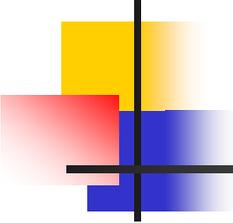


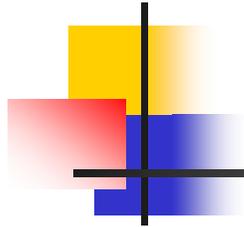
Agentes Móviles

Ingeniería del Software II
Curso 2009/2010
Sergio Ilarri Artigas
sillarri@unizar.es



Índice

- Agentes vs. Objetos
- Caracterización de la Movilidad
- Agentes Móviles: definición, ventajas, aplicaciones
- Movilidad: Fuerte y Débil
- Plataformas de Agentes Móviles
- SPRINGS:
 - *Proxies* dinámicos
 - Problema de *Livelock*



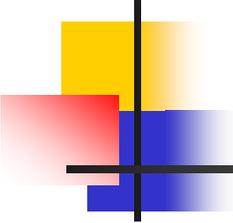
¿Diferencia entre objeto y agente?

Agentes vs. Objetos (I)

**Objects do it,
because
they are commanded to ...**

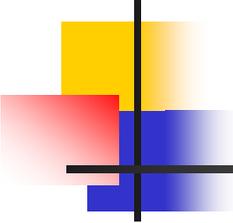
**Agents do it
for money ...**





Agentes vs. Objetos (II)

- Ambos encapsulan el estado
- Los agentes, además, encapsulan un comportamiento
 - Los objetos no, dado que no tienen ningún control sobre la ejecución de sus métodos
- Diferencia importante:
 - *Invocamos* métodos de objetos
 - *Pedimos* a los agentes que ejecuten acciones



Agentes vs. Agentes Móviles

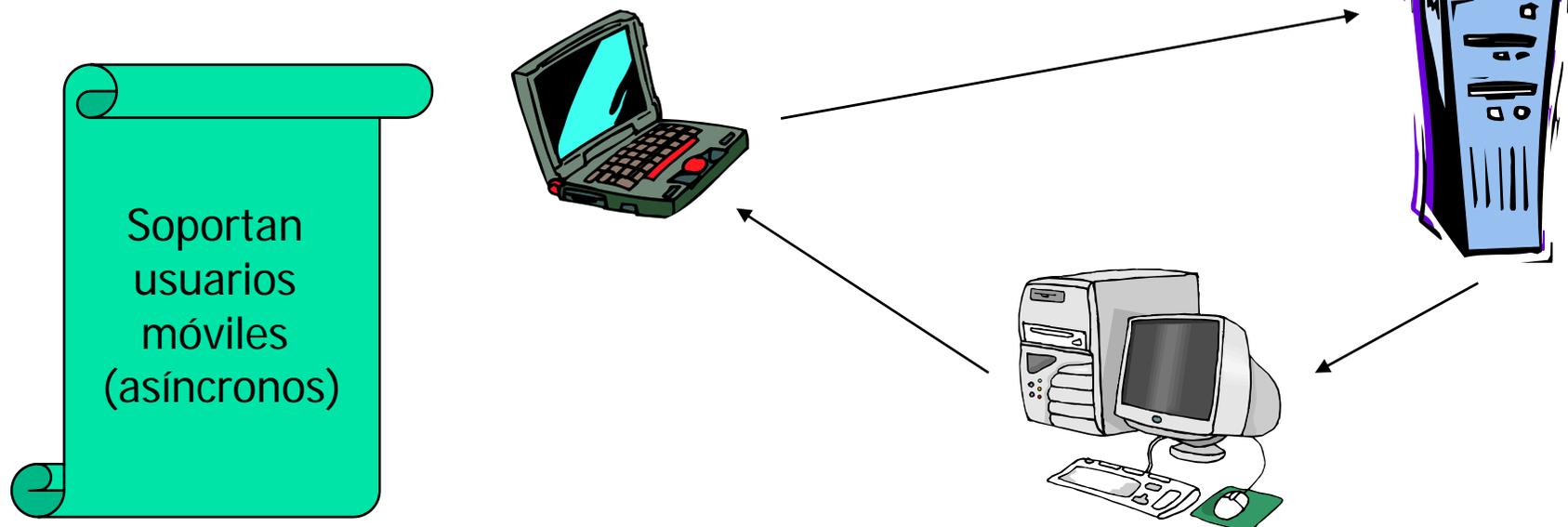
- Agentes:
 - Inteligencia artificial
 - Son "inteligentes"
 - Pueden moverse o no
- Agentes móviles:
 - Computación distribuida
 - Son móviles
 - Pueden ser inteligentes o no

