Approach to the Development of Continuous Carrier Systems

WATANABE Tetsu, NITTA Tetsurou, HARUI Daisuke, YAGINUMA Osamu, YOKOTA Nobuyuki, HIDAKA Yoshinari

Abstract

Systems development on cloud platforms (PaaS, IaaS, etc.) has recently become common practice among the telecom carriers and the targeting of highly important mission-critical systems is now expanding. Systems integration (SI) in cloud platforms has certain advantages, such as flexibility in resource procurement. However, it also poses issues specific to the cloud platform, such as fault complications and resource conflicts. This paper introduces the issues posed in the process of systems development on cloud platforms and the SI techniques proposed as a means of solving these issues. The point is that the ultimate purpose of the process is to achieve optimum quality and that it should be approached based on the integrated collaboration of the application (App) development teams and the platform (PF) teams.

Keywords

cloud platform, OMCS, high availability, statistical technique, quality, private cloud, PaaS, IaaS, multi-tenancy

1. Introduction

One of the interesting recent changes in system development techniques is the dissemination of developments utilizing the PF services of cloud platforms (PaaS, IaaS, etc.). Meanwhile, the carrier systems require "uninterrupted" platforms and "flexible" development so that they can deploy the carrier services effectively. The ability of the PaaS/IaaS platforms to secure these features and "support the App development" is an important issue of the efforts supporting development. In this paper, the authors introduce some examples of the approach to SI taken in the development of continuous carrier systems, as seen from the viewpoint of the PF builders.

2. Characteristics of Systems Development on Cloud Platforms, Issues Involved

There are roughly speaking two important characteristics of systems development on cloud platforms. One is the "PF shared type" that several systems share a common platform, and the second is the "PF/App separation type" that the PF building vendors and the App development vendors work separately to achieve a common project. This means that an approach is required that is different from that of the traditional so-called "silo type" SI development, in which the platform is developed per individual system.

2.1 Issue 1: Considerations for Shared Type Usage

With regard to the issue of PF sharing by several systems, it is the progress of virtualization technology, etc., that enables the sharing of a single hardware (HW) resource by several systems. On the other hand, a more complex PF design is required than previously due to the sharing of resources including storages and networks as well as to the need for overall flow control of the multi-tenancy operation.

2.2 Issue 2: Disadvantages of PF/ App Separation

For the issue of PF and App vendor separation, a trend is emerging in the cloud environment, in which the PF is separated into layers and each layer is implemented as a service provided by individual vendors (PaaS/IaaS).

This trend has resulted in an increase in the number of system development modes that are separating the PF building vendors and the App development vendors.

As a result, if an App developer designs a system without having sufficient knowledge of the PF specifications and bottleneck points (loose couplings), occurrences of performance problems and unexpected service anomalies due to difficulties in identifying the effects of faults will tend to increase. If this should happen, it would be impossible to build an "uninterrupted system" however hard the PF builder tries to implement a high availability design.

3. Approach toward Issues

3.1 Arrangement of Design Process, White Box Implementation

For the use of shared PFs as in issue 1, we define a high availability design process in the following three steps in order to secure the quality that also covers the non-functional design (**Fig. 1**).

(1) Clear SLA definition, grand design development

In this case, step 1 consists of a clear definition of "requirements" in the SLA and grand design development (common design of the PF) in order to ensure the understanding of the PF design. The grand design should prescribe the design elements to be secured including the non-functional requirements of importance for the infrastructure design. As a result, even if a new model or new product (middleware: MW) is added later, returning to the grand design allows the process to benefit from the original design policy.

(2) White box design

Step 2 is to implement each product as a white box (**Fig. 2**). As for the HW design, the building target components are separated into "fault-potential positions" and the effects on the operations and services in the case of a fault are visualized. When designing the OS and MW, the specifications of operations in the case of system change are clarified for all of the parameters that become the design elements.

The point in this activity is that the designer has optimum understanding of the design parameters and their potential behaviors. This enables a consistent design even when HW, OS and MW with different processing properties are combined. It also leads to ensuring the business continuity under any fault pattern (which is the "Input" procedure in step 3) based on combinations of operations assumed on the desktop.

