# Legacy Migration of Mission-critical Systems of Headquarters

ICHIKAWA Kimiyoshi[*1] TAGAMI Takahiro[*2]

***Abstract:***

*In terms of host computer, it is said that there is a risk that sustainable operation and maintenance will not be possible in the future because of successive withdrawal of host computer makers and decreasing trend of host computer engineers. In addition, host computer has low affinity with advanced ICT technology. JFE Steel decided to migrate all the systems of headquarters from host computer environment to open environment and accomplished the project in 2021. Also, it means that digital transformation infrastructure in JFE Steel has been completed.*

## 1. Introduction

The first precondition for overcoming the “2025 digital cliff,” which has been a subject of warnings by Japan’s Ministry of Economy, Trade and Industry (METI), is evolution of legacy systems to an open architecture. Because mission-critical headquarters systems are high availability systems with more than 10 000 system users, and requests for system improvement are received on a daily basis, projects to migrate legacy systems to an open environment must be completed in a short time to reduce the duration of the freeze for system improvement. The following introduces the innovations adopted in the legacy migration project at JFE Steel.

## 2. Background of Development

The aims of this project were two-fold: To avoid risk and to enhance corporate value by completing infrastructure development to accelerate digital transformation (DX).

### 2.1 Shrinking Host Computer Market and Withdrawal of Host Computer Makers

Domestic sales of host computers were on the order of ¥1 trillion in the mid-1990s, but have remained on a level of less than ¥50 billion in recent years. As a result, Hitachi and Fujitsu, which are major computer makers, have announced one after another that they will withdraw from this business, facing a situation where continuing to operate systems with host (mainframe) computers, as in the past, invites substantial risk.

In addition, there are also many reasons for concern from the viewpoint of the supply of host system IT engineers, as the host system programming languages COBOL and JCL (JOB Control Language) are no longer offered as part of IT technology training.

### 2.2 Continuation of Structural Reform of IT Infrastructure as the Basis for Accelerating DX

Promotion of DX will play a role in the future growth of companies. However, because advanced ICT technologies are all based on open system technology, direct application to systems and databases running on host computers will be difficult. This means that structural reform of the IT infrastructure, that is, migration of legacy systems to an open environment, will be essential for efficient promotion of DX.

## 3. Overview and Features of Mission-critical Headquarters Systems

### 3.1 Overview of Mission-critical Headquarters Systems

The scale of the migration target system assets in this project is shown in **Table 1**.

[†] Originally published in *JFE GIHO* No. 51 (Feb. 2023), p. 11−16

[*1] Staff Manager, IT Innovation Leading Dept., JFE Steel

[*2] Group Manager of Sales System Group, Sales, Production & Logistics Systems Planning & Development Dept., Tokyo Office, JFE Systems

Table 1 Migration target assets

| Migration taget assets | | |
|---|---|---|
| Java | Number of steps | 17 million |
| COBOL | Number of steps | 24 million |
| Data Base | Number of databases | 15 824 |
| | Data volume | 5 TB |
| Files | Number of files | 57 511 |
| Report | Number of form types | 7 642 |
| Interface | Number of interfaces | 7 326 |
| CPU | MIPS | 15 000 |

Mission-critical headquarters systems consist of business management systems such as accounting, human resources and purchasing, and the sales, production and logistics systems used in those business activities.

In particular, the sales, production and logistics systems are high-load systems with a large volume of data.

### 3.2 Features of Sales, Production and Logistics Systems

The features of the sales, production and logistics systems are as follows.

(1) One feature of systems in the steel industry is the extremely large number of items and volume of data possessed by the systems. In the steel industry, products are frequently produced to meet the specific requirements of customers. Therefore, in addition to general contractual items, manufacturing instructions in which different requirement specification items are set for each type of product are sent to the steel works. Further, after product assortment, data for each actual product are stored for convenience in handling of individual product units.

(2) Systems conform to the business model of each product type, resulting in a high-load system with an enormous number of program assets.

(3) A closely-coupled condition exists between systems because DOA (Data Oriented Architecture) centering on a database is adopted.

(4) In addition to requests for modifications from customers, these systems must also be modified frequently to comply with revisions of the legal code and trade. This means that a prolonged freeze for system modification while implementing this project will have a severe negative impact on business.

