"SystemDirector Enterprise" for Improving the Development Efficiency of Multitenancy Compatible Applications

KOBAYASHI Shigenori, KOIZUMI Ken

Abstract

In order to provide multiple service user enterprises (tenants) with applications as services it is necessary to reduce the service provision costs by the efficient use of available resources. For this purpose, every application must have a multitenancy, with which a single application can handle multiple tenants. This paper describes a technique of multi-tenant application development that makes use of the multitenancy development support function provided by SDE (SystemDirector Enterprise).

Keywords

SaaS, cloud computing, multitenancy, SystemDirector Enterprise

1. Introduction

The rapid development of IT technology including the web and virtualization technologies and the increased need for on-demand type applications by business users have promoted rapid developments in SaaS (Software as a Service) via internet.

To provide multiple service users (tenants) with applications via the SaaS model, it is necessary to reduce service provision costs by applying an efficient use of resources. For this purpose, a mechanism is required to make a single application capable of handling multiple tenants. An architecture in which a single system is shared by multiple tenants is referred to as multitenant architecture.

This paper describes the architecture and development techniques of the multitenant application framework offered by SDE (SystemDirector Enterprise).

2. Multitenancy Models

Multitenancy can be constructed either as a single-instance model or as a multi-instance model as shown in **Fig. 1** . Instantiation produces an "image" that allows tenant users to run data on their computers. However, the actual data of the individual tenant is accommodated in the instance.

Implementation of multitenancy necessitates the isolation of applications and data.

A pattern for isolating applications is to isolate them in the OS, middleware and application levels (**Fig. 2**). SDE provides both the framework and a development support tool for performing single process pattern isolation.

The patterns for isolating data include; the one for isolation at the instance level of the DBMS (Data Base Management System), the one for isolation at the logical division unit level, called the DBMS scheme and that of isolation using tabular columns (**Fig. 3**). SDE supports isolation based both on individual DBs and that based on shared DBs and individual schemes.

Fig. 4 shows an example of a system for which applications and data are constructed with multitenancy.

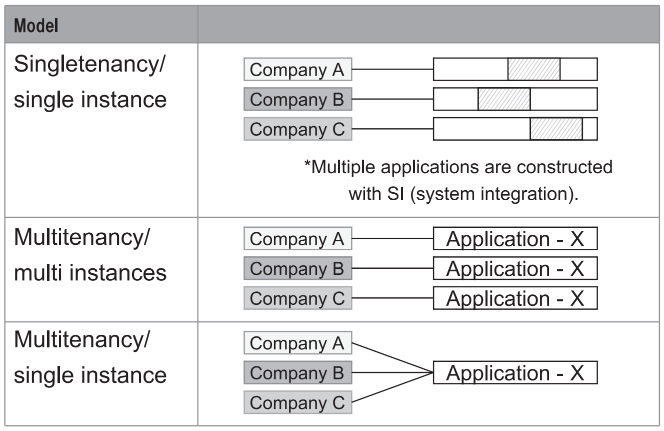

Model
Singletenancy/ single instance
Company A
Company B
Company C
*Multiple applications are constructed with SI (system integration).
Multitenancy/ multi instances
Company A
Company B
Company C
Application - X
Application - X
Application - X
Multitenancy/ single instance
Company A
Company B
Company C
Application - X

Fig. 1 Relationship between tenant and the instance.

This example employs single process type application isolation and shared DBs/individual scheme data isolation.

The patterns used in application and data isolation are selected with regard to trade-offs between the application service level and the resource utilization efficiency (degree of tenant intensity). From the perspective of the service provider what is important eventually improve the investment recovery rate by supporting as many tenants as possible while using as few resources as possible. Below, we discuss the technique for implementing a single process pattern using an SDE that is capable of providing services with few resources.

