行政院及所屬各機關出國報告
(出國類別：實習)

寬頻網路整合管理系統技術實習—Enterprise JavaBean Programming 暨 參加 TM Forum Conference 出國實習報告書

服務機關：中華電信研究所
出國人　職　　稱：助理研究員
　　　　姓　　名：許永義
出國地區：美國洛杉磯、拉斯維加斯
出國期間：91 年10 月20 日至91 年11 月1 日
報告日期：91 年 12 月 23 日

146/C09200567

系統識別號:C09200567

公 務 出 國 報 告 提 要

頁數: 11　含附件: 否

報告名稱:

寬頻網路整合管理系統技術實習—Enterprise JavaBean Programming暨參加 TM Forum Conference

主辦機關:

中華電信研究所

聯絡人／電話:

楊學文／03-4244218

出國人員:

許永義　中華電信研究所　91830專案研究計畫　助理研究員

出國類別: 實習

出國地區: 美國

出國期間: 民國 91 年 10 月 20 日 -民國 91 年 11 月 01 日

報告日期: 民國 91 年 12 月 23 日

分類號/目: H6／電信　／

關鍵詞:　寬頻網路,Enterprise,JavaBean,Programming,TM,Forum

內容摘要: 本計畫(專案830計畫)目前正協助開發ADSL/ATM寬頻網路管理系統，本系統採用三階式Web-based架構，其目的在於減輕使用者端程式維護所須耗費的人力及物力；並且所使用的程式語言技術採用可攜性高的Java程式語言，而SUN公司之Java技術居於市場領先地位。職奉准至美國SUN公司教育訓練機構參加「Enterprise JavaBean Programming」訓練課程，此訓練課程教導有關J2EE相容EJBs的建立，以及如何將它們結合成強韌的企業應用程式等必要的知識。研習內容包括：Session and Entity Beans、EJBs 的Container架構暨運作管理功能技術。期望藉由此項訓練，學習J2EE架構中各項相關的技術，以提升未來程式開發及系統維護所需的技能。另實習結束後繼續參加電信管理論壇(TeleManagement Forum)所舉辦之國際性研討會，本公司為該組織之會員，參與該研討會以了解下一代維運支援系統(NGOSS)的架構與趨勢，及世界各國發展的情形與經驗，有助於本公司規劃、設計、開發下一代的維運支援系統。

摘　　要

本計畫(專案 830 計畫)目前正協助開發 ADSL/ATM 寬頻網路管理系統，本系統採用三階式 Web-based 架構，其目的在於減輕使用者端程式維護所須耗費的人力及物力；並且所使用的程式語言技術採用可攜性高的 Java 程式語言，而 SUN 公司之 Java 技術居於市場領先地位。職奉准至美國 SUN 公司教育訓練機構參加「Enterprise JavaBean Programming」訓練課程，此訓練課程教導有關 J2EE 相容 EJBs 的建立，以及如何將它們結合成強韌的企業應用程式等必要的知識。研習內容包括：Session and Entity Beans、EJBs 的 Container 架構暨運作管理功能技術。期望藉由此項訓練，學習 J2EE 架構中各項相關的技術，以提升未來程式開發及系統維護所需的技能。

另實習結束後繼續參加電信管理論壇(TeleManagement Forum)所舉辦之國際性研討會，本公司為該組織之會員，參與該研討會以了解下一代維運支援系統(NGOSS)的架構與趨勢，及世界各國發展的情形與經驗，有助於本公司規劃、設計、開發下一代的維運支援系統。

寬頻網路整合管理系統技術—

Enterprise JavaBean Programming

暨參加 TM Forum Conference

出 國 實 習 報 告 書

目　　錄

寬頻網路整合管理系統技術—

Enterprise JavaBean Programming

暨參加 TM Forum Conference

出 國 實 習 報 告 書

1. 目的

本計畫(專案 830 計畫)目前正協助開發 ADSL/ATM 寬頻網路管理系統，本系統所使用的程式語言技術採用可攜性高的 Java 程式語言，而 SUN 公司之 Java 技術居於市場領先地位。職奉准至美國 SUN 公司教育訓練機構參加「Enterprise JavaBean Programming」訓練課程，此訓練課程教導有關 J2EE 相容 EJBs 的建立，以及如何將它們結合成強韌的企業應用程式等必要的知識。研習內容包括：Session and Entity Beans、EJBs 的 Container 架構暨運作管理功能技術。期望藉由此項訓練，學習 J2EE 架構中各項相關的技術，以提升未來程式開發及系統維護所需的技能。