Thus, the most difficult challenge in carrying out this project was to achieve the migration of the sales, production and logistics systems, which has the above-mentioned features, without impacting the company's business. To achieve this, it was necessary to minimize the system downtime for the migration work by shortening not only the work period for migration of the system, but also the time required for production migration.

## 4. Legacy Migration Method

In legacy migration, it is essential to address the incompatibility that occurs when migrating from a host environment to an open environment.

### 4.1 Selection of Migration Method

#### 4.1.1 Java and other open programming languages

Assets created in an open source programming language such as Java cause little incompatibility when transplanted to an open infrastructure. Therefore, in principle, simple migration was used for open system assets.

#### 4.1.2 COBOL, JCL and other host programming languages

The following two migration techniques are available for host programs.

(1) Rehosting

This is a method in which existing system assets are migrated to an open environment, in principle without modification, after starting compatible software on an open server. While this technique has the advantage that migration can be completed in a short time, it was not adopted in this project because the knowledge of host engineers will also be necessary in the future, and it does not solve the problem of depletion of host IT engineers.

(2) Rewriting

This is a technique in which the host programming language is converted to an open programming language. The rate of occurrence of incompatibilities is also high, which increases the difficulty of this approach, but it was adopted for this project, as it is the only technique that can achieve the intended purposes of legacy migration.

#### 4.1.3 Data character code

EBCDIC (Extended Binary Coded Decimal Interchange Code), which is a character code that is widely used in host computers, was converted to the open system character code S-JIS (Shift JIS) using a conversion tool.

**Table 2** shows the influence of character code type conversion from EBCDIC to S-JIS. The main effects were as follows.

(1) Change of code value

Table 2 Influence of character code type conversion

| Character code type | EBCDIC | Shift-JIS | Possible failure |
|---|---|---|---|
| Change of code value (ex."A") | 0XC1 | 0X41 | Garbled characters |
| Change of sort order | Lowercase letter<br>↓<br>Uppercase letter<br>↓<br>Numeric character | Lowercase letter<br>↑<br>Uppercase letter<br>↑<br>Numeric character | Output error of following JOB |
| Double-byte character contorol code | Exist | Not exist | Print misalignment |

Incorrect conversion may cause garbled screens or forms.

(2) Change of sort order

The sort order is different in EBCDIC and S-JIS. This means the size relationship of the character values also differs. When the output result of a previous JOB is used as the input for a following JOB, this can cause an error in which the output result of the following JOB is different from the current condition.

(3) Existence of double-byte character control

Although EBCDIC has a control code which shows the starting point and end point of double-byte characters before and after Japanese characters (*Kanji*) and other double-byte characters, this control code does not exist in S-JIS.

Therefore, when EBCDIC is converted to S-JIS, the lack of control code elements before and after double-byte characters may cause misalignment of the print position when printing forms, etc.

Since it is necessary to guarantee the current record length in order to prevent this, a common part which adds half-size (single-byte) spaces for 2 bytes after form data was developed in this project.

## 4.2 Migration Work Process by Rewriting

**Figure 1** shows process of legacy migration by rewriting. The content of each step is as follows.

① **Analysis of migration targets**

An inventory of system assets is conducted, the systems to be migrated are determined, and the degree of difficulty is analyzed by migration target.

② **Conversion design**

The conversion method and conversion specifications are designed for each migration target and technical element to be migrated.

③ **Pilot verification of conversion tool**

An automatic conversion tool is prepared based on the above-mentioned conversion design, and the possibility of performing conversion as intended is verified.

④ **Conversion**

The automatic programming language of large-volume migration targets is converted using the automatic conversion tool.

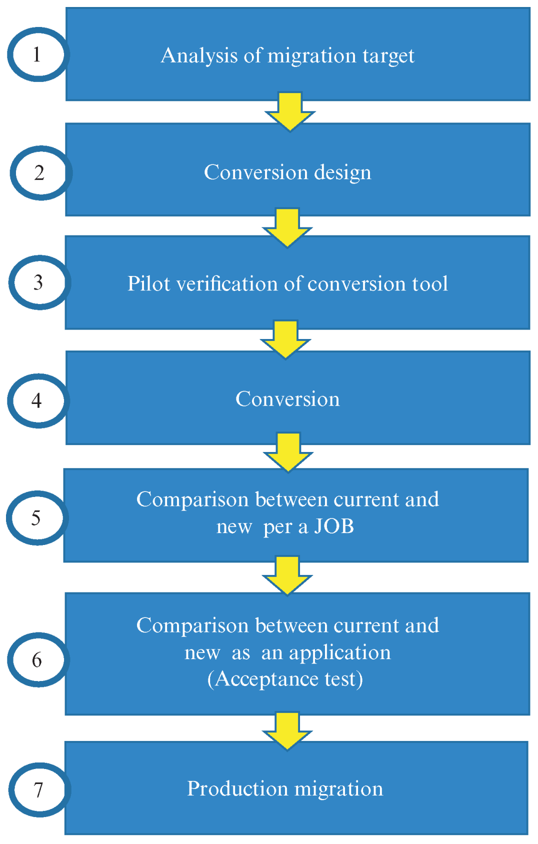
1
Analysis of migration target
2
Conversion design
3
Pilot verification of conversion tool
4
Conversion
5
Comparison between current and new per a JOB
6
Comparison between current and new as an application (Acceptance test)
7
Production migration

