This feature makes the encapsulation of the special function within special component class.

2.2.4 Loop Class

Loop class contains the pointer for volume and junction objects. It also has a pointer for steam table use. The overall hydraulic solver and boundary function should be included as loop member.

2.2.5 Other Classes

Heat Structure, Reactor Kinetics and Control&Trip classes should be designed to complete the system. The heat structure class needs the connections to volume object as pointer. Reactor kinetics needs also the volume and heat structure pointers to get reactivity feedback. The general control class needs connection pointers to all classes. The conceptual structure of the overall system code is shown in Figure 1.

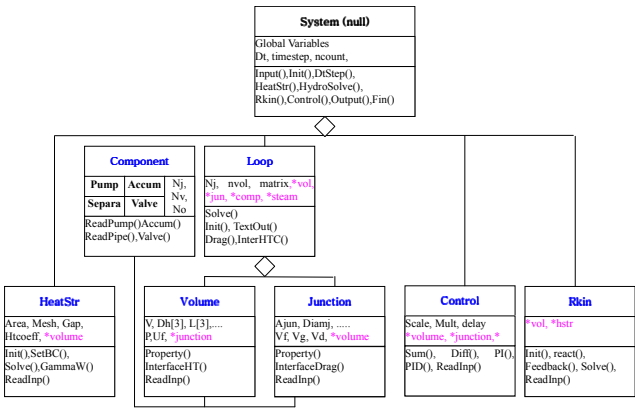

Figure 1. Conceptual Structure of system code

3. Conclusion

The PILOT code was designed with OOP concept and re-constructed with C++ objective language. It was demonstrated that basic thermal hydraulic solver can be realized with OOP concept and each functions can be well encapsulated within basic classes. Once the function is realized within class, there are many beneficial aspects in both development and maintenance. However the actual design of system code needs top design of functional procedure as well as realization of bottom classes. Another negative aspect is that there are too many connections between objects. It makes the program difficult to read, however the complication is inevitable and resulted from the system code characteristics. The objective program also needs many subjective decisions before the design of class. The complete analysis of whole code system is essential for the success of the objective oriented program.

REFERENCES

[1] Visual C++ 2005 Express Version Reference Manual, MS Company (2005)
[2] J.J Jeong, et. al., “Basic Pilot code Development for two fluid, three field model”KAERI/TR-3151/2006 (2006).
[3] M.K. Hwang et. al., “Development and Verification of a Pilot Code based on Two-fluid Three-field Model”, KAERI/TR-3239/2006 (2006)
[4] Compaq Visual Fortran 6.6 Version Language Reference Manual, Compaq Co. (2001)
[5] John Reid, “The New Features of Fortran 2003” ISO/IEC JTC1/SC22/WG5 N1579, Fortran Standard Technical Committee, http://www.j3-fortran.org/