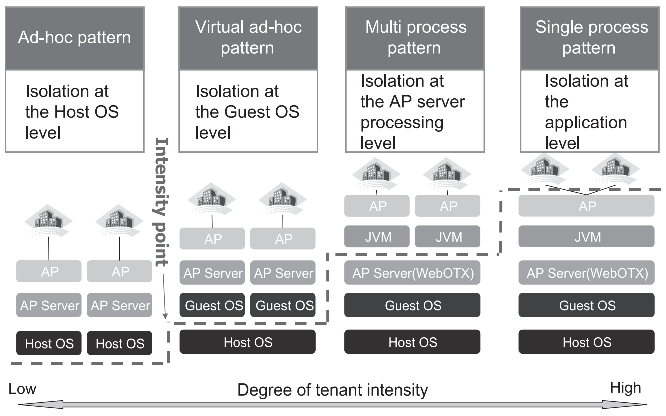

Ad-hoc pattern
Isolation at the Host OS level
Virtual ad-hoc pattern
Isolation at the Guest OS level
Multi process pattern
Isolation at the AP server processing level
Single process pattern
Isolation at the application level
Intensity point
AP
AP Server
Host OS
Guest OS
JVM
AP Server(WebOTX)
Low
Degree of tenant intensity
High

Fig. 2 Patterns of application isolation.

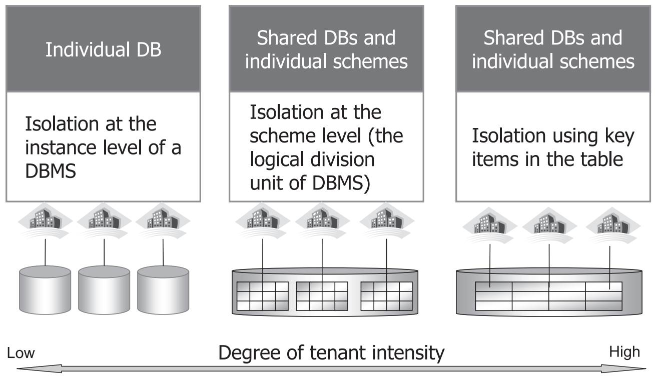

Individual DB
Isolation at the instance level of a DBMS
Shared DBs and individual schemes
Isolation at the scheme level (the logical division unit of DBMS)
Shared DBs and individual schemes
Isolation using key items in the table
Low
Degree of tenant intensity
High

Fig. 3 Patterns of data isolation.

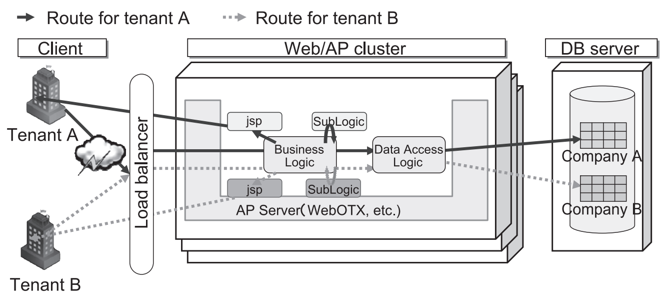


Fig. 4 Example of a multitenant application.

3. Issues in Multitenancy Development and in Functions Provided by SDE

In order to implement multitenancy of the single process pattern it is necessary to enable it from the design stage. The design should therefore be carried out by examining the following issues.

(1) Multitenant Application Development Process

A development process that allows separate control of tenant-common processing (a common processing for all tenants) and per-tenant processing (a specific processing for individual tenants).

(2) Customizable Architecture

Architecture for the development of multitenant applications;

SDE provides the following functions for supporting multitenant application development.

- Multitenancy development guide.
- Multitenancy development support generator.
- Multitenant framework.

The details are shown in **Fig. 5** .

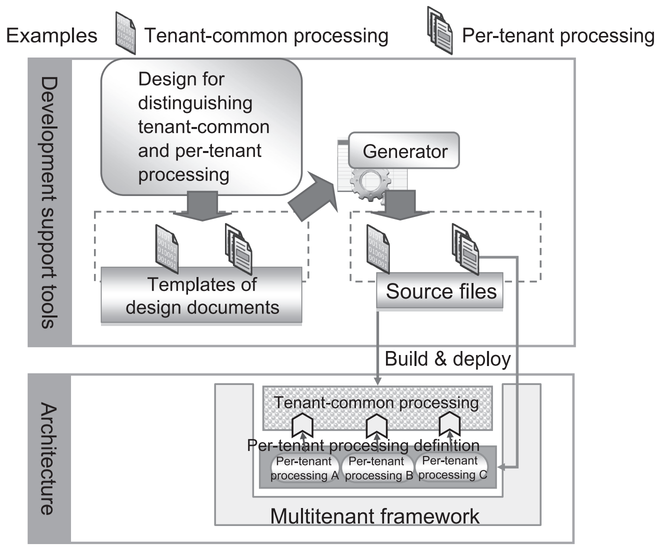


Fig. 5 Multitenant application development support function of SDE.