另實習結束後繼續參加電信管理論壇(TeleManagement Forum)所舉辦之國際性研討會，本公司為該組織之會員，參與該研討會以了解下一代維運支援系統(NGOSS)的架構與趨勢，及世界各國發展的情形與經驗，有助於本公司規劃、設計、開發下一代的維運支援系統。

2. 過程(實習內容)

2.1 EJB architecture

Multi-tier distributed applications often consist of a client that runs on a local machine, a middle-tier that runs on a server that contains the business logic, and a backend-tier consisting of an enterprise information system (EIS). An EIS can be a relational database system, an ERP system, a legacy application, or any data store that holds the data that needs to be accessed. This figure shows a typical EJB multi-tier distributed system with three tiers: the client; the server, the container, and the beans deployed on them; and the enterprise information system

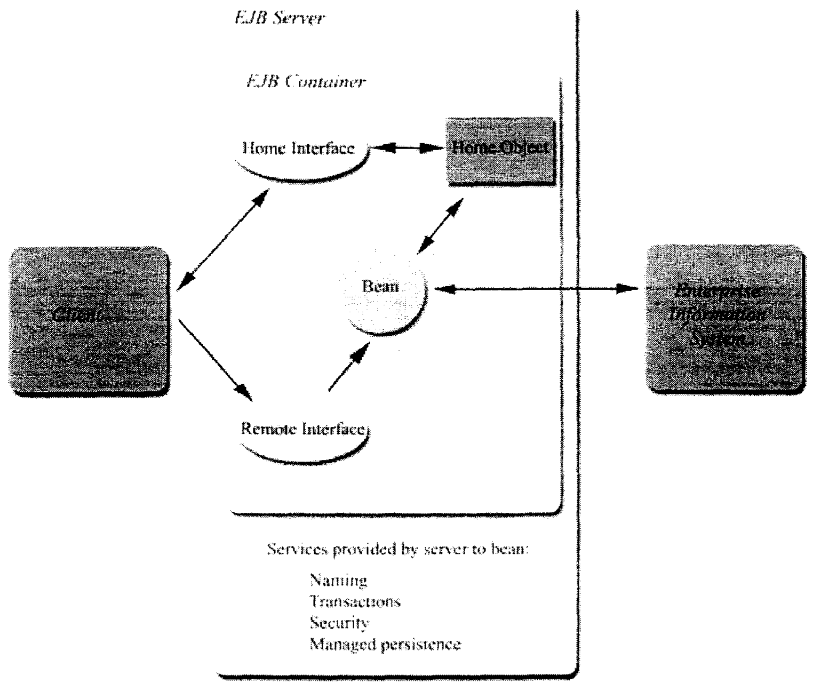

Figure 1 : EJB architecture diagram

2.2 The EJB Server

The EJB server provides system services to enterprise beans and manages the containers in which the beans run. It must make available a JNDI-accessible naming service and a transaction service. Frequently an EJB server provides additional features that distinguish it from its competitors. The Borland Enterprise Server AppServer Edition 5.0.2+ is an example of an EJB server.

2.3 The EJB Container

A container is a runtime system for one or more enterprise beans. It provides the communication between the beans and the EJB server. It provides transaction, security, and network distribution management. A container is both code and a tool that generates code specific for a particular enterprise bean. A container also provides tools for the deployment of an enterprise bean, and a means for the container to monitor and manage the application.

The EJB server and EJB container together provide the environment for the bean to run in. The container provides management services to one or more beans. The server provides services to the bean, but the container interacts on behalf of the beans to obtain those services. Almost always the EJB server and the EJB container are made by the same vendor and are simply two parts of an application server.

Although it is a vital part of the Enterprise JavaBeans architecture, enterprise bean developers and application assemblers don't have to think about the container. It remains a behind-the-scenes player in an EJB distributed system.

