JAVA-BASED INTEGRATING SIMULATOR FOR BLAST FURNACE

Masanobu Koga*, Masatoshi Ogawa, Harutoshi Ogai**, Masahiro Ito*****
Kenko Uchida**** **and Shinroku Matsuzaki*****

**Department of Control Engineering and Science*
Kyushu Institute of Technology,
680-4 Kawatsu, Iizuka City, Fukuoka, 820-8502, Japan
koga@ces.kyutech.ac.jp
***Waseda University*
2-7 Hibikino, Wakamatsu, Kitakyushu, Fukuoka, 808-0135, Japan
****NIPPON STEEL CORPORATION,*
20-1Shintomi, Futtsu City, Chiba, 293-8511, Japan
*****Waseda University,*
3-4-1 Okubo, Shinjuku, Tokyo 169-8555, Japan

Abstract: The method of integrating many partial simulators for a complicated system in the consideration of interaction is proposed. Java Native Interface is used to integrate the simulators programmed in different languages. Remote Method Invocation is used to integrate the simulators in different computer environment. By applying the method to the partial simulators on the blast furnace, java-based integrating simulator for the blast furnace is constructed in order to support the operation of operator. Developing the simulator by the method can save considerable labour and decrease time consumption greatly than redeveloping. Furthermore, the visualization system comparing both the simulation results and the actual measurement data is reported.

Keywords: Simulators, Compatibility, Distributed simulation, Process simulators, Steel industry, Object oriented programming

1. INTRODUCTION

Recently, rationalization and enlargement of production facilities in various industrial processes are performed. The blast furnace process in steel industry is upsized to ensure a stable supply of a large amount of low-cost hot metal. The blast furnace processes is complicated, because three phase of gas, liquid and solid in container interact together. Furthermore, the amount of using the cheap fuel increases, because the fuel of the high quality is insufficient. Thus, the trouble in facilities and operation increases. Therefore, the development of system that supports the operation of operator by predicting the blast furnace operation is demanded.

On the other hand, a system of object to be controlled is complicated, the improvement in performance and safety of the system is required, and the demand that simulates the entire system is increasing. Formerly, only a part of system for large scale system was modelled and the partial simulation was performed because of restrictions of computer performance. If the entire system simulation is performed by integrating the existing partial simulators, there will be a merit which can save labour and time considerably more than redeveloping the simulator. On the contrary, since their partial simulators had

been developed using different programming languages and computer environment, their compatibility were weak.
The examples of the partial simulators are shown in Fig.1.

In this research, the method of integrating two or more partial simulators in consideration of the interaction of the simulators for a complicated system is proposed. As the example, this method is applied to two partial simulators of both Rabit model (burden distribution model) and Bright model (internal model) in blast furnace process. Accordingly, an java-based integrating simulator for blast furnace is constructed in order to support the operation of operator. In addition, the simulation results are shown as visual information by a visualization system. The visualization system provides visual information to compare the simulation results with an actual measurement data. The comparison of both simulation results and measurement data during one-month by the visualization system is shown.