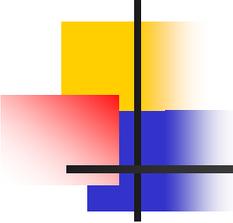
Conflicto y esperada
reconciliación

Agentes Móviles: Definición

Agentes software:

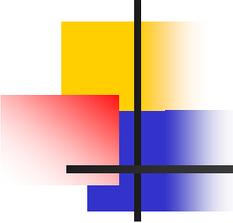
- Se mueven de ordenador a ordenador
- A petición del usuario, autónomamente
- Ejemplo: gestión de viajes





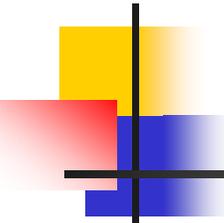
Los Ancestros...

- Frente al paradigma clásico cliente/servidor, surge la idea de movilidad de código
- Código móvil:
 - Evaluación remota
 - Código bajo demanda
 - Procesos móviles



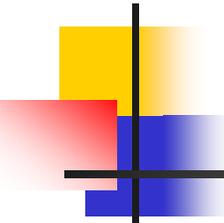
Paradigma Cliente/Servidor

- Aproximación clásica en sistemas distribuidos
- Cliente y servidor estáticos
- El cliente le pide algo al servidor
- El servidor lo hace y le devuelve al cliente un resultado
- Conceptos de lenguajes de programación:
 - *RPC*
 - *RMI*
 - *Servicios web*
- Frente a *C/S*, los agentes móviles
 - Paradigma de diseño alternativo
 - Paradigma de diseño complementario



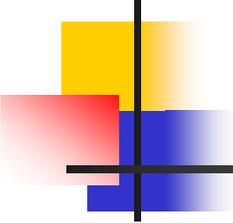
Paradigma de Evaluación Remota

- *Remote Evaluation (REV)*
- Extiende la idea de *RPC*
- El cliente envía código al servidor para que lo ejecute
- Ejemplos:
 - El lenguaje *Postscript* para impresoras
 - El envío remoto de trabajos por lotes
 - *NCL (Network Command Language)*
 - *SQL* remoto



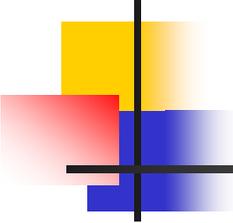
Paradigma de Código Bajo Demanda

- *Code-on-demand*
- También se envía código... pero del servidor al cliente
- Por tanto, se usan recursos del cliente
- Ejemplo: *applets* de *Java*