2.4 How an enterprise bean works

The bean developer must create these interfaces and classes:

- The remote home and/or local home interface for the bean
  The home interface defines the methods a client uses to create, locate, and destroy instances of an enterprise bean.
- The remote and/or local interface for the bean
  The remote or local interface defines the business methods implemented in the bean. A client accesses these methods through the remote interface.
- The enterprise bean class

Once the bean is deployed in the EJB container, the client calls the create() method defined in the home interface to instantiate the bean. The home interface isn't implemented in the bean itself, but by the container. Other methods declared in the home interface permit the client to locate an instance of a bean and to remove a bean instance when it is no longer needed. EJB 2.0 beans also allow the home interface to have business methods called ejbHome methods.

When the enterprise bean is instantiated, the client can call the business methods within the bean. The client never calls a method in the bean instance directly, however. The methods available to the client are defined in the remote or local interface of the bean, and the remote or local interface is implemented by the container. When the client calls a method, the container receives the request and delegates it to the bean instance.

2.5 Type of Enterprise Beans

An enterprise bean can be a session bean, an entity bean, or a message-driven bean

2.5.1 Session Bean

There are two types of session beans: those that can maintain state information between method calls, which are called stateful beans, and those that can't, which are called stateless beans.

1. Stateful session beans

A stateful session bean executes on behalf of a single client. In a sense, the session bean represents the client in the EJB server. Stateful session beans can maintain the client's state, which means they can retain information for the client.

Stateful session beans are objects used by a single client and they maintain state on behalf of that client. For example, consider a shopping cart session bean. As the shopper in an online store selects items to purchase, the items are added to the "shopping cart" by storing the selected items in a list within the shopping cart session bean object. When the shopper is ready to purchase the items, the list is used to calculate the total cost.

Session beans can be short-lived. Usually when the client ends the session, the bean is removed by the client

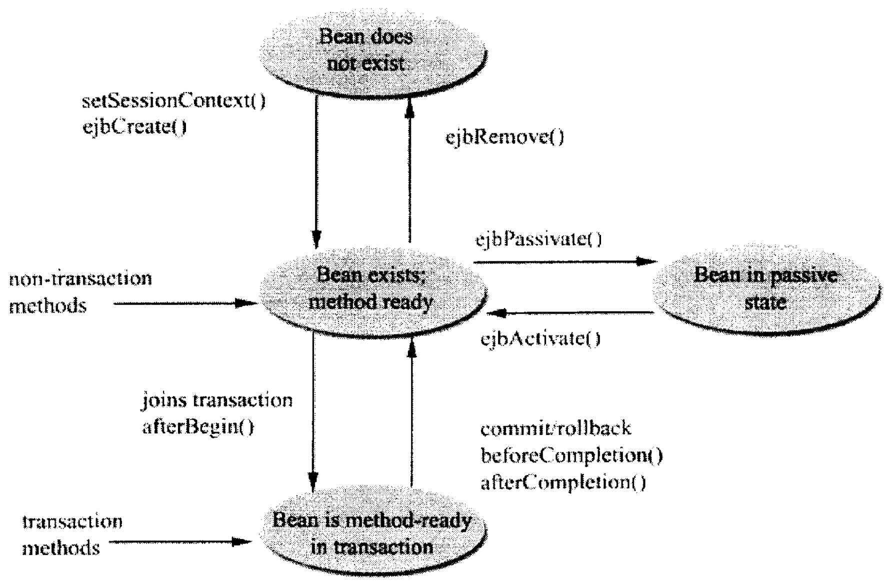

Figure 2: stateful session bean life cycle

2. Stateless session beans

Stateless session beans don't maintain state for any specific client. Because they don't maintain conversational state, stateless beans can be used to support multiple clients.

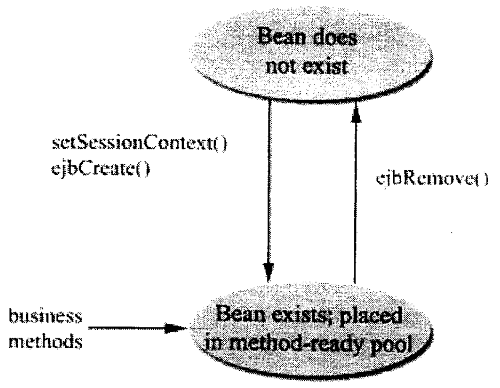

Figure 3: stateless session bean life cycle

2.5.2 Entity Beans

An entity bean provides an object view of data in a database. Usually the bean represents a row in a set of relational database tables. An entity bean usually serves more than one client.

