Objective Oriented Design of System Thermal Hydraulic Analysis Program and Verification of Feasibility

Bub Dong Chung, Jae Jun Jeong and Moon Kyu Hwang

Korea Atomic Energy Research Institute, 150 Dukjin-dong, Yuseong,Daejeon, 305-353,Korea.
bdchung@kaeri.re.kr, jjjeong@kaeri.re.kr , mkhwang@kaeri.re.kr

1. Introduction

The system safety analysis code, such as RELAP5, TRAC, CATHARE etc. have been developed based on Fortran language during the past few decades. Refactoring of conventional codes has been also performed to improve code readability and maintenance. TRACE, RELAP5-3D and MARS codes are examples of these activities. The codes were redesigned to have modular structures utilizing Fortran 90 features. However the programming paradigm in software technology has been changed to use objects oriented programming (OOP), which is based on several techniques, including encapsulation, modularity, polymorphism, and inheritance. It was not commonly used in mainstream software application development until the early 1990s. Many modern programming languages now support OOP. Although the recent Fortran language also support the OOP [1], it is considered to have limited functions compared to the modern software features. In this work, objective oriented program for system safety analysis code has been tried utilizing modern C# language feature [2]. The advantage of OOP has been discussed after verification of design feasibility.

2. Methods and Results

In this section, the general four steps of system code development are described. The steps are consists of analysis step, design step, implementation step and verification step.

2.1 Analysis Step

This step requires the derivation of class based on the technical specification of software requirements. All noun terms of the specification document are possible to be classes. Typical nouns in the specification are volume, junction, heat structure, property, heat structure, components, trip, control and kinetics. All of them are possible to be classes. The solver is considered an assembly of behavior of system and excluded from classes. The derived classes [3] are presented in Figure 1. Specialized component classes are derived from the inheritance of common component class.

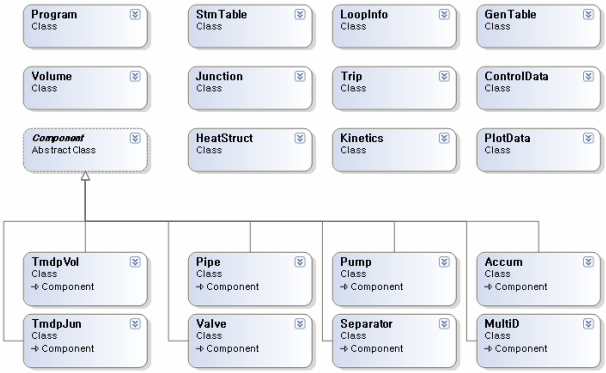

Figure 1. Derived Classes for the System Analysis Code

2.2 Design Step

Each class should be designed to contain unique variables and functions, which is needed to solve each behavior of separated objects. System analysis code is a program which solves the complex interaction between thermal hydraulics, components, trip and control and kinetics. Many classes should refer other objects and connected with each others. Volume class refers junction object and junction class also refers volume object. All component class refers the trip, control, and general table objects to represent the behavior of component. Kinetics class is designed to refer volume and heat structure objects to get feed-back effect. Trip and control class should be designed to refer any class according to the user input.

2.3 Implementation Step

All functions needed to system analysis program should be implemented in main program class, and all global variables such as file names, time step and objects are also contained in main program. The whole procedure consists of the input process, initialization and transient run. During the implementation of each procedure, the member functions of each class were used if it can be separated from main program class. For example, the component behaviors belong to each component objects and volume model and correlations belong to volume object only. The

corresponding functions are implemented as class member functions. The ArrayList and List collection, which is a unique feature of C#, have been used to contain the various objects. The hydraulic system contains loops, and each loop contains the components. The Pilot code, PiCod-TF [4, 5] has been used as hydraulic solver, although many parts were modified to generalize for branching and boundary treatment. Simple special model, such as critical flow and component models such as trip valve, accumulator, and separator model, have been implemented. Simple steady solver for 1D heat structure has been also added. The resulting code, SYSTF is flexible enough to be able to model NPP behavior.