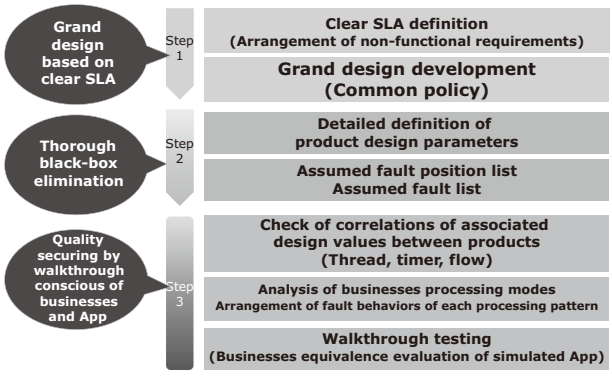

Fig. 1 Process of high-availability design.

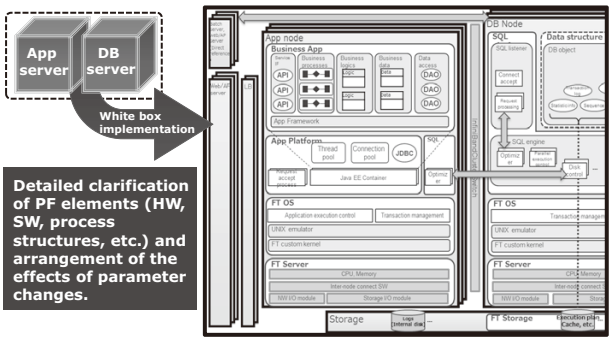

Fig. 2 White box implementation.

(3) Walkthrough

Finally, step 3 is the walkthrough. This term is borrowed from the theatrical art meaning the rehearsal, in which all of the scenes are played to test the flow of the plot and movements of actors. Modeled in the PF building, this operation consists of deskbound verification of the "processing steps" from the client's request to the reply. Or, that the design meets the system requirements from the viewpoint of availability, by compiling them into processing patterns compiled from the viewpoint of the infrastructures and taking the system faults assumed in step 2 into consideration (**Fig. 3**). This step helps eliminate the theoretical traps. It is also important to check and evaluate the walkthrough conducted as a desktop operation by running the actual machines using a simulated App in order to confirm the correctness of the assumptions of the operations in the case of a fault.

3.2 Deployment of Development Guideline, Establishment of Tenant Follow-up System

This section discusses the second issue of the separation between the PF and the App vendors.

The aim of the approach here is that the PF team plays

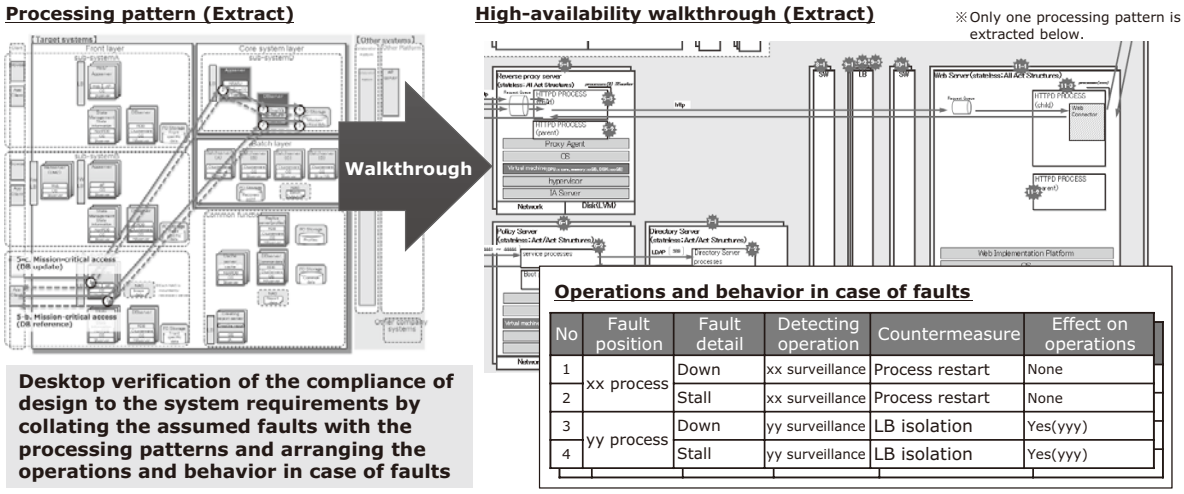

Fig. 3 Processing pattern arrangement and walkthrough.

the leading role in confirming; 1) overall architecture; 2) actual implementation and; 3) development process. Confirmation of these three points allows the PF building team to perform optimum follow-up at optimum timings.