Unlike session beans, entity beans are considered to be long-lived. They maintain a persistent state, living as long as the data remains in the database, rather than as long as a particular client needs it.

The container can manage the bean's persistence, or the bean can manage it itself. If the persistence is bean-managed, the bean developer must write code that includes calls to the database.

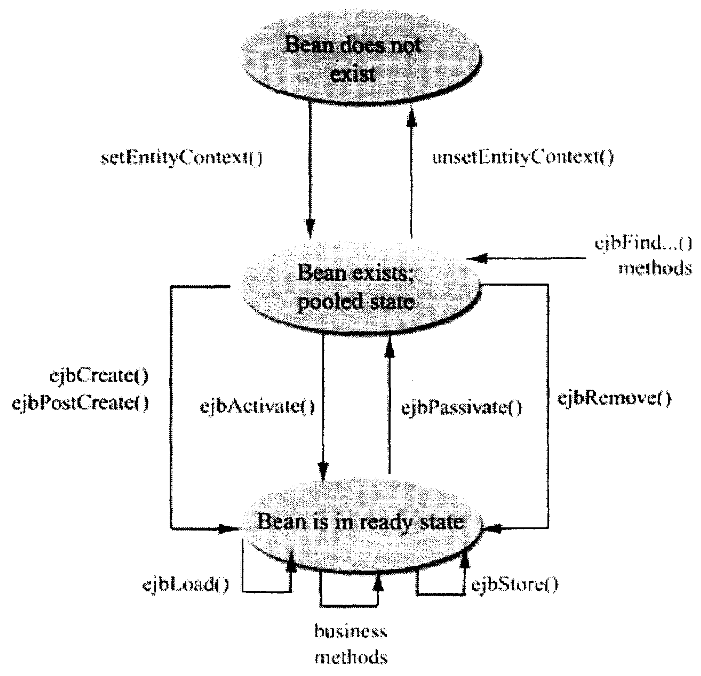

Figure 4: stateless session bean life cycle

2.5.3 Message Driven Beans

The EJB 2.0 specification introduced message-driven beans. They behave as a Java Message Service (JMS) listener, processing asynchronous messages. The EJB container manages the bean's entire environment.

Message-driven beans are similar to stateless session beans because they maintain no conversational state. Unlike session and entity beans, clients don't access them through interfaces. A message-driven bean has no interfaces, just a bean class. A single message-driven bean can process messages from more than one client. A message-driven bean is essentially a block of application code that executes when a message arrives at a particular JMS destination.

2.6 Remote and Local Access

An EJB 2.0 component can be accessed remotely or locally. Clients that access a remote bean use the bean's remote and remote home interfaces. A remote home is often referred to as the home interface. A client with remote access to a bean can run on a different machine and use a different Java Virtual Machine (JVM) than the bean itself. In method calls to a remote bean, parameters are passed by value, which helps maintain loose coupling between the client and the bean.

A client with local access to a bean must run in the same JVM as the bean it accesses. A local client won't be an external client application, but rather another enterprise bean or web component. In method calls to a local bean, parameters are passed by reference, resulting in a tighter coupling between the calling bean or web component and the called bean

.Like the remote interface, the local interface provides access to the bean's business methods, while its local home interface provides access to the methods that control the life cycle of the bean as well as its finder methods. Often entity beans that have a container-managed relationship with other entity beans have local access to them.

Because beans with local interfaces must run in the same JVM, there is no need for remote calls. Therefore, the overhead of serializing and transporting objects is reduced. Usually this means greater performance.

2.7 Developing entity beans

An entity bean directly represents data stored in persistent storage, such as a database. It maps to a row or rows within one or more tables in a relational database, or to an entity object in an object-oriented database. It can also map to one or more rows across multiple tables. In a database, a primary key uniquely identifies a row in a table. Similarly, a primary key identifies a specific entity bean instance. Each column in the relational database table maps to an instance variable in the entity bean.

Because an entity bean usually represents data stored in a database, it lives as long as the data. Regardless of how long an entity bean remains inactive, the container doesn't remove it from persistent storage.

The only way to remove an entity bean is to explicitly do so. An entity bean is removed by calling its remove() method, which removes the underlying data from the database. Or an existing enterprise application can remove data from the database.

2.8 Persistence and entity beans

All entity enterprise beans are persistent; that is, their state is stored between sessions and clients. As a bean provider, you can choose how your entity bean's persistence is implemented.