Procesos Móviles

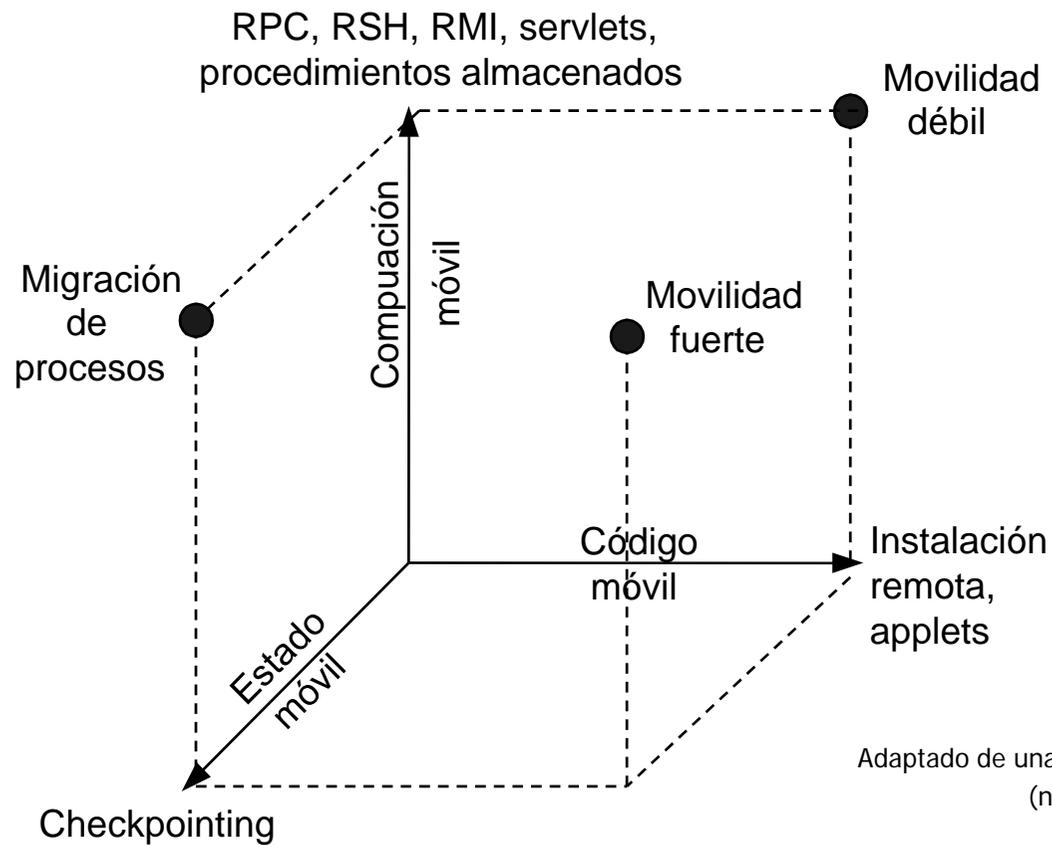
- Área de los sistemas operativos distribuidos (finales de los 80)
- Migración de procesos para balancear carga
- Ejemplo: *Sprite*
- Ejemplo de técnica de implementación: *checkpointing*
 - *Imágenes periódicas*
 - *Envío de la imagen*

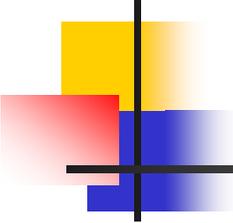


Procesos vs. Agentes Móviles

- Frente a los procesos móviles, los agentes móviles:
 - No sólo se mueven por balanceo de carga
 - Acceso a servicios
 - La migración no la inicia el *SO* o *middleware*
 - Agente autónomo
 - No sólo se mueven una vez
 - Migración multi-salto, itinerario