2.4 Verification Step

The main goal of SYSTF is to provide the architecture of OOP system code, and thus the verifications are only limited against conceptual problems. The verifications are performed to check step by step. As a primary step of verification, the code has been verified many simple cases. The test cases include the single phase pipe problem, single phase network problems, two phase pipe problems, multi-loop network problems, other special model problems. Many fundamental performances have been checked during the simple tests cases. The integrated verifications were performed against the conceptual plant modeling as shown in Figure 2. Steady state, non LOCA and LOCA transient calculations were performed to check integrated feasibility and thermal hydraulic behavior.

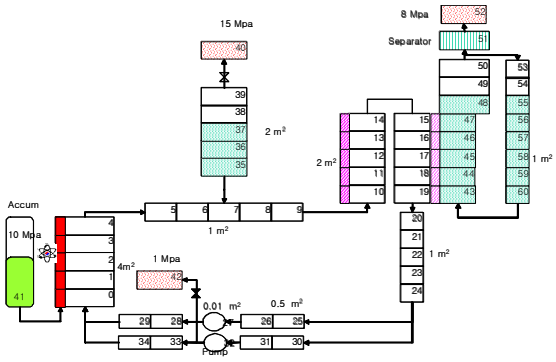

Figure 2. Conceptual Plant Model

Figure 3 shows the Window GUI of SYSTF, and presents the system transient during hypothetical LOCA event. The typical transient behaviors such as break flow, secondary separator model, pump trip, flashing and void generation, accumulator and valve actuation were simulated reasonably with no problems. SYSTF code features meet all the major functional requirements as a system analysis code.

3. Conclusion

The feasibility of OOP implementation for system analysis code has been validated through the various tests. The results show the advantage of OOP in the implementation of models. The component model associated with volumes and junction can be modeled in class level. The physical correlations can be also modeled as volume class level. The functional modularity for components and correlations was enhanced significantly in the system code. The implementation work with other models was much easier than procedure program. Although it is difficult to achieve the agreed and rigorous definition of in the analysis and design step, the benefit in implementation and verification step could compensate the classification effort. Reduction of on-going maintenance cost is surely to be one of big benefits achieved by OOP [6].

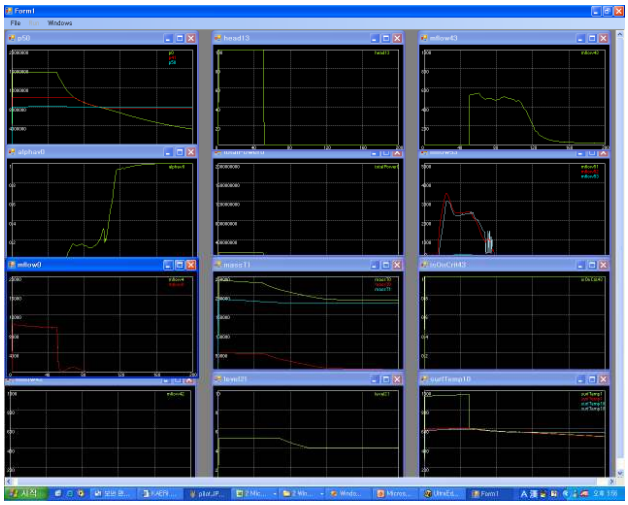

Figure 3. LOCA Transient Behavior

REFERENCES

[1] John Reid "The New Features of Fortran 2003", WG5 Convener, ISO/IEC JTC1/SC22/WG5 N1579, 2003, http://www.nag.co.uk/sc22wg5/
[2] Visual Studio 2008 Express Version Reference Manual, MicroSoft Co., 2008
[3] B.D.Chung et al., "Conceptual OOP design of Pilot Code for Two-Fluid, Three-field Model with C++ 6.0", KAERI/TR-3241/2006, 2006.
[4] J.J.Jeong et al, "Basic Pilot Code Development for Two-Fluid, Three-Ffield Model", KAERI/TR-3151/2006, 2006.
[5] M.K.Hwang et al., "Development and Verification of a Pilot Code based on Two-fluid Three-field Model", KAERI/TR-3239/2006, 2006.
[6] B.S.Moskowitz, et al." Modular, Objective-Oriented Redesign of a Large Scale Monte Carlo Neutron Transport Program", B-T-3292, Bettis Atomic Power laboratory, 2000.