A Restructuring of the HS Package for the MIDAS Computer Code

S.H.Park a, K.R.Kim a, D.H.Kim a, S.W.Cho b
a T/H Safety Research Division, KAERI, P.O.Box 105, Yusong, Daejon, 305-600 Korea, shpark2@kaeri.re.kr
b Korea Radiation Technology Institute Co. Kusongdong 19, Yusong, Daejon, 305-353 Korea

1. INTRODUCTION

As one of the processes for a localized severe accident analysis code, KAERI is developing a severe accident code MIDAS, based on MELCOR. The existing data saving method uses pointer variables for a fix-sized storage management, and it deteriorates the readability, maintainability and portability of the code.

But new features in FORTRAN90 such as a dynamic allocation have been used for the restructuring[1,2]. The restructuring of the data saving and transferring method of the existing code makes it easy to understand the code. Before an entire restructuring of the code, a restructuring template for a simple package was developed and tested. The target for the restructuring was the HS package which is responsible for calculation the heat conduction within an intact, solid structure and energy transfer across its boundary surfaces into control volumes. The verification was done through comparing the results before and after the restructuring.

2. Existing Structure

The MELCOR code is composed of 3 parts: MELGEN which checks the input data and makes the restart file needed for the calculation, MELCOR which calculates with time using the restart file and makes a log/plot file, and the PLOT- related programs.

Each part has information including the physical packages such as COR, HS, SPR, FL, CVH, DCH, RN1, RN2, and shares the data through each package, with dozens of subroutines per each package, and all the hundreds of subroutines which record the messages and control the execution flow.

At first the data saving and transfer method is analyzed[3]. The subroutines which read / write the restart file are MXXRS and MXXRSW, and they read or write according to the entry conditions. Through this process, arrays for 4 data types of real, integer, logical, and character are read and/or written for each package.

MELCOR reserved and used 4 data types for the data storage and transfer effectively. Data is saved in the most effective way within a fixed array and it is transferred through 2 steps. In the first step the information of each package for each data type (the start location and the size) is conveyed to the next step. At the second step pointers and number variables used within the packages are transferred as arguments, and they are used as local variables.

To analyze the pointer variables, comments and arguments passing within the subroutines which are related to data movements are analyzed. The pointer variables of the first step are applied similarly to all the packages and they are included as a common block named 'HSDB' for HS package within several subroutines, and use 2 variables per data type. The second step pointer variables are applied differently according to the packages, defined in the common blocks named HSPNT, HSPNTR HSDGPT, HSFTPT and HSRDPT. The pointer variables of the HS package can be found in a common block within the subroutines of the HS packages, which point out a specific location among the database of 4 data types.

The contents that the pointer variables convey can be searched in the subroutines within the packages. Based on these contents, the module is constructed for the HS package. Also the HS data was found to be referenced in the BH, COR, FDI, and RN1 packages besides the HS packages.

3. DEVELOPMENT OF THE RESTRUCTURED HS

Before an entire restructuring of the code, a restructuring template for a simple package was developed and expanded into the entire MELCOR code[4,5]. FORTRAN90 was used to apply its specific features to the code restructuring. The data transfer structure in the HS package was modularized from a single array into a derived data type for an easy understanding and usage. Also the variables used in the HS package were transformed into a modularized type, and their efficiency was increased through a dynamic allocation.

100 subroutines in the HS package and 200 subroutines in the other packages using the HS data were analyzed, through a minimum manual job using an automatic conversion program(MeltoMid) related to the subroutines that were restructured[6].

3.1 Construction of the Module

In MELCOR, the subroutine HSDBD, which is the second routine of the database manager level is, connects with subroutine HSRUN0. The subroutine HSRUN0 then applies the actual-meaning variables instead of the pointer variables. In the restructured HS, however, these pointer variables were removed by using the module HS_MDL which was constructed based on the analysis of the pointer variable's usage. The ALLOCATABLE statement in the module allowed for a dynamic storage allocation.

3.2 Subroutine Reorganization

In advance of the restructuring of the subroutines, the whole existing MELCOR code written in FORTRAN77 was transferred into FORTRAN90 like the restructuring process of the other packages. It was confirmed that the results before and after a conversion were the same. Among the subroutines in the HS package, only 51 subroutines were found to use the pointer and number variables, and the others used the local variables for data tansfer. All the pointer and number variables in the 51 subroutines were transformed into meaningful variables defined in module HS_MDL.