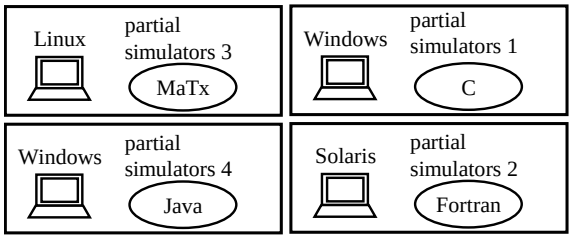

Fig.1. The examples of the partial simulators

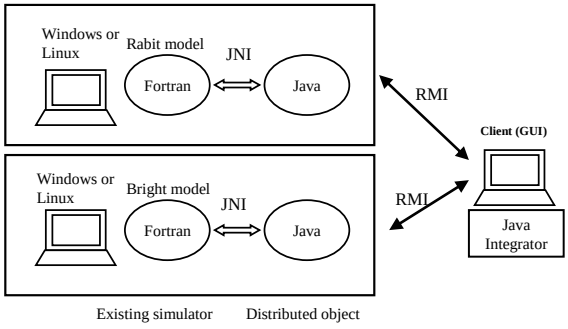

Fig.2. Integration of Bright and Rabit by JNI & RMI

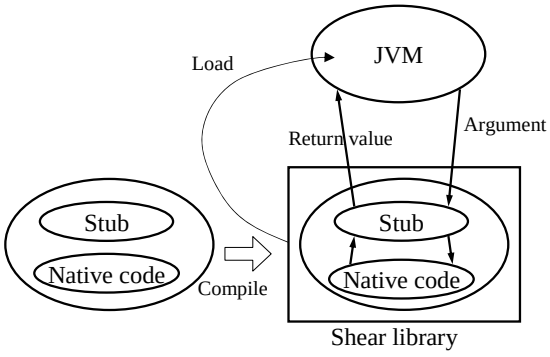

Fig.3. The conception of JNI

2. INTEGRATED METHOD OF SIMULATORS

Remote Method Invocation (RMI) and Java Native Interface (JNI) to integrate the two or more partial simulators in proposed method are used. RMI is used to integrate those partial simulators executed in different computer environment. JNI is used to integrate those simulators programmed in different programming languages. The conception of integration of partial simulator by these methods is shown in Fig.2.

2.1 Java Native Interface (JNI)

JNI is a structure to execute native code, C, C++, or Fortran code from Java code. JNI is described in (Gordon, 1998). JNI can execute the following in native code.

- An object is generated.
- A class variable is accessed.
- An instance variable is accessed.
- An exception is generated or caught.
- A class is loaded.

For using JNI, native code and stub (a few lines consist of native code) which intercedes native code are compiled. Consequently, a share library is made. The share library is SO file (Share library) on the UNIX platform or DLL file on the Windows platform. The function of native code (C code) is called through the Stub. Moreover, the execution results are returned to a Java program through the Stub. The conception of JNI is shown in Fig.3. JNI can execute Fortran code from Java by calling Fortran subroutine from C function.

Next, the procedure to access Fortran and C code from Java by using JNI in the following is shown.

(1) Export from Fortran and making a share library.

The export subroutine names of Fortran code to be used from Java by using JNI are declared. Next, the labelled Common block variables of Fortran code is declared to be used from Java code. Moreover, the Fortran code is compiled to make a share library.

(2) Use Fortran code from C code.

A C function to call the subroutine in Fortran code from C is made. In addition, The C function is defined in the form that can be called from Java. Some C structures that correspond to the labelled Fortran Common blocks in header file of C are defined. The extern of the variables referring to the memory domain of Fortran is declared. Therefore C program can access the Common block variables of Fortran by accessing the C structure member.

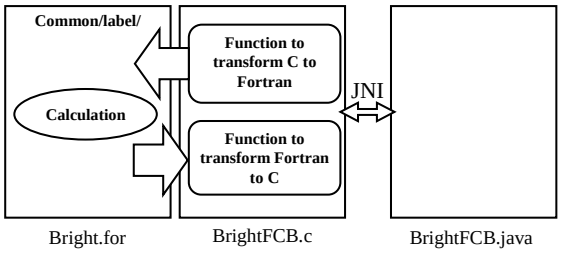

Fig.4. Access method to Common Block

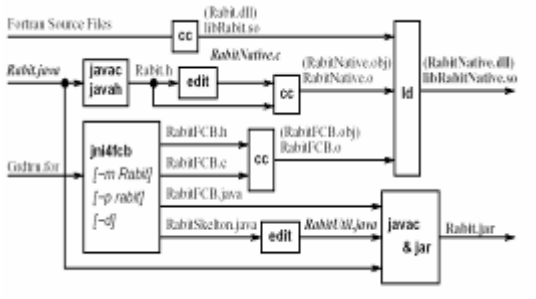

Fig.5. Outline of the procedure to create Java program which accesses Fortran code by Jni4FCB

(3) Use C and Fortran code from Java code.

The native method for the function of C that will be used from Java is declared in Java code. Moreover, share library which contains the C code in static block is loaded. By compiling the Java code, class file are created.

A C header file containing information on C function from Java class file by using *javah* commands is created. C code which contains function implemented by function prototype described in the header file is developed. A share library to load from java integrated both the share library which is created by compiling the C code and the share library which is created from Fortran code is made. In Fig.4, the conception for accessing the Common Block variables of Fortran from Java is shown.

