Objetos Locales contra Objetos Remotos

Gonzalo Mena Mendoza
mena@computer.org
Facultad de Informática de la Universidad Autónoma de Querétaro
23 de enero de 2006

Introducción

La lectura explica la diferencias entre el diseño de objetos locales y el de objetos remotos, con el objetivo principal de evitar errores comunes. Estas diferencias hacen necesarias muchas consideraciones de diseño ausentes en los sistemas centralizados. Entre estas consideraciones se encuentran aquellas sobre el ciclo de vida, las referencias entre objetos, la latencia de las peticiones, la activación y desactivación de objetos, el paralelismo, la comunicación entre objetos, la tolerancia a fallos y la seguridad.

Análisis

Se discuten las diferencias entre el diseño de objetos centralizados implementados en un lenguaje orientado a objetos y el diseño de objetos distribuidos. Las diferencias son en múltiples aspectos:

1. **Ciclo de vida**: el ciclo de vida de los objetos distribuidos en más complicado debido a que la información sobre su localización debe ser considerada. La creación de objetos distribuido se convierte en un problema de diseño mientras que para los objetos locales es normalmente un problema de implementación.
2. **Referencias**: las referencias entre objetos distribuidos son más complicadas y de mayor tamaño pues tienen que ser capaces de localizar al objeto, además de que debe añadirse información de seguridad y sobre el tipo de datos.
3. **Latencia**: la latencia de las peticiones a objetos distribuidos es comúnmente de dos órdenes de magnitud mayor que el gasto de llamar un método localmente. Esto puede requerir de optimizaciones en el diseño de las interfaces para mejorar los tiempos de respuesta.
4. **Activación**: los objetos pueden requerir de ser activados antes de servir una petición y luego desactivados cuando ya no son utilizados, lo cual se añade al tiempo de latencia ya de por sí largo. La activación y desactivación deben ser transparentes y los objetos que guardan estado deben ser persistentes.
5. **Paralelismo**: los objetos distribuidos tienen potencialmente la capacidad de realizar tareas con mayor rapidez que los locales al poder explotar el paralelismo. Puede necesitarse la implementación de controles de concurrencia.
6. **Comunicación**: para arreglárselas con la latencia de las peticiones, necesita haber formas adicionales de comunicaciones entre peticiones síncronas y entre peticiones “entre iguales”.
7. **Fallos**: los objetos distribuidos tienen mayor riesgo de incurrir en fallos, por lo que tanto el cliente como el servidor necesitan diseñarse de manera que puedan tolerar los fallos. Los clientes necesitan revisar que el servidor haya ejecutado la petición. Los objetos del servidor deben construirse de manera que puedan participar en transacciones.
8. **Seguridad**: los objetos distribuidos se comunican a a través de redes posiblemente inseguras y por lo tanto necesitan diseñarse de tal modo que la seguridad y privacidad de los datos no se vean afectadas.

Conclusión

El principal punto de este capítulo consiste en que el diseño de sistemas de objetos distribuidos requiere de muchas consideraciones de diseño que no existen para los objetos centralizados o que eran decisiones de implementación. Si bien esto hace más complicado el diseño y la utilización de objetos distribuidos, estos también presentan ventajas frente a los objetos centralizados ya que tienen el potencial de brindar un mejor desempeño a través de la paralelización y una mejor seguridad y tolerancia a los fallos al ser consideraciones a priori.

Referencias

Emmerich, Wolfgang. “Engineering Distributed Objects”, cap. 2, *Designing Distributed Objects*. Wiley (2000).