In addition, the PF team compiles and provides the notes and cautions on the PF in question into guidelines (SQL development guidelines to be observed in the App development, Java development guidelines, etc.). This can ensure availability and minimize the performance risks when the App is in-service. The approach described above has already been put into practice in several projects. What is important is that it is an "App/PF joint development technique", which is a characteristic of the case described in this paper.

(1) Confirming the overall architecture

In the confirmation of the overall architecture, the PF and App vendors present and coordinate their proposals for the architecture in order to deepen mutual understanding. It is important that such an approach eliminates any deviations in the elements required for the system in question that exist between the PF and App vendors. This process is also important in eliminating missing points in guidelines, as well as making the system highly effective (**Fig. 4**).

(2) Confirming the actual procedure of the App development and implementation

In the confirmation of the actual procedure of the App development and implementation, the PF team should assess the points of App implementation optimally in order to inform the developer of the items to be defined in the development guidelines. Subsequently, visualizing the relationships between the items in the guidelines and the "properties of the project in question" makes it possible to check the adequacy level objectively and to improve the quality of the developed App as well as of the guidelines themselves (**Fig. 5**).

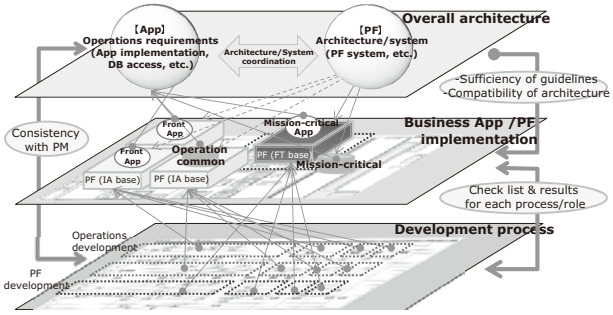

Fig. 4 Approach to ensuring the business App/PF implementations.

(3) Confirming the development process

Finally, in the confirmation of the development process, the PF team should compile and deploy the guidelines matching the App development and implementation with the App developer based on the results of coordination of the overall architecture. It is also required to apply the guidelines to the development "process" in order to confirm their observance. This requires a positive approach from the PF team. As shown in **Fig. 6**, it is important to explain the App development process and to also ensure the quality of the upstream promotion activity of the development process. As the top-down leadership of the client is an important factor in these

