JADE

Programación Distribuida y Tiempo Real

Movilidad - Migración

Movilidad y Migración

Conceptos generales

- Movilidad: Transferir programas o código ejecutable
- Movilidad de código y Migración de código/procesos se usan como sinónimos, aunque necesariamente no lo son.

Razones para migrar código

Conceptos generales

- Aumentar la eficiencia: repartir carga computacional y/o disminuir carga de la red de comunicaciones
- Permitir la carga dinámica de código: código no conocido a priori o código por demanda y/o mejorar la distribución/instalación del código en sistemas grandes y/o muy distribuidos

Razones para migrar código

Conceptos generales

- Últimamente, más relacionado con la eficiencia: mejorar la capacidad o velocidad de respuesta para un usuario

Migración - Modelos

Según lo que transfiere

- Débil: solamente el código de un proceso. No es un proceso en ejecución sino el código ejecutable
- Fuerte: código y estado del proceso. Es un proceso en ejecución, es lo que tradicionalmente se llama migración

Según quien inicia la migración

- Proactiva: el origen del código inicia la transferencia
- Reactiva: el destino del código inicia la transferencia

Movilidad Débil

¿Dónde se ejecuta el código?

- En el proceso receptor, se transfiere una porción del código. Ejemplo: JavaScript
- En un proceso separado. Ejemplo: Applet

Movilidad Fuerte

¿Que se hace con el proceso original?

- Migrar: el proceso literalmente se “**mueve**” y deja de existir en el sistema inicial/original
- Clonar: se crea una copia exactamente igual en otro sistema y ambos coexisten

Gráfico Clásico (Tanenbaum, 2Ed)

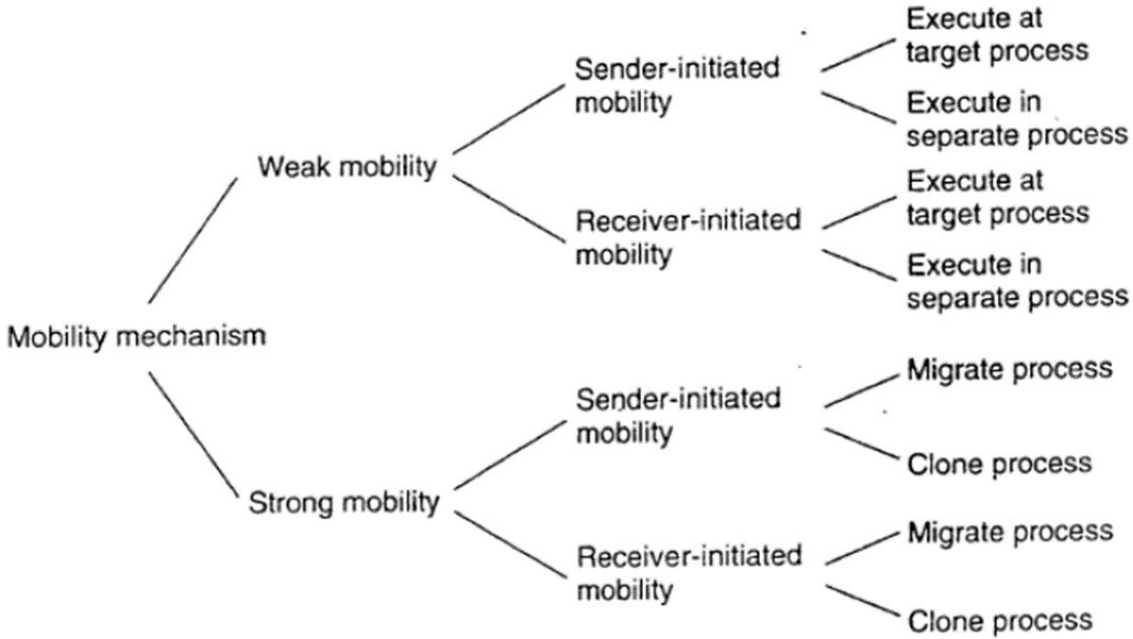


Figure 3-18. Alternatives for code migration.

