Selección de herramientas de software para los programas de control del radiotelescopio de 40M

P. de Vicente, Rubén Bolaño

Informe Técnico IT-OAN 2003-4

Índice

1. Introducción

Este documento describe las herramientas de software seleccionadas para escribir los programas de control del radioatelescopo de 40M del Observatorio Astronómico Nacional.

En todos los casos la elección de las herramientas de software se ha hecho atendiendo a los criterios que se relacionan a continuación y tras probarlas con ejemplos reales para verificar si se adecuan a las necesidades del OAN. Algunos de los ejemplos están disponibles en el CVS local del Observatorio Astronómico Nacional.

- Software que resuelva los problemas que se pretende que aborde. Aunque se cumplan las condiciones siguientes, si el software no sirve para los programas de control, no se utilizará.
- Software gratuito y con licencias de uso libre, como por ejemplo GPL (General Public License) o BSD, que permitan que el software sea reutilizable sin ninguna traba por otros observatorios o personas individuales.
- Software extendido en una gran variedad de instituciones y empresas, que asegure una comunidad que lo respalde.
- Software con perspectivas de crecimiento en el futuro, que avale la durabilidad del proyecto en el OAN.
- Software con documentación amplia y/o una comunidad de usuarios y desarrolladores que permitan su mantenimiento (resolución de problemas e implementación de nuevas prestaciones).
- Software que permita generar de modo sencillo documentación propia y que asegure su mantenimiento por un número mínimo de personas.

2. Sistemas operativos

El sistema operativo de elección para el ordenador maestro local de control y todos los ordenadores clientes que se comunican con él será **Linux**.

Entre los ordenadores clientes se encuentran aquellos que deben controlar los receptores y los backends. La monitorización y control de los receptores se hace en este momento utilizando un multiplexor de datos conectado a un puerto serie. En el caso de los backends es probable que los dispositivos externos de medida, como un voltímetro digital requieran el uso de un puerto GPIB. Las tarjetas GPIB-PCI de NI (National Instruments) funcionan correctamente desde Linux. Uno de nosotros (R. Bolaño) ha realizado pruebas con un contador, y un analizador de espectros utilizando una conexión GPIB.

El autocorrelador previsto, fabricado en IRAM , será controlado por una tarjeta MVME2434-1 con microprocesador PowerPC, que correrá bajo Linux porque está probado (por terceros) sobre otros modelos de la serie 2400, en particular sobre MVME2432-1.

Por otra parte ninguno de estos ordenadores, salvo quizá el autocorrelador, requiere en este momento el uso de sistemas operativos de tiempo real. De cualquier modo Linux, aunque no

está diseñado como sistema de tiempo real, permite la instalación de una capa de tiempo real (RTAI) que garantiza una latencia de hasta 10 microsegundos.

De acuerdo con C. Williams [1] la versión normal del núcleo de Linux presenta un tiempo de latencia medio de 88μs, el 92,84 % de los casos presenta latencias inferiores a 100 μs y una latencia máxima de 232,6 ms. Ese mismo núcleo con los parches de "Preemption" y de "low-latency", reduce el tiempo de latencia medio hasta 54 μs y el máximo tiempo de latencia hasta 1,3 ms. Estos parches aunque no son estándar en la serie de núcleos 2.4 están disponibles ya, y estarán incluidos en las series 2.6 de Linux.

En este instante el requisito más fuerte en el tiempo de latencia lo imponen las medidas de holografía que requieren tiempos de integración típicos de 250 milisegundos. Con objeto de comprobar los tiempos de latencia comandando un analizador de espectros de FFT desde un ordenador remoto a través de la red se hizo un programa de prueba que midiera la demora al comandar desde un ordenador remoto a través de una red local ocupada con tráfico normal hasta el ordenador al cual se conecta el analizador FFT empleando un puerto GPIB. Los resultados indican que el tiempo de latencia típico está comprendido entre 0,5 y 0,7 milisegundos.

Por tanto, Linux cumple todos los requisitos mencionados en la introdución para las necesidades actuales.

2.1. Distribución de Linux elegida

La distribución de elección es **Debian 3.0**. Se ha elegido Debian porque:

- es la distribución en uso en el OAN desde el año 1999 y disponemos de una rica experiencia acumulada.
- es extraordinariamente estable. En su detrimento no dispone de los últimos paquetes, pero siempre es posible sortear esta dificultad empleando rutas que proporcionan versiones nuevas recreadas ("backported") para la versión estable.
- permite la descarga de paquetes por red. Permite incluso montar una réplica de los servidores Debian en el propio observatorio minimizando el tiempo de instalación y actualización en los diferentes ordenadores.
- dispone de un sistema de paquetes que resuelve extraordinariamente bien las interdependencias y dispone de las herramientas para gestionarlos.
- proporciona abundante información para la creación de nuevos paquetes a partir de los "tarballs" de las aplicaciones.
- proporciona paquetes para 11 arquitecturas diferentes permitiendo que sea la única distribución que se emplea dentro del OAN, incluso para máquinas con microprocesadores especiales (itanium y motorola).

Los informes técnicos IT-OAN/CAY 2001-12 y IT-OAN/CAY 2002-6 describen la instalación y actualización de Debian en diferentes tipos de ordenadores, basados en procesadores Intel de 32 bits.