Java program access the variables of Fortran Common Block through the C structure corresponding to the Java class. The tool to enable Java program to access Fortran is developed in the present work, is called Jni4FCB (JNI for Fortran Common Block).

2.2 Jni4FCB (JNI for Fortran Common Block)

Jni4FCB is a useful support tool that is developed in the present work. By using the Jni4FCB, Java program can access a Common Block variable of Fortran. Furthermore, the Jni4FCB generates automatically the interface programs to access the variable. If Common Block variables are declared, almost interface programs to access them from Java are created. The procedure to make Java program which accesses Fortran code by Jni4FCB is shown in Fig.5.

2.3 Remote Method Invocation (RMI)

RMI provides the environment to use distributed objects in java. RMI is described in (Nakayama, 2000; Cay and Gary, 2002). The feature of RMI is shown in the following.

- It uses the objects which are arranged at terminals on network.
- It exchanges data by calling a method among the terminals through the distributed objects on network.

The process flow of RMI is shown in Fig.6 and the following.

1. The server generates a distributed object.
2. The server registers the distributed object to name server.
3. The client queries the distributed object to name server, gets reference of the distributed object.
4. The client calls a method of the distributed object.

2.4 Distributed computing environment for Rabit & Bright

The process flow of distributed simulation environment by using RMI and JNI is shown in Fig.7.

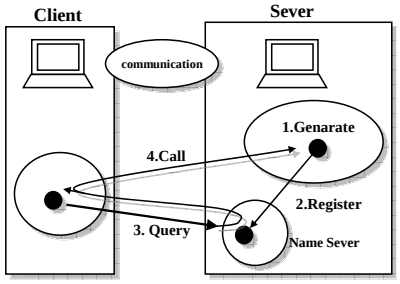

Fig.6. Process flow in RMI

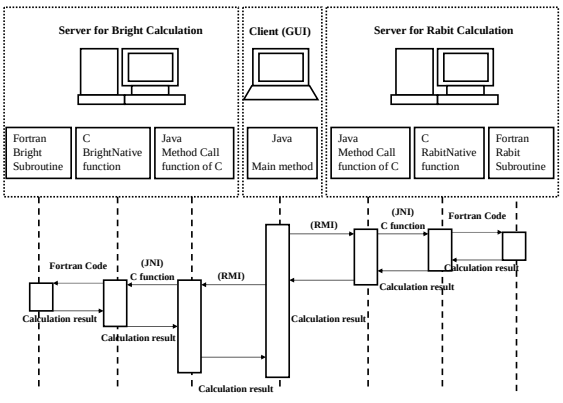

Fig.7.The process flow of distributed environment by RMI and JNI

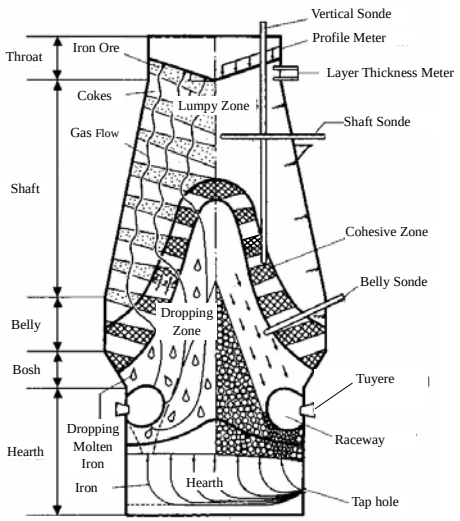

Fig.8. Outline of blast furnace process

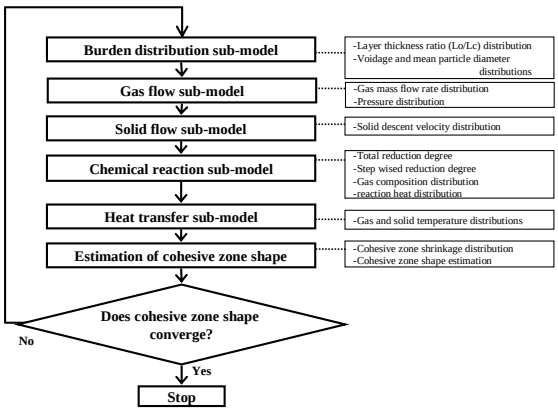

Fig.9. Configuration of two-dimensional mathematical model of blast furnace (BRIGHT model)