JADE

JADE

en PDyTR

- Java Agent DEvelopment Framework (JADE): Plataforma de desarrollo de agentes en Java.
- Coordinación de Agentes FIPA: Permite la coordinación de múltiples agentes basados en el estándar FIPA.

JADE

en PDyTR

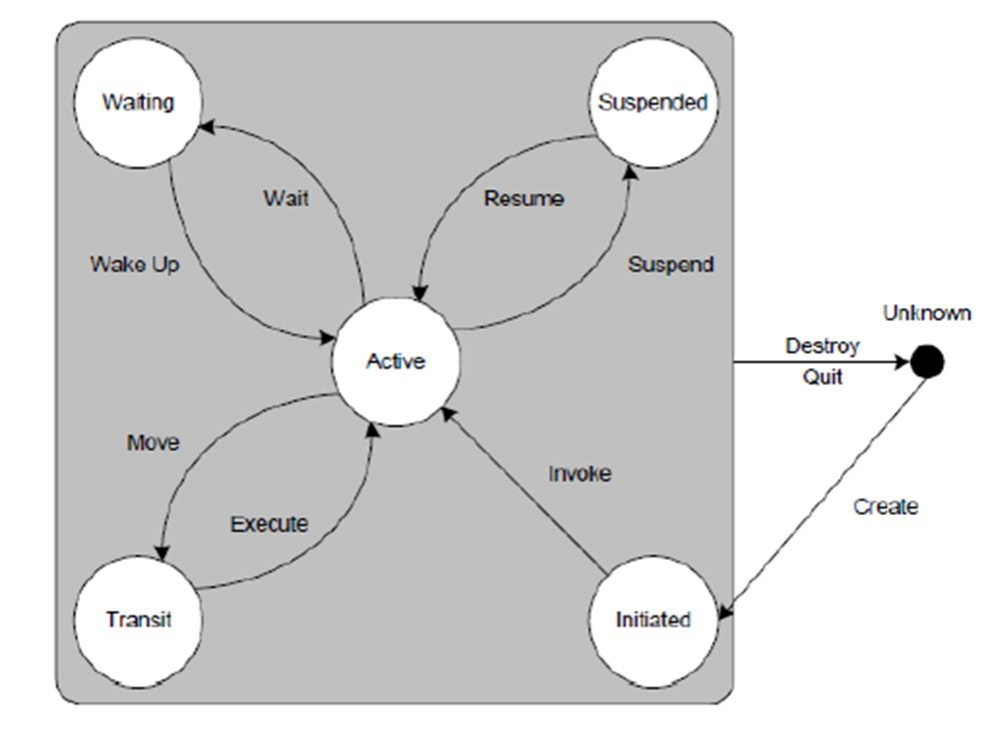
- Lenguaje de Comunicación FIPA-ACL: Proporciona una implementación estándar para la comunicación entre agentes.
- Detección de Servicios: Facilita la detección de los servicios disponibles en el sistema.