Fig. 1 Process of legacy migration

⑤ **Comparison between current and new**

Agreement between the output results before and after conversion is confirmed per a JOB.

⑥ **Acceptance test**

The JOBs are combined, and agreement of the output results before and after conversion in operation as a total system is confirmed. Non-function verification is also conducted, such as response.

⑦ **Production migration**

Character code conversion of production data is performed, the data are migrated, the converted programs are deployed in the open environment, and system operation is started.

The most time-consuming steps in the process outlined above are ⑤ Comparison of current and new and

⑥ Acceptance test. This is because incompatibilities frequently occur due to conversion of the programming language and character codes, requiring a repeated process of Test → Fix bugs → Test.

As mentioned above, since an extended system development freeze of mission-critical headquarters systems would have a substantial impact on business, the key point was how to complete this process in the minimum possible time.

## 5. Measures for Achieving Short Work Period

Various measures for completion of legacy migration in a short work period were devised and implemented. Among those measures, the following three introduces that were particularly effective in shortening the work periods for the above-mentioned ⑤ Comparison test of current and new and ⑥ Acceptance test, as well as innovations related to the migration of the enormous production database groups, which contributed to minimizing the system downtime during production migration work.

### 5.1 Shortening Work Period by Building Test Automation Tool

In conventional system tests, the entire process from preparation of the test data by the SE in charge of the test to verification of the correctness of the output results when those data are used as input data is performed by manual work.

Since it would be difficult to complete the test in a short time by this conventional test method, the authors constructed a tool which performs the tests automatically.

**Figure 2** shows a conceptual diagram of this test automation tool.

The test automation tool is a tool that originates from the test control database (DB), and manages the information necessary in automatic test execution in the test control DB. This information includes the JOB names of test objects, the test run order, JOB input data and output data names, and the information required in the comparison between current and new (compare information), etc.

The functions of the test automation tool are as follows.

(1) Control of the information necessary in test execution in the test control DB, including the JOB names of the test objects, the test run order, the JOB input data and output data names, and the information required in the comparison between current and new (compare information), etc. The data necessary in tests are extracted from the production environment according to the input data names of JOBs in the test control DB and transferred to the host development environment (automatic acquisition of test data).

(2) The process of Input data setup → JOB execution → Control execution result data is performed repeatedly according to the test run order of the test control DB.

(3) The test data and execution result data used in (2) are transferred to the open development environment.

(4) When the data are imported to the open development environment, they are converted from the host system character code (EBCDIC) to the open system character code (S-JIS).

(5) The JOBs are executed in the same run order as in (2), and the execution result data are stored.

(6) Using the compare information of the test control DB, the execution result data in (2) and (5) are compared from the viewpoint of 12 patterns, and the comparison results and differential data are output (see **Table 3**).

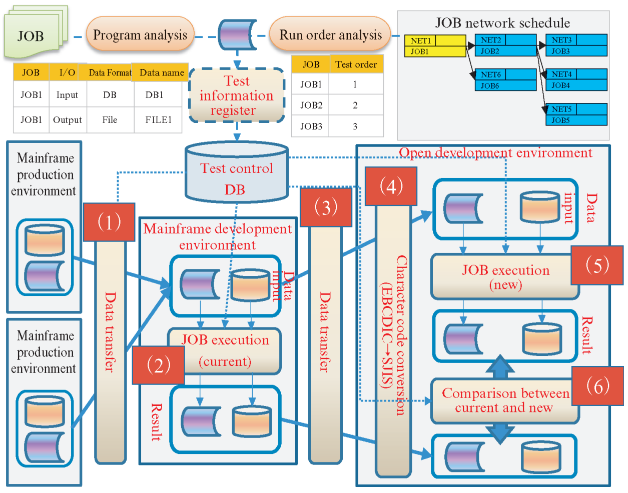

Fig. 2 Conceptual diagram of test automation tool

The application of this test automation tool is not limited to the comparison between current and new in a JOB units, but was also expanded to the acceptance test in order to confirm the correctness of the output results of the system as a whole. As a result, a JOB network linking multiple JOBs was recorded in the test control DB, and execution of a similar test for the system as a whole was realized.