3. BLAST FURNACE PROCESS AND MODEL

3.1 Outline of Blast Furnace Process

A blast furnace is a moving-layer type counter-flow reaction vessel. Iron ore and coke, both in gains, are charged from its top one after the other so as to form a in through blast injection nozzles (tuyeres) at its lower portion and the hot air makes the coke burning to generate high-temperature reduction gas; then iron oxide in the iron ore is reduced and melted by the high-temperature reduction gas into molten pig iron. This reaction vessel is a vertical and cylindrical device of five sections, namely a throat, a shaft, a belly, bosh and a hearth, from top to bottom. The shaft and the bosh are of a truncated cone shape. (See Fig.8)

3.2 Configuration of the Blast Furnace Model

The two-dimensional mathematical model of the blast furnace (BRIGHT model) consists of the following sub-models: the burden distribution model, gas flow model, solid flow model, chemical reaction model, heat transfer model and cohesive zone shape judgment model. The sub-models are described in (Sugiyama, 1983; Sugiyama and Sugata, 1987a; Sugiyama and Sugata, 1987b). Composition of two-dimensional mathematical model of the blast furnace (NBRIGHT model) is introduced in (Naito and Nishimura, 2000).

Given a set of operating conditions, the procedure illustrated in Fig.9 is repeated to predict the shape of the cohesive zone and know the converged values of reduction degree distribution and temperature distribution. The calculation results are decided to be converged when the average value of the change in the shape of the cohesive zone fell within 0.5m.

4. BLAST FURNACE INTEGRATING SIMULATOR

4.1. Application to Blast Furnace Sub-Model Rabit

Rabit is a sub-model calculating burden distribution of a blast furnace. When Jni4FCB is applied to the Rabit model, common block used in Rabit is declared in Fortran program, Grdtrn.for. In the Grdtrn.for, there is 26 Common Blocks, and 155 variables are declared. Consequently, RabitFCB.java, RabitSkerton.java, RabitFCB.c, and RabitFCB.h are generated automatically from Grdtrn.for. A class diagram and sequence diagram about a class generated by Jni4FCB are shown in Fig.10.

4.2 Application to Blast Furnace Sub-Model Bright

Bright, is a sub-model calculating the reaction, the solid, the gas, the liquid flow and the heat transfer inside the furnace. When Jni4FCB is applied to the Bright model, common block used in Bright is declared in Fortran program, SBRIGHT.for. Consequently, BrightFCB.java, BrightSkelton.java, BrightFCB.c and BrightFCB.h are generated automatically. A sequence diagram about a class generated by Jni4FCB is shown in Fig.11.

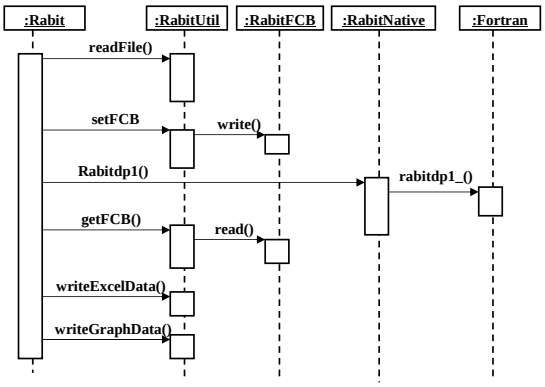

Fig.10. Generated sequence diagram (Rabit)

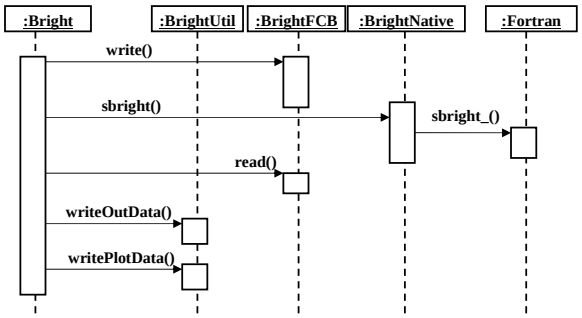

Fig.11. Generated sequence diagram (Bright)