You can implement the bean's persistence directly in the entity bean class, making the bean itself responsible for maintaining its persistence. This is called *bean-managed persistence.*

Or you can delegate the handling of the entity bean's persistence to the EJB container. This is called *container-managed persistence.*

2.8.1 Bean-managed persistence

An entity bean with bean-managed persistence contains the code to access and update a database. That is, you, as the bean provider, write database access calls directly in the entity bean or its associated classes. Usually you write these calls using JDBC.

The database access calls can appear in the entity bean's business methods, or in one of the entity bean interface methods. (You'll read more about the entity bean interface soon.)

Usually a bean with bean-managed persistence is more difficult to write because you must write the additional data-access code. And, because you might choose to embed the data-access code in the bean's methods, it can also be more difficult to adapt the entity bean to different databases or to a different schema.

2.8.2 Container -managed persistence

You don't have to write code that accesses and updates databases for entity beans with container-managed persistence. Instead, the bean relies on the container to access and update the database.

Container-managed persistence has many advantages compared to bean-managed persistence:

- Such beans are easier to code.
- Persistence details can be changed without modifying and recompiling the entity bean. Instead the deployer or application assembler can modify the deployment descriptor
- The complexity of the code is reduced, as is the possibility of errors.
- You, as the bean provider, can focus on the business logic of the bean and not on the underlying system issues.

Container-managed persistence has some limitations, however. For example, the container might load the entire state of the entity object into the bean instance's fields before it calls the ejbLoad() method. This could lead to performance problems if the bean has many fields.

2.9 TM Forum Conference 摘要

此次 TM Forum Conference 會議，TeleManagement World 提供許多來自世

界各國操作與管理領域的先趨所做的精采的報告與系統展示，在此次 Las Vegas 所舉辦的會議主題包括：

- ✓ Competitive Strategies
- ✓ Revenue Management / OpEx to CapEx
- ✓ New Generation Business Cases
- ✓ Mobility
- ✓ SLA/QoS
- ✓ Service Resource Management
- ✓ The Service Provider Enterprise
- ✓ **New Generation Networks**
- ✓ Software Development
- ✓ Controlling Fraud

職所參與的主題在 New Generation Network 這個領域上的演講，TM Forum 所描述的 NGOSS 是一個可以提供快速且有彈性的整合電信公司的 OSS 系統架構概念， 將其歸納下列特性

- ■ Framework – Integrates Multiple Points-of-view(如 Business Process Analyst, System Designer…)
- ■ Methodology – Business Process Driven
- ■ Development – Model-based
- ■ Architecture – Tech Neutral & Tech Specific, based on Distributed Computing principles
- ■ Interoperability – Contract/Component-based using Shared Information Models
- ■ Compliance – Testable Adherence