SDE provides design document templates for enabling designs that separately distinguish and manage tenant-common processing and per-tenant processing. It also provides a function for the automatic generation of source codes for the multitenant framework based on the information in the design document (source code generator). These development support tools and frameworks that are compliant with the multitenant application development process support the efficient development of multitenant applications.

3.1 Multitenant Application Development Process

With regard to the development of multitenant applications, the stage of separately developing tenant-common processing and per-tenant processing can be efficiently carried out by adopting the concept of "production line development."

With production line development a characteristic that can be varied within an application is referred to as "variability" and points that can be varied within "variability" as "variability points." The variation options are called "variants."

When this is viewed from the perspective of a multitenant application, the "variability" can be substituted by the "per-tenant processing." More specifically, the points that are changeable in per-tenant processing can be regarded as "variables" and the actual contents of per-tenant processing as "variants" (**Fig. 6**).

It is important to manage the "variability" described above and to document tenant-common processing and per-tenant processing in the development process and to continue to manage them throughout the development process.

3.2 Multitenancy Framework

The "production line development" recommends improvement of the application development/maintenance efficiencies by using an architecture that fixes variability in a standard form. Similarly, the development/maintenance efficiencies of multitenant applications can also be improved by applying a standard architecture capable of including per-tenant processing into tenant-common processing so that the "development of standard applications" and the "development of customized applications" may be isolated as will be described later.

SDE provides a multitenancy framework using the Spring Framework that makes use of the DI (Dependency Injection) feature of the DI container. The DI is a technique for solving the dependency between components by the DI container that acts as a third party solution instead of solving dependency by the components themselves.

Fig. 7 shows a diagram of a multitenant framework based on the Spring Framework.

The use of the DI container in the multitenant framework provides the implemented multitenant framework with the following features.

- Componentization of modules (classes).
- Elimination of dependency between components.
- Externalization of customized information.

SDE provides the mechanism for switching transactions per tenant (scenario controller function) by utilizing the functions of the DI container described above.

The scenario controller makes it possible to define the "tenant execution scenario" for each tenant. For the transactions

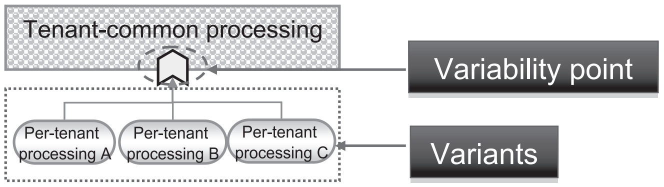
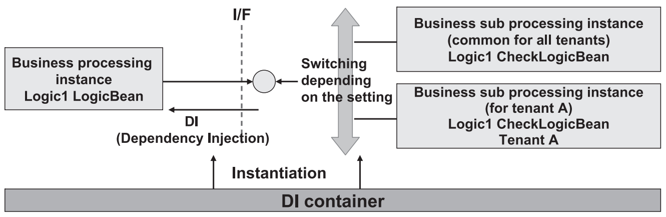


Fig. 6 Relationships between "variability" in production line development and per-tenant processing.



I/F
Business processing instance Logic1 LogicBean
Switching depending on the setting
Business sub processing instance (common for all tenants) Logic1 CheckLogicBean
Business sub processing instance (for tenant A) Logic1 CheckLogicBean Tenant A
DI (Dependency Injection)
Instantiation
DI container

Fig. 7 Customization using the Spring Framework.

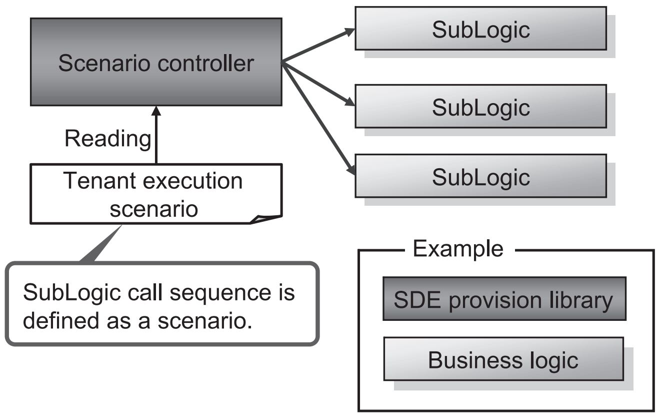


Fig. 8 Scenario controller.