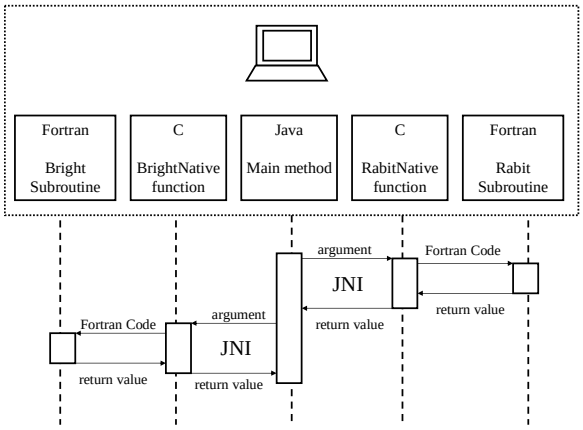

Fig.12. The process flow of the integrating simulator

4.3 The process flow of the integrating simulator

The process flow of the simulator integrated for Rabit and Bright by using JNI is shown in Fig.12. Thus, developing the simulator by the proposed method can save considerable labour and decrease time consumption more than redeveloping. Moreover, the value calculated by Rabit in memory without using file can be sent to Bright.

4.4 Online observation tool for simulation data

An online observation tool to observe data during the execution of simulation confirms the progress of the simulation. The tool is developed by integrating the sub routine which obtains the data of a Fortran program in the procedure of calculation. Moreover, the code which obtains the data by using the multithread of Java is described independently apart from calculation code. Therefore, it has the feature of obtaining Fortran data in parallel from the Java code during the calculation process of Fortran.

5. BLAST FURNACE INTEGRATING VISUALIZATION SYSTEM

It is difficult that the operator comprehend a large amount of data obtained from the blast furnace immediately. Thus, visualization technology to provide comprehensibly the large amount of data is demanded. Therefore, the blast furnace integrating visualization system to provide comprehensibly information for operator is described.

5.1 Configuration of the Blast Furnace Integrating Visualization System

The visualization system consists of the database system 1), the visualizing software 2) and the integrating simulator.

1) The database system stores the calculation data and the measurement data for a long period. The calculation data is got by the blast furnace integrating simulator. The measurement data is collected by process computer.

2) The visualizing software creates the visual information on both the calculation data and the measurement data. The data is converted to visual information on personal computer. The software can compare measurement data with calculation data of the time in Fig.13.

The database system, the process computer and the personal computer are connected to network. The system analyzes a data by on-line. In addition, offline analysis is possible by the data from the database system extracted.

5.2 Visualization of both Simulation Results and Measurement Data

The visualization system provides visual information on both the calculation data and the measurement data. The system shows graphics to compare the measurement data with the calculation data. Moreover, analysis time series data by animation. (See Fig.14)