Agente

Agente

- Entidad autónoma, con capacidad de decisión y comunicación

Ciclo de vida



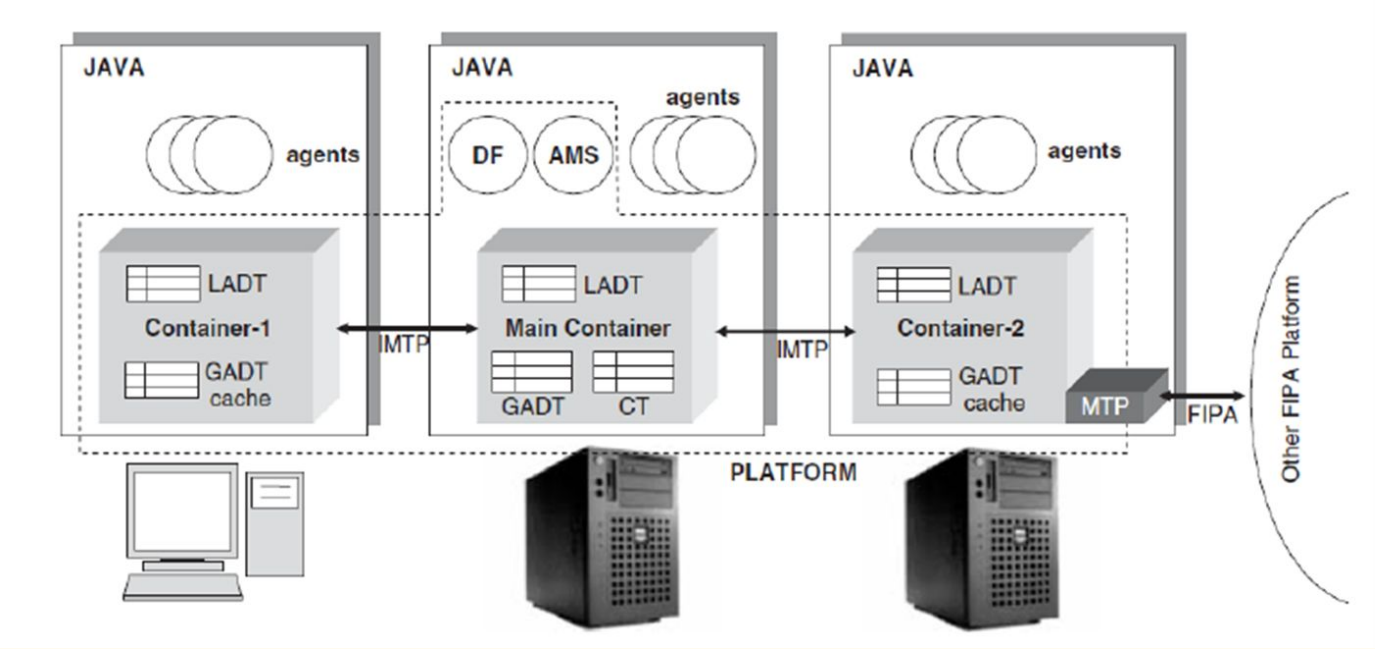
Contenedores

- Ambiente de ejecución inmediato del agente.

Plataforma

- Conjunto de contenedores

Contenedores / Plataformas



LADT: Local Agent Descriptor Table

GADT: Global Agent Descriptor Table

DF: Directory Facilitator

AMS: Agent Management System

Instalación y ejemplo

Dependencias

Dependencia

- JADE-all-4.3.0.zip

Prueba

- java -cp lib/jade.jar jade.Boot –gui
- java -cp lib/jade.jar jade.Boot -gui -localhost 127.0.0.1

Ejecución del ejemplo

```
javac -classpath lib/jade.jar -d classes
myexamples/AgenteMovil.java

java -cp lib/jade.jar:classes jade.Boot -gui

java -cp lib/jade.jar:classes jade.Boot -gui -container -host
localhost -agents mol:AgenteMovil
```

doMove()

Cuando un agente recibe un mensaje o realiza una acción que implica cambiar su estado o ubicación, el método doMove() se llama para llevar a cabo esa acción. Los agentes pueden ser móviles y cambiar de ubicación dentro de la plataforma JADE, y este método se utiliza para gestionar esas transiciones de manera personalizada.

afterMove()

Después de que un agente ha completado una acción que involucra su movimiento, el método afterMove() se llama automáticamente. Puede ser utilizado para realizar tareas posteriores a la mudanza, como actualizar su estado o notificar a otros agentes sobre su nueva ubicación. Esto permite al agente tomar medidas específicas después de haberse movido a una nueva ubicación.

¡Muchas gracias!

¿Preguntas?