Caracterización de la Movilidad

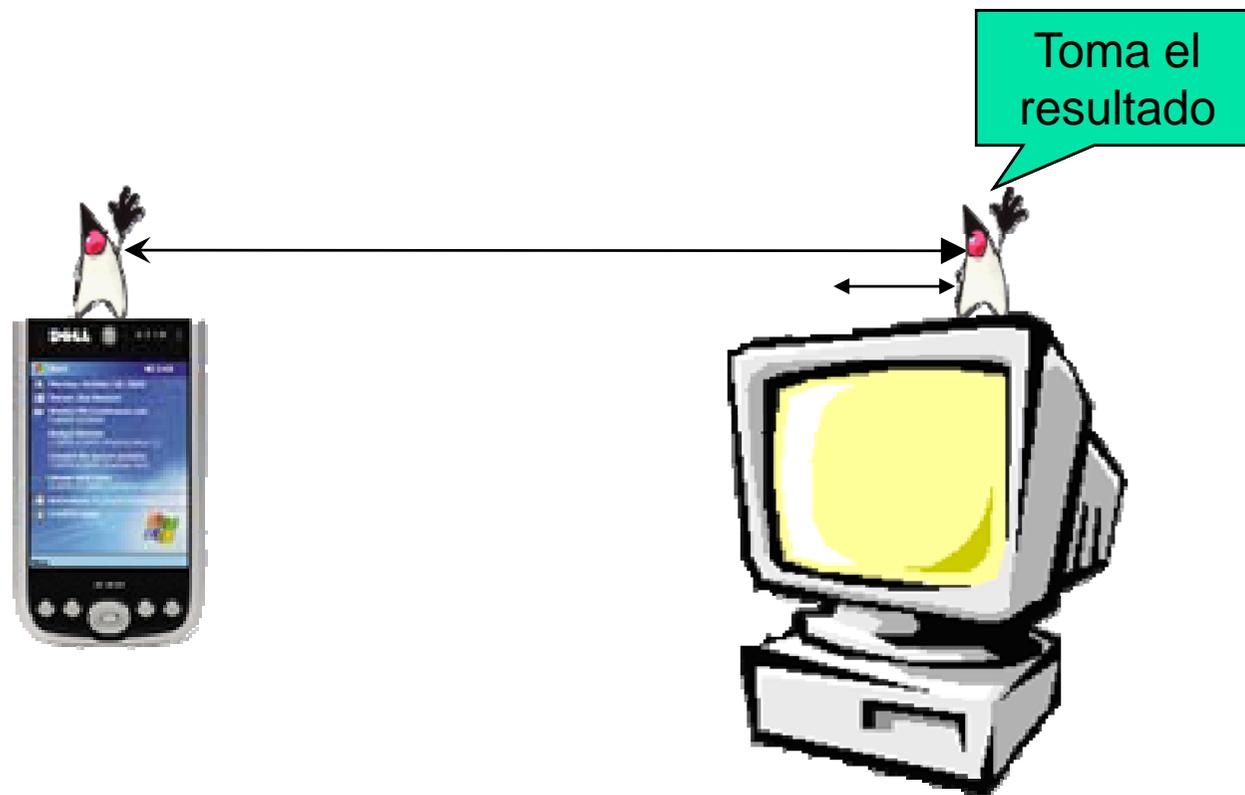




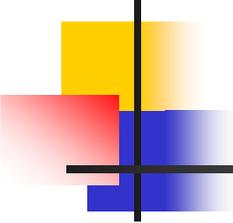
Agentes Móviles

- Programa que se ejecuta en un cierto contexto de ejecución o *place*, y viaja de *place a place*
 - Capaces de transportarse a sí mismos entre ordenadores
 - Necesita cierta infraestructura (plataforma de agentes)
 - Agentes móviles \neq código móvil
- Alternativa a RPC/RMI, pero también es una tecnología complementaria

Una Alternativa a C/S

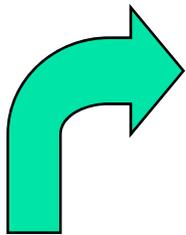


Los agentes móviles pueden reducir el uso de la red y la transferencia de datos

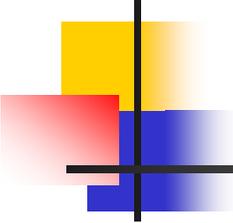


Características

- Algunas características interesantes de los agentes (móviles)
 - Autonomía: no/mínima interacción con el creador
 - Interoperabilidad: hardware, SO, etc.
 - Reactividad: cambios/eventos del entorno
 - Cooperación: objetivo común
 - “Inteligencia” (especialistas)
 - Movilidad

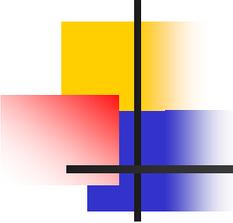


Agentes móviles en particular (sinergia)