In the newly constructed module, most of the array variables were transformed into member variables which has a meaning for the implication. The comparison of the variables before and after a modification is shown in Table 1.

In subroutine HSDBD the pointer variables in an argument are removed and in subroutine HSRUN0 the direct variables are used instead of the local variables.

The HS data which is the target restructuring data used in the BH, COR, FDI, RN1 packages besides the HS package, are also analyzed and restructured.

Table 1. Contents of the array change

Usage in existing method
IBVL(NUMHS)
IBVR(NUMHS)
IFLOL(NUMHS)
IFLOR(NUMHS)
MATSRC(NNMSRC)
NCRDG(NNMSRC)
DGENGO(NNMSRC)
ICFRAD(2,NNPAIR)
IHSRAD(2,NNPAIR)
IFTIML(NNMFT1)
IFTIMR(NNMFT1)
Usage in new method
HS_HS(NUMHS)%IBVL
HS_HS(NUMHS)%IBVR
HS_HS(NUMHS)%IFLOL
HS_HS(NUMHS)%IFLOR
HS_SR(NNMSRC)%MATSRC
HS_SR(NNMSRC)%NCRDG
HS_SR(NNMSRC)%DGENGO
HS_PA(NNPAIR)%ICFRAD(2)
HS_PA(NNPAIR)%IHSRAD(2)
HS_F1(NNMFT1)%IFTIML
HS_F1(NNMFT1)%IFTIMR

4. RESULT AND VERIFICATION

In order to verify the new results, a two-step process was done. At first, a simple language conversion process from FORTRAN77 to FORTRAN90 was checked by comparing the major variables in the HS package before and after a conversion. To compare the results, the core fuel parameter, characteristics of core nodalization, flow path and containment volumes are added to the input file. Some sequences were calculated such as the steady state and SBO (Station Blackout) accident. The comparison was performed in a nuclear power plant of 4300MW_t until 15,000 sec and in APR1400 until 80,000 sec. The major variables were the same. Therefore the language conversion was confirmed to be successful.

The next step, which was the main part here, which was to verify the new HS results against the current results. As only the HS package was restructured using the module and the derived-type variables, the interface program for the data communication between the restructured HS package and the other original packages was developed. The interface program was checked by comparing the values of the HS package variables with the original values, and it provided the same values for the HS package variables after the read/write processing.

5. CONCLUSIONS

To restructure the data storage and transfer scheme, the data management process was analyzed for the entire MELCOR code. Especially for the HS package, the data information and pointer variables were analyzed, and the modules were configured and applied to the HS package. In this procedure, the data structure was restructured to remove the pointer variables by using the direct variables through the FORTRAN90 features such as the MODULE and USE statements.

Using the reconstructed modules, the subroutines in the HS packages were restructured. By comparing the important variables in the HS package, it was confirmed that the trends were the same but differences appeared in the CPU time comparison. They are supposed to be caused by an array transformation in some subroutines within MELCOR and a different choice of the computational path according to an extremely little value difference. Other possibilities were also checked. Based on the various results of the similar trends, the output of various executing results, and consulting other experiences, it has been confirmed that the HS restructuring was done right.

Therefore, the propriety of the restructuring applied to the HS package was verified. Through the restructuring process, the base was constructed for an easy grasp of the variables everywhere in the code and it became easy for a code improvement and for an addition of new models. The restructuring process proposed in this paper will be extended to the entire code for the MIDAS development.

REFERENCES

[1] A Multi-Dimensional Thermal-Hydraulic System Analysis Code, MARS 1.3.1, Vol.31, Number 3, pp.344-363, June 1999.

[2] S.H.Park, H.D.Kim, D.H.Kim, Y.M.Song, B.D.Chung, Development of Restructuring Template for MELCOR, 5th PSAM Conference, Japan, 2000.11.26 – 2000.12.2

[3] D.H.Kim, S.H.Park, Analysis of MELCOR Code Structure, KAERI/TR-1543/00, June, 2000.

[4] D.H.Kim, S.H.Park, Y.M.Song, A Restructuring Proposal Based on MELCOR for Severe Accident Analysis Code Development, KAERI/TR-1536/2000, March, 2000

[5] D.H.Kim, et al., Experimental and Analytical Research on Severe Accident Phenomena, KAERI/RR-2216/2001, May, 2002.

[6] Y.M.Song, S.H.Park, D.H.Kim, Development of a Computer Program for Automatic Variable Conversion in MELCOR Code, Proceedings of the Korean Nuclear Autumn Meeting, 2000.