Through these efforts, a large reduction in test man-hours was successfully achieved.

### 5.2 Invention of Efficient Testing Method

Based on the constraint that the project must be completed in a short work period, a test method for efficiently securing a certain level of system quality was conceived and implemented. The concept of the estab-

Table 3 Current-new comparison tests for 12 patterns

| | | Handling of half-width space | | |
|---|---|---|---|---|
| | | Do nothing | Trim half-width space | Trim both half-width and double-byte space |
| Pretreatment of current-new comparison tests | As it is | Pattern1 | Pattern5 | Pattern9 |
| | Mask some items | Pattern2 | Pattern6 | Pattern10 |
| | Sort in order of S-JIS | Pattern3 | Pattern7 | Pattern11 |
| | Sort in order of S-JIS and mask some items | Pattern4 | Pattern8 | Pattern12 |

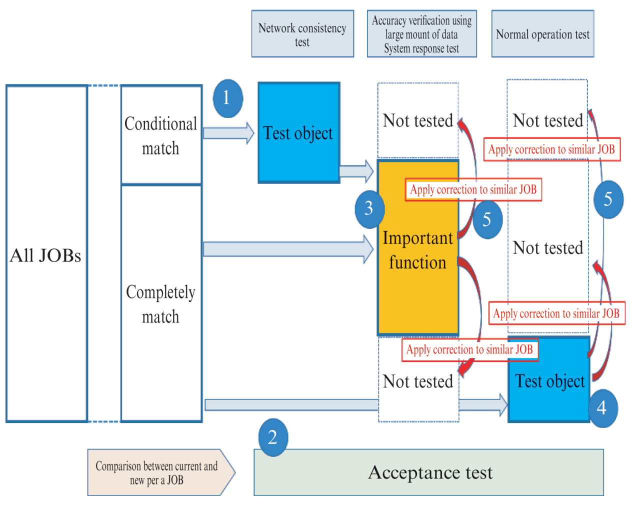
Network consistency test
Accuracy verification using large mount of data System response test
Normal operation test
All JOBs
Conditional match
Completely match
1
Test object
Not tested
Not tested
Apply correction to similar JOB
Apply correction to similar JOB
3
5
5
Important function
Not tested
Apply correction to similar JOB
Apply correction to similar JOB
Not tested
Test object
4
2
Comparison between current and new per a JOB
Acceptance test

Fig. 3 Conceptual diagram of efficient testing

lished test plan is as shown below.

(1) Testing of all system functions would invite an excessively long work period, and was substantially impossible in light of the features of the mission-critical headquarters systems.
(2) Every effort should be made to eliminate duplication of the test. Important functions from the viewpoint of their effects on business should be selected, and those functions should be tested carefully.
(3) Because latent defects which are not included in important functions are minor in terms of their effects on business, they should be responded separately by prompt action when discovered after the start of production.

**Figure 3** shows the concept of the efficient testing method.

(1) All comparison of current-new tests per a JOB are executed. For JOBs in which mismatch phenomena such as differences in the sort order due to character code conversion, etc. were discovered in this process, JOB network consistency tests are performed as “Conditional matches” (matches with conditions attached), and pass/fail is judged based on the results. This test also serves as an acceptance test.
(2) After all comparison of current-new tests per a JOB pass, a test is conducted to determine whether execution incompatibility occurs when the entire system is started.
(3) For important functions, accuracy verification of the output results is conducted using a large amount of data in order to improve system quality.
(4) For non-important functions, only normal operation is verified (normal operation test).
(5) For nonconforming items detected by each test, the quality of functions other than the test object is also improved by horizontal development, in which the same corrections are applied to similar programs with functions that were not targets of the test concerned.