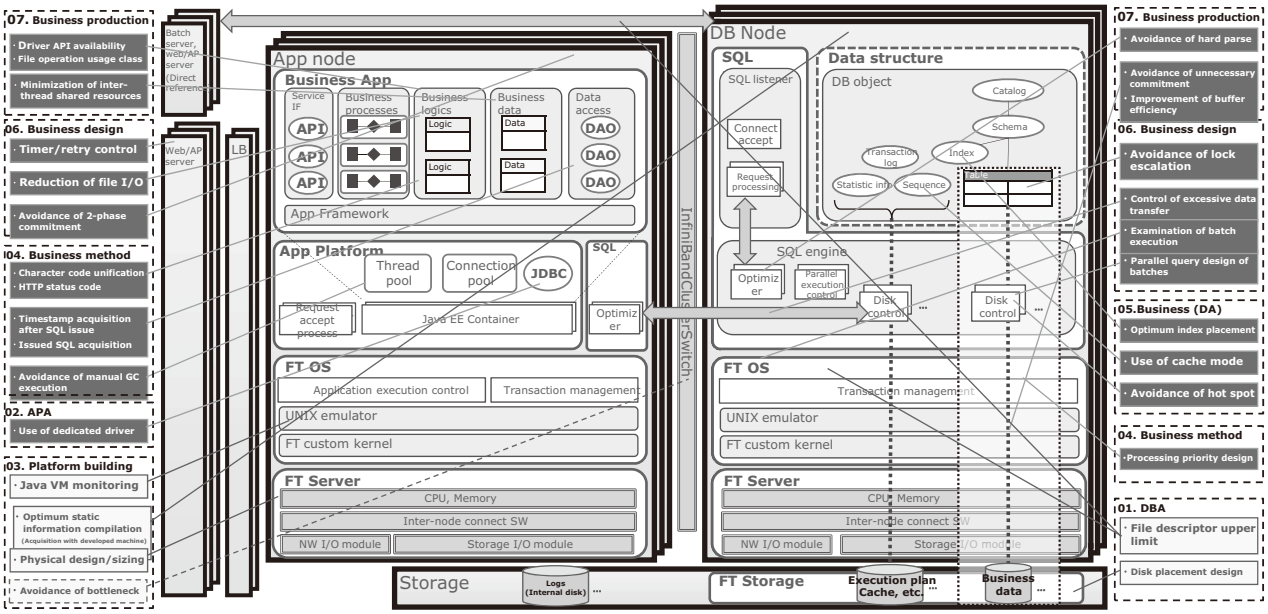

Fig. 5 Guidelines and visualization of the App /PF implementation.

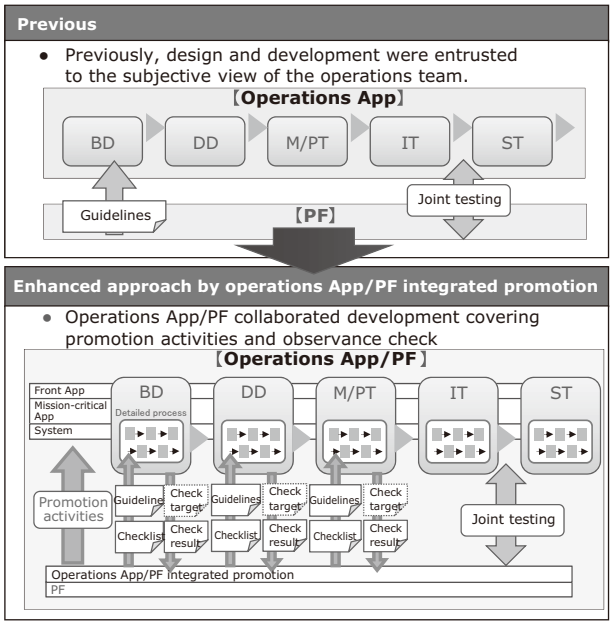

Fig. 6 Improving the development process.

activities, it is essential for the vendors to prepare an overall navigation system for following up the optimal information provision and other activities.

4. Future Systems Development

It is only since the big changes brought about by the trend of open systems in the 1990s that the system development technique has entered a new era of change, which is currently being brought about by the trend of cloud computing (provision of PF as a service). Such changes are likely to continue, and may occur at any time in the future because of changes in technology, market environment and/or customer needs.

We expect that user-oriented models such as omni-channel systems aiming at improving the UX (User Experience) at contact with customers will expand in the future telecom business. In order to support this new trend and provide common values for the users, the App and PF vendors will be required to enhance and integrated their collaboration in the field of system design.

As seen in the above, developmental techniques evolve by inheriting past techniques and applying them to the present and to the future. It is on the extension of the wisdom of foregoers that a new era comes. We at NEC, a company contributing to the market as a SIer, aim to continue our efforts by defining the signs of innovative changes accurately. This will be achieved by meeting challenges with agility and without neglecting the need for mindful renovation.

* Java is a registered trademark of Oracle Corporation and/or its affiliates in the U.S. and other countries.

Authors' Profiles

WATANABE Tetsu
Senior Manager
Common Carrier Solutions Division

NITTA Tetsurou
Manager
Common Carrier Solutions Division

HARUI Daisuke
Manager
Common Carrier Solutions Division

YAGINUMA Osamu
Assistant Manager
Emerging SI Technology Development Center

YOKOTA Nobuyuki
Assistant Manager
OMCS Division

HIDAKA Yoshinari
Assistant Manager
Common Carrier Solutions Division

Information about the NEC Technical Journal

Thank you for reading the paper.
If you are interested in the NEC Technical Journal, you can also read other papers on our website.

Link to NEC Technical Journal website

Japanese English

Vol.10 No.3 Special Issue on Telecom Carrier Solutions for New Value Creation

Vol.10 No.3
July 2016

Special Issue TOP

General Paper

NEC Information