NGOSS 架構整合多項觀點，如下圖所示：

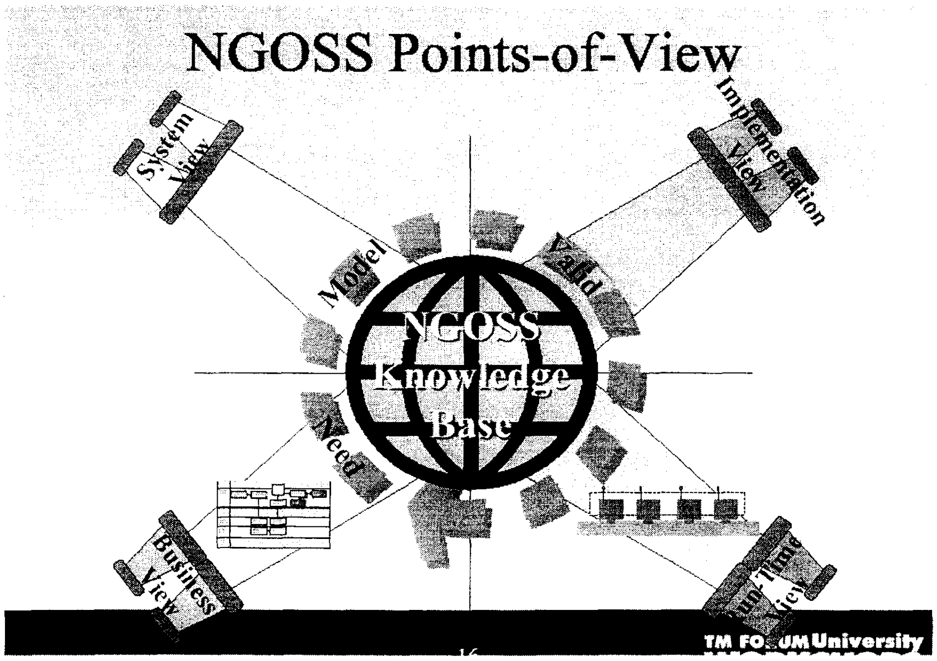

Figure 5 : NGOSS Points Of View

其中提出 eTOM 是 Business Process Automation，eTOM 提供 NGOSS Business View，它的 process, flows and information 是 NGOSS system view 需求的輸入

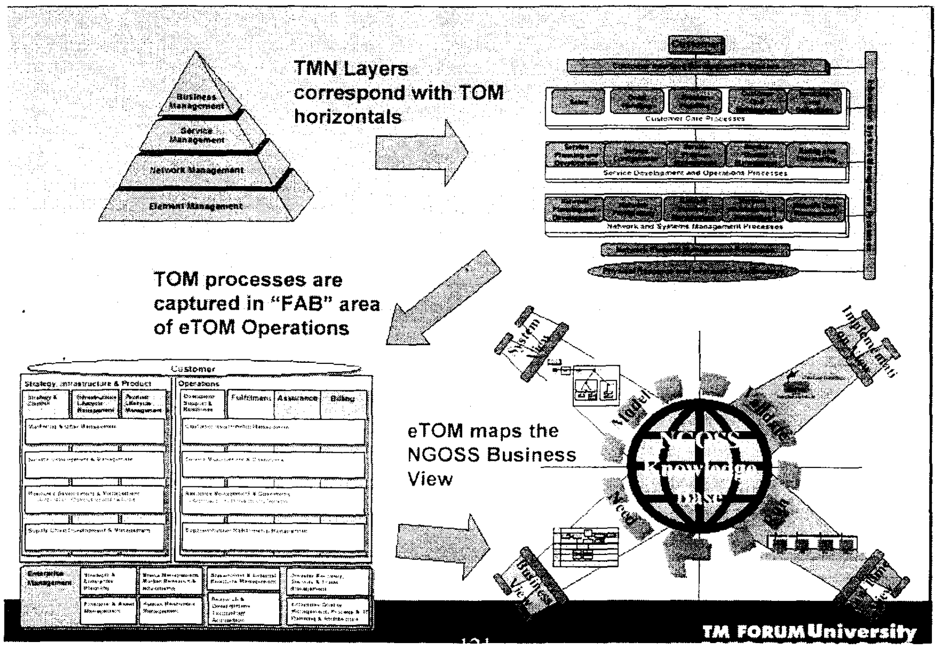

Figure 6: eTOM: Linkage to NGOSS

從此會議中一些先進的經驗報告，使職對 NGOSS 系統有了初步的認識，增進以後發展 OSS 系統的經驗。

3. 心得

Enterprise JavaBeans 定義的伺服端元件模型，可以讓我們開發的企業物件在不同的品牌的 EJB container 之間移來移去。元件代表的是一種簡單化的程式設計模型，可以讓開發人員專注在其企業用途上。EJB 伺服其要負責把該元件變成分散式物件，而且要管理各種服務，諸如交易、永續保存、同時共用以及安全性等等。除了定義 bean 的企業邏輯外，開發人員定義 bean 再執行期間的屬性時，採用的方法和選擇視覺物件的顯示屬性相似。元件的交易行為、永續保存、以及安全問題可以由一些屬性來定義。

總而言之，就是 EJB 使得分散式元件系統的開發更加簡單，因為 EJB 是以堅實的交易環境來管理分散式元件系統，且 EJB 提供相當簡單的平台，更具生產力，而且節省開發的工作。

4. 建議

Enterprise JavaBeans 2.0 和 1.1 是標準的分散式元件模型，遵照 EJB 標準做出來的產品陸陸續續從 IT 界的各個環節問世，J2EE 技術架構值得進一步深入的研究，未來發展公司的 OSS 系統時，可以朝 J2EE 架構發展，以期用較少的負擔，進而發揮其最大效益。

5. 其他相關事項

參考網頁：

- http://java.sun.com
- http://www.tmforum.org