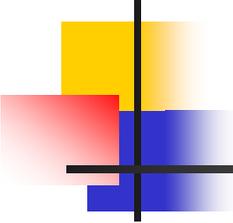
Ventajas

- Evitan instalaciones innecesarias
- “Salvan” la latencia de red: comunicación local
- Encapsulan protocolos (bases de datos, etc.)
- Asíncronos/autónomos: desconexiones
- Adaptativos
- Reaccionar entorno
- Moverse: balanceado de carga, localidad datos
- Integración de sistemas heterogéneos
- Robustez/tolerancia a fallos
- Adaptación de *interfaces*



Campos de Aplicación (I)

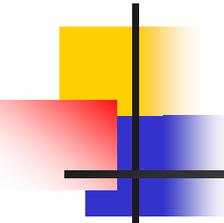
- Algunos entornos donde se han utilizado:
 - Recuperación de información distribuida
 - Procesamiento paralelo
 - Asistente personal
 - Diseminación de información
 - *E-commerce*
 - Gestión de red
 - heterogeneidad, monitorización, personalización, enrutamiento
 - Aplicaciones de *workflow*
 - *Brokering*
 - Entornos distribuidos: entornos móviles, ubicuos, inteligentes, P2P, ...
 - ...



Campos de Aplicación (II)

- ¿Y la *killer application*?
 - No hay una aplicación clave
 - ¿Cómo puede defenderse un lenguaje estructurado frente a un lenguaje ensamblador?

Agentes Móviles en Entornos Móviles



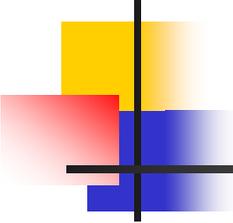
- Apropriados para computación inalámbrica:
 - Desconexiones:
 - Breve conexión: enviar agente a red fija
 - Antes de desconexión: coger agente de red fija
 - Descarga de trabajo del cliente
 - Contribuyen a limitar el uso de las comunicaciones inalámbricas:
 - Reducir los datos a intercambiar por el enlace inalámbrico
 - Evitar interacciones entre cliente y servidor
 - Sólo comunicar agente y resultado

Plataformas de Agentes

Móviles

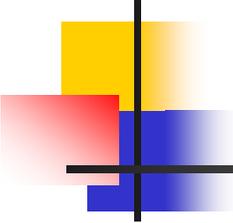
- Aglets
- Voyager
- Tryllian
- Grasshopper
- Jade, Tracy, Mole, SeMoa, ...
- Todas presentan problemas...
 - Por ejemplo, de escalabilidad y concurrencia
 - Por ello, desarrollamos SPRINGS





Movilidad: Cómo Funciona

- Los agentes móviles se crean en *places* y viajan entre *places*
- *moveTo(newHost)*
 - Se *interrumpe* la ejecución del *thread*
 - Se *serializa* el código, datos y (quizá) el estado de ejecución del agente (qué estaba ejecutando):
 - Movilidad fuerte y movilidad débil
 - El agente se reconstruye en el *place* destino y continúa su ejecución

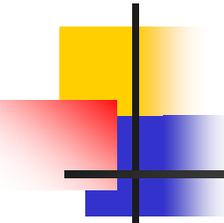


Movilidad Fuerte

```
public class AgenteMovFuerte extends Agente
{
    public static void main(String[] args)
    {
        System.out.println("En ordenador origen");
        moveTo(destino);
        System.out.println("En ordenador destino");
    }
}
```

Dificultades:

- No es posible con Java estándar
- No todos los recursos son móviles: múltiples *threads*, ficheros abiertos, etc.

