UMA-Java
Una extensión del lenguaje Java
que incorpora un Manejo transparente de Objetos Remotos
asegurando su corrección mediante el uso de Asertos.

El proyecto denominado ***UMA-Java:*** *Una extensión del lenguaje Java que incorpora un Manejo transparente de Objetos Remotos asegurando su corrección mediante el uso de Asertos*, tiene como objetivo el desarrollo de una extensión del lenguaje *Java*, uno de los lenguajes orientados a objetos que está siendo más utilizado hoy en día en el mundo de las aplicaciones sobre redes, ampliando su potencia y legibilidad.

El desarrollo de aplicaciones cliente/servidor usando *sockets* conlleva el diseño de un protocolo consistente en un lenguaje común entre el cliente y el servidor. El diseño de dicho protocolo no siempre es sencillo y es una fuente de errores tales como el *deadlock*.

En vez de trabajar directamente con *sockets*, las aplicaciones cliente/servidor, pueden ser desarrolladas mediante la invocación de métodos de objetos que están ejecutándose en otra máquina virtual Java (posiblemente en otro host). Java-RMI intenta mejorar estos aspectos, pero dejando al programador la tarea de buscar, enlazar y registrar los objetos remotos. Junto con esto, dos objetos de un mismo tipo "servidor" usando Java-RMI serán enlaces al mismo objeto, mientras en UMA-Java, serán enlaces al mismo objeto si asignamos sus identificadores u objetos distintos si usamos su constructor. UMA-Java realiza todo este trabajo de manera transparente al programador, el cuál sólo deberá indicar que un objeto es remoto mediante el uso de la palabra reservada *REMOTE* para que el sistema se encargue de lanzar un proceso en la máquina remota y comunicar ambos objetos, por lo que el programador tiene la sensación de realizar llamadas a métodos locales de una clase local, mientras el sistema es el encargado de pasar los argumentos, ejecutar el método y devolver los resultados de la máquina remota al objeto que ha realizado la llamada.

Junto con las características anteriores, el sistema añade a Java una nueva propiedad consistente en el manejo de asertos. Cada clase podrá definir un invariante de clase que

deberá ser preservado por todas las posibles modificaciones de un objeto. Así mismo, cada método de una clase podrá definir una expresión lógica a modo de precondición y otra a modo de postcondición, con lo que podremos asegurar la correcta ejecución de nuestro sistema basándonos en las proposiciones que han sido definidas.

Objetivos

Los objetivos de UMA-Java, por tanto, serán son los siguientes:

- Soporte de invocación de objetos remotos en diferentes máquinas virtuales Java.
- Soporte de retorno de resultados desde el servidor a los clientes.
- Integrar el modelo de objetos distribuidos en el lenguaje Java de una forma natural, mientras se mantiene la semántica de objetos ya definidas en Java.
- Realizar aplicaciones distribuidas lo más simples y legibles posibles.
- Preservar la seguridad que provee el *runtime system* de Java.
- Establecer un mecanismo que nos asegure que un programa que complete su ejecución realiza su tarea correctamente, debido al cumplimiento de los asertos definidos.

Etapas de Desarrollo

El proyecto se llevará a cabo siguiendo 6 etapas:

- Estudio de Java-RMI. Análisis de ventajas e inconvenientes.
- Diseño e Implementación de las clases Java necesarias para el proyecto (asertos, servidores remotos, etc.).
- Análisis lexicográfico del código fuente.
- Análisis sintáctico del código fuente.
- Generación de un entorno gráfico para la visualización y ejecución del sistema.
- Desarrollo de una herramienta de instalación desde CD-ROM a cualquier máquina.

Medios Materiales

Todo el proyecto será desarrollado en lenguaje C mediante el uso de las herramientas lex y *yacc* , junto con las clases que sean necesarias desarrollar en Java. Para su mayor portabilidad nuestro software incluirá una herramienta de instalación que permita su uso sin la necesidad de modificaciones tanto en PC's, Estaciones de Trabajo, etc.

- Compilador de *C*.
- Herramientas *lex, yacc*.
- JDK 1.1.4 (*Java Developed Kit*)
- Entorno de Red (PC's, estaciones de trabajo, etc.)

Referencias

- *Java Language Tutorial Reference*. Sun Microsystem Inc.
- *Remote Method Invocation System*. Sun Microsystem Inc.
- *Getting Started using RMI*. Sun Microsystem Inc.
- *RMI and Object* Serialization. Sun Microsystem Inc.
- *Distributed Object Programming Using Java.* Qusay H. Mahmond.
- *Compiladores : principios, técnicas y herramientas*. Alfred V. Aho, Ravi Sethi.
- *Introduction to compiling techniques : a first course using ANSI C, LEX and YACC* . J.P. Bennet.
- *Lex & yacc*. John R. Levine, Tony Mason, Doug Brown.
- *Lex & yacc*. Tony Mason and Doug Brown.