Transaction	Execution scenario for the transaction
Tenant A (Standard scenario)	Standard initial processing → Standard processing 1 → Standard processing 2
Tenant B	Initial processing for tenant B → Standard processing 1 → Standard processing 2
Tenant C	Initial processing for tenant C → Standard processing 1 → Standard processing 2 (dummy) → Additional processing for tenant C

Fig. 9 Transaction switching per tenant.

that do not belong to the scenario used in common by all tenants but should be defined for individual tenants, processing may be implemented by replacing the business logic components (which are called SubLogics with SDE) (**Fig. 8**).

In the examples shown in **Fig. 9** , the standard transaction is executed as a tenant A transaction that does not include per-tenant processing. With tenants B and C that involve customized application per tenant, the transactions are executed by replacing tenant-common processing by per-tenant processing.

4. Development of Multitenant Applications Using SDE

The development of multitenant applications can be divided into the “development of the standard application” and the “development of customized applications.”

(1) Development of the Standard Application
This process develops the standard application to be provided as a service that is carried out by the SaaS provider.

(2) Development of Customized Applications
This process develops customized processing for each tenant based on the result of the standard application development. It is carried out by the SaaS provider and SSP (SaaS Solution Provider).

The division of the development process makes it possible to isolate two interests. These are the construction of a rugged standard application and the rapid development of customizations for specific tenants (**Fig. 10**).

4.1 Development of the Standard Application

This process develops the standard application by assuming the design and packaging of a multitenancy model. Also, this process is based on the development process defined in the

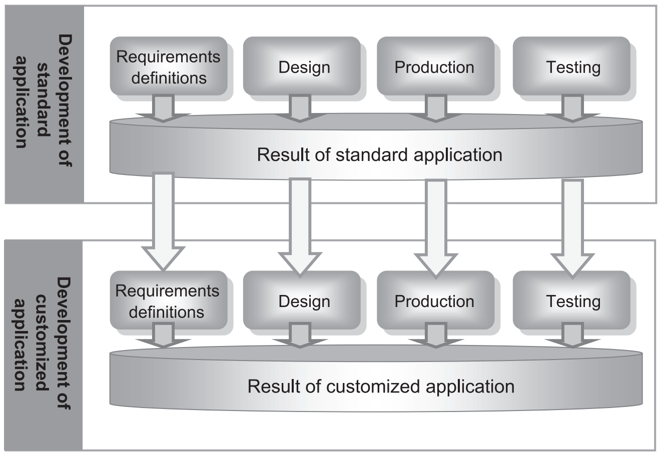

Development of standard application
Requirements definitions
Design
Production
Testing
Result of standard application
Development of customized application
Requirements definitions
Design
Production
Testing
Result of customized application

Fig. 10 Multitenant application development process.

development methodology. The difference from ordinary development lies in the multitenancy design, i.e., the “variability management.” This process includes definitions of the “variables (targets of variability) ” in the standard application according to the functional and non-functional requirements of the application.

In order to facilitate development of multitenant applications by developers skilled in business applications development, SDE now provides a multitenant application development guide. This does not alter the development process in the business applications development methodology that has been provided thus far.

4.2 Development of Customized Applications

This process develops customizations for specific tenants based on the standard application.

With SDE, the development of per-tenant processing is performed using a separate design document from that for tenant-common processing. The processing specifications are customized from the following perspectives.

(1) Addition/deletion of SubLogic calls.
(2) Modification of SubLogic calls sequences.
(3) Modification of SubLogic implementations.

5. Conclusion

In the above, we introduced the multitenant application development technique adopted by SDE. As SaaS business is expanding very rapidly at present, we at NEC are also adopting measures to meet this trend by endeavoring to prepare a common platform for SaaS applications. In the future, we aim to enhance SDE by incorporating the latest technologies such as the SaaS application development support function and by linkage with the SaaS platform service "RIACUBE/SP."

Authors' Profiles

KOBAYASHI Shigenori
Manager
Software Process Innovation and Standardization Division

KOIZUMI Ken
Assistant Manager
Software Process Innovation and Standardization Division

The details about this paper can be seen at the following.
Related URL:
SystemDirector Enterprise:
http://www.nec.co.jp/cced/SystemDirector/SystemDirectorEnterprise/index.html