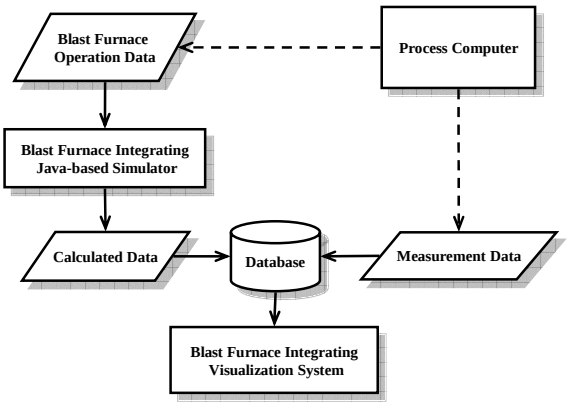

Fig.13. Outline of visualization system configuration

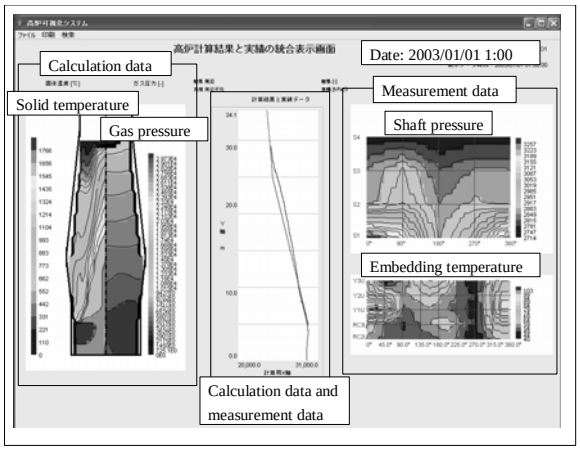

Fig.14. Image panel of visualization system

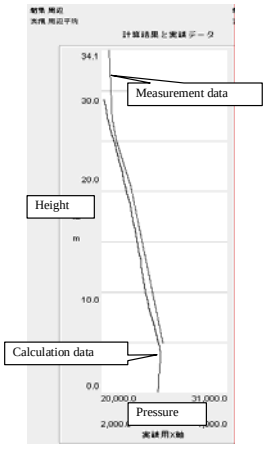

Fig.15. The example of comparison result of both the simulation results and real measurement data (2003/1/11/1:00)

The left-hand side graphics in Fig.14 provides visual information on the calculated data by the form of the blast furnace. In addition, the graphics shows the calculation data on 2-dimensional planes of both height direction and circumference direction of a blast furnace. For example, the visual information in Fig.14 shows solid temperature and gas pressure, by different colours and isograms.

The two right-hand side graphics in Fig.14 provides the spatial distribution characteristic of the measurement data. In addition, the graphics shows the measurement data on 2-dimensional planes of both height direction and the circumference direction of a blast furnace, also measurement data of each measurement sensor is arranged on 2-dimensional plane by corresponding with 3-dimensonal position information on the sensor exactly.

The central graphics in Fig.14 shows comparison of both the calculation data with the measurement data in certain time. Y-axis represents height position of blast furnace. X-axis represents temperature or pressure in this time.

5.3 Visualization Results of Using both Simulation Results and Measurement Data

The distribution of gas pressure of the calculation data is compared with that of shaft pressure of the measurement data by using the central graphics in Fig.14. The comparison is done by updating every 24 hours of one-month data. The example of comparison result of both the simulation results and real measurement data is shown in Fig.15. The results of comparison are summarized. Two distributions are mostly in good agreement in almost all days. Consequently, the visualization system can predict and evaluate the future actual distribution from the calculation data. Furthermore, if there is a situation that the two distributions are not in agreement, it can evaluate ideal operation was not performed.

6. CONCLUSION

In this paper, the useful method of creating a large-scale total simulator by integrating the sub-models made in the past has been proposed. The java-based integrating simulator for blast furnace has been developed for the blast furnace by using JNI and RMI. RMI has been used to integrate the two or more partial simulators in different computer environment. In addition, JNI has been used to integrate those simulators programmed in different programming languages. Developing the simulator by the proposed method was able to save considerable labour and time more than redeveloping. The comparison result of both the simulation results and real measurement data by the visualization system has been reported. the visualization system can predict and evaluate the future actual distribution from the calculation data. Thus, the blast furnace operation by using both the java-based integrating simulator for blast furnace and the visualization system can be supported. In other word, their systems can support control of the blast furnace.

REFERRANCES

Sugiyama, T. (1983). *Nisiyama Memorial Technically Lecture Series*, ISIJ**, 94**, 137-173

Sugiyama, T. and Sugata, M. (1987a). *Nippon Steel Technical Report*, **35**, 34-42.

Sugiyama, T. and Sugata, M. (1987b). *Seitetsu Kenkyu.* **325**, 34

Naito, M. and Nishimura, T. (2000). *Asia Steel International Conference 2000. Vol.B* (Iron making), pp.268-276

Nakayama, S. (2000). *Java Distributed Object initiation*, GIHODO, Inc

Cay, S.H. and Gary, C. (2002). *Core Java2* Vol.2, Chap.11 ASCII, Inc

Gordon, R. (1998). *JNI Java Native Interface Programming*, Pearson Education, Inc