### 5.3 Reduction of Number of Tests by Patterning of Migration Targets

In addition to the test methodology described above, measures were also taken to reduce the number of tests based on the distinctive features of legacy migration. The way of thinking about these measures is as follows.

(1) Unlike normal system development, the business logic itself is not changed by conversion.
(2) Based on the conversion design prepared in advance, conversion of the programming language and character codes was performed using a vendor’s automatic conversion tool. Therefore, the occurrence of incompatibility was similar for each pattern.
(3) Tests were not performed targeting all JOBs. Rather, patterning of the migration targets was carried out in advance, and tests were performed for only a small number of JOBs representative of those patterns.
(4) When the result of the test of the representative small number of JOBs was pass, other JOBs which belong to that pattern were considered to pass. Incompatibilities that occurred in a representative case were regarded as also occurring in other system assets belonging to the same pattern, and the same system improvements were carried out immediately.

**Table 4** shows the number of tests of reports and interfaces in normal system development and the number of tests conducted in this project.

The work period was shortened by reducing the number of tests to less than half of that in normal sys-

Table 4 Results of reducing number of tests by patterning

| | Number of test run | |
|---|---|---|
| | Normal system development | Concerned project |
| Report | 7 642 | 2 540 |
| Interface | 7 326 | 4 204 |

tem development by this innovative patterning technique.

### 5.4 Innovation in Production Migration of Databases

The greatest challenge in production migration was how to complete the production migration work in a short time while maintaining consistency between databases (DBs) when migrating more than 15 000 DBs.

Production migration of the DBs by the conventional method would require nearly 1 week, and a complete downtime of migration target system is necessary during that period. However, due to the features of the mission-critical headquarters systems, a full 1-week shutdown is not possible, even during the company's year-end/new-years holidays.

As a technique for solving this problem, the following production migration method was adopted (see **Fig. 4**).

(1) The DBs to be migrated on the day of production migration are limited to those with a low frequency of updating and small migration work load.
(2) For DBs with a high frequency of updating and a large DB migration work load, a replica DB of the production DB on the host (mainframe environment) is created in the open environment of the migration destination from 2 weeks before the production migration work date.
(3) In case the DB in the host production environment is updated during the period up to the production migration work date, the replica DB in the open production environment is also updated for the same period (synchronous updating).
(4) As a result of this procedure, the content of the two DBs is completely identical at the point in time just before the system stop for the start of production migration work.

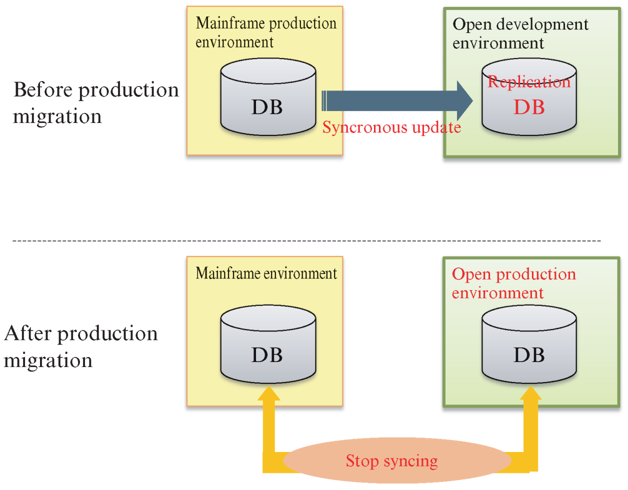
Before production migration
Mainframe production environment
DB
Syncronous update
Open development environment
Replication
DB
After production migration
Mainframe environment
DB
Open production environment
DB
Stop syncing

Fig. 4 Production migration of data bases

(5) After the production migration work, the replica DB is set as true, and synchronization with the DB on the host is aborted.

By applying this technique, we succeeded in limiting the production migration work and accompanying system downtime to approximately 52 hours.

The various measures for shortening the work period described above were successful, and legacy migration of the large-scale system could be completed in about 3 years.

## 6. Conclusion

As a result of the efforts described in this paper, complete migration of mission-critical headquarters systems to an open environment was achieved for the first time in the steel industry, and system-related risks to business continuity were eliminated.

An additional aim of this project was infrastructure improvement to accelerate DX. This was also realized as a result of the project. However, in reality, whether it is effective in the true sense will depend entirely on the promotion of DX projects in the future.

The authors intend to work to further improve corporate value by deepening our collaboration with the business division and thinking hard in order to actively promote DX projects.