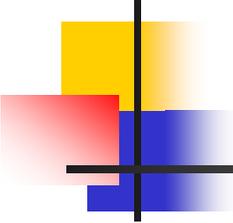


Movilidad Débil (Voyager)

```
public class AgenteMovDebil extends Agent {  
    public void metodoDestino (Object init) {  
        System.out.println ("En ordenador destino");  
    }  
    public void move(String destino) {  
        Iagent proxy = Agent.of(this);  
        System.out.println ("En ordenador origen");  
        proxy.moveTo(destino, "metodoDestino");  
    }  
}
```

callback

```
public static void main(String[] args) {  
    Voyager.startup("8000");  
    String serverClass = "AgenteMovDebil";  
    AgenteMovDebil ag = new AgenteMovDebil();  
    ag.move("tcp://fargo.sdsu.edu:8000");  
}
```



Movilidad Fuerte vs. Débil

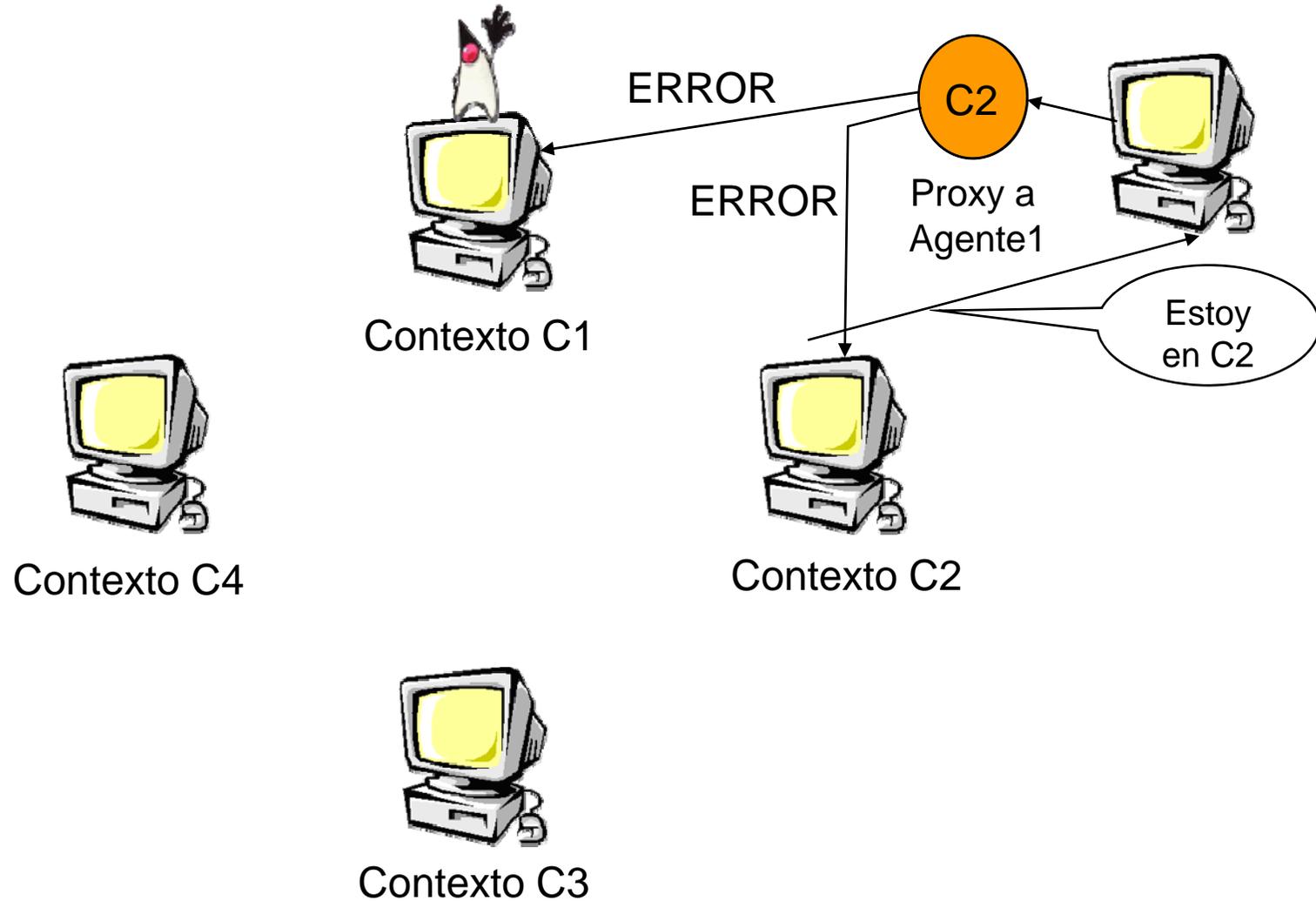
- Movilidad débil
 - No se transfiere el estado de ejecución
 - En destino, se ejecuta siempre un método predefinido o un método *callback* especificado por el programador
 - El programador guarda y restaura el estado de ejecución
- Movilidad fuerte
 - Se transfiere el estado de ejecución
 - Plataformas basadas en lenguajes de script
 - Plataformas basadas en Java:
 - Preprocesado para cambiar el código y capturar “a mano” el estado
 - Modificaciones a la máquina virtual de Java

SPRINGS

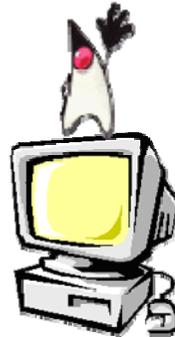


- Escalable, mucho mejor rendimiento (3000 agentes en *test* exigente)
- Transparencia de localización:
 - Llamadas: proxies dinámicos
 - `callAgentMethod("MovingAgentExample", "go" args);`
 - Movimientos: *callbacks*
 - `moveTo("C2", "end");`
- Reintentos automáticos
- Prevención de *livelock*

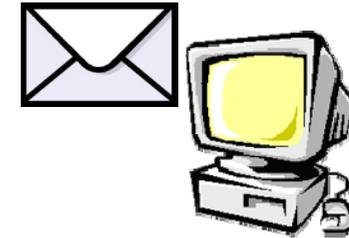
Problema de *Livelock*



Problema de *Livelock*



Contexto C1



Contexto C2



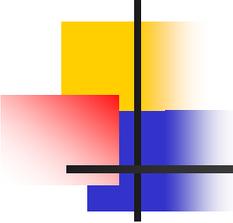
Contexto C4



Contexto C3

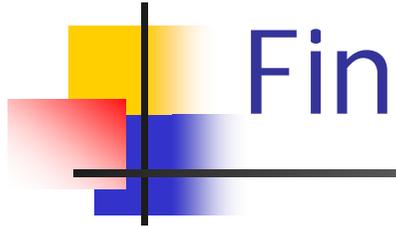
En SPRINGS

- Primero se actualizan los proxies, luego se renuda el agente
- Se retardan agentes muy rápidos



Referencias

- *Seven Good Reasons for Mobile Agents*, Danny B. Lange and Mitsuru Oshima, Communications of the ACM, Volume 42 , Issue 3 (March 1999), pp. 88-89, ACM Press, ISSN:0001-0782, 1999
- *Wireless Computational Models: Mobile Agents to the Rescue*, C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou, DEXA99 International Workshop on Mobility in Databases and Distributed Systems, August 1999, pp 428-433, IEEE Computer Society
- *SPRINGS: A Scalable Platform for Highly Mobile Agents in Distributed Computing Environments*, S. Ilarri, R. Trillo and E. Mena, 4th International WoWMoM 2006 workshop on Mobile Distributed Computing (MDC'06), Buffalo, New York (USA), IEEE Computer Society, ISBN 0-7695-2593-8, pp. 633-637, June 2006
- *Mobile Agents: Basic Concepts, Mobility Models, and the Tracy Toolkit*, Peter Braun and Wilhelm R. Rossak, Morgan Kaufmann Pub., 464 pages, ISBN 1558608176
- *Comparison and Performance Evaluation of Mobile Agent Platforms*, R. Trillo, S. Ilarri and E. Mena, The Third International Conference on Autonomic and Autonomous Systems (ICAS'07), Athens, Greece, IEEE Computer Society, ISBN 978-0-7695-2859-5, June 2007.



Fin

Gracias por vuestra atención