3. Lenguajes de programación

Se ha intentado en la medida de lo posible el empleo de lenguajes orientados a objetos. Este tipo de lenguajes permiten mantener más limpiamente grandes proyectos al utilizar clases y encaran los problemas relacionados con objetos reales de un modo más adecuado. El código resultante es más flexible, modular y escalable como resultado de la aplicación de las propiedades que ofrecen las clases como herencia, polimorfismo, sobrecarga de operadores, etc. Desafortunadamente los lenguajes orientados a objetos no han sido comunes en los ambientes científicos (aunque sí en los de ingeniería) y por tanto en el OAN no disponemos de una experiencia acumulada que proporcione una amplia base de programadores. Todo lo contrario, el número de personas que los manejan con cierta comodidad es bastante reducido.

A pesar de ello consideramos que es una apuesta con futuro por su gran impacto y penetración en numerosos ambientes técnicos y científicos, incluida la astronomía. Estos mismos criterios están presentes en otros proyectos de astronomía como ALMA o la actualización de GILDAS donde se han elegido los lenguajes orientados a objetos.

Se ha elegido:

- **C++**. Para el cálculo de operaciones que requieren más potencia y velocidad e interacción con el hardware.
- **Python**. Para la interacción con el usuario.
- **Python** + **Qt**. Para el interfaz gráfico del usuario en una primera fase.

C++ y Python, al igual que C, permiten el uso de programación de bajo nivel y por tanto el manejo sencillo de dispositivos de periféricos y de la memoria del ordenador. C++ comparte la misma sintáxis con C, y Python está programado en C y por tanto hereda de este casi toda su funcionalidad. El único inconveniente en C++ procedía del compilador de GNU, que en versiones antiguas no optimizaba tan bien el código como en C. Los programas escritos en C eran ligeramente más rápidos que en C++. Sin embargo esta circunstancia parece resuelta en los compiladores de la serie 3.1 y posteriores, disponibles actualmente como versiones estables.

Python es un lenguaje de “scripting” más lento que los lenguajes propiamente dichos como C o C++, pero extraordinariamente dotado. Los motivos por los que se ha elegido de entre Tcl y Perl fundamentalmente son:

- es un lenguaje orientado a objetos, aunque no disponga de todas las características de dichos lenguajes.
- es un lenguaje multiplataforma.
- permite la adición de módulos independientes del núcleo central del programa.
- dispone de versiones en Java.
- es un lenguaje con una gran velocidad de expansión.
- dispone de una comunidad muy potente que garantiza su futuro y la solución de problemas.

Para proporcionar entorno gráfico a los programas es necesario un toolkit gráfico. La elección aquí no es tan evidente como en los casos anteriores. La elección natural por tradición para Python sería Tk, que es el toolkit de Tcl. Sin embargo otros módulos gráficos para Python que se pueden evaluar. Por otra parte si deseamos emplear el mismo toolkit para C++ y para Python la elección natural es Qt. Qt es una biblioteca gráfica extraordinariamente potente. KDE está basado en ella y en los últimos meses ha sido incorporada en el núcleo de Linux en las series 2.5 para usarla en el configurador gráfico de este. Es multiplataforma, pero tan sólo es libre y gratuito en sistemas operativos Linux y Unix, y es de pago en Windows y OS X.

La programación del interfaz gráfico en Qt se puede realizar empleando una aplicación denominada “Qt-Designer”. Esta aplicación permite diseñar el entorno gráfico de modo sencillo. Los archivos gráficos se guardan en formato XML como archivos de texto y una aplicación tercera genera el código adecuado en C++ o en Python. El método de implementación en el resto de los programas se puede hacer empleando herencia.

De momento hemos descartado el uso de Java, por falta de personal y exceso de tareas, pero contemplamos su uso para el entorno gráfico como una posibilidad. Java no dispone de un entorno de desarrollo gráfico potente que sea abierto, lo que de momento impide su adopción inmediata en el proyecto del radiotelescopio de 40M.

Con objeto de estimar la viabilidad de C++, Python y Qt hemos utilizado estas tres herramientas en programas reales dentro del OAN con éxito en todos los casos. Vease por ejemplo el informe IT-OAN/CAY 2003-5.

4. Bases de datos

La programación del control del radiotelescopio de 40M requiere el uso de una base de datos en diferentes áreas. Se ha elegido MySQL como base de datos, porque cumple los criterios mencionandos en la introducción. Además MySQL:

- es una base de datos SQL, es decir que utiliza el lenguaje común estandarizado conocido como “Structured Query Language”.
- es una base de datos relacional lo que permite situar datos en diferentes tablas y enlazarlas mediante relaciones para combinarlas entre sí.
- es extremadamente rápida comparada con otras bases de datos. Vease [3] para información sobre bancos de pruebas y comparación con otros entornos.
- Qt proporciona métodos de acceso a MySQL.

El OAN ha utilizado MySQL en algunos pequeños proyectos y por tanto dispone de cierta experiencia en su manejo.

5. Comunicación entre procesos

La programación del sistema de control del radiotelescopio de 40M, requiere numerosos procesos en ejecución simultánea y que su comunicación se realice de modo transparente y de

modo sencillo. Los procesos pueden residir en la misma CPU o en CPUs diferentes. Para garantizar que la comunicación se hace abstrayendose del lugar físico de ejecución de los procesos, se ha elegido **CORBA** (Common Object Request Broker Architecture). En aquellos casos en los que no es posible implementar CORBA, como es el caso de la comunicación entre el ACU y el ordenador maestro de control de la estación, se emplearán sockets. Del mismo modo es probable que también se emplee memoria compartida y semáforos en casos especiales.

CORBA es una arquitectura e infraestructura que permite que aplicaciones de diferentes partes que se adhieren a este estándar se puedan comunicar de modo transparente. CORBA permite la comunicación entre ordenadores de diferentes fabricantes y en diferentes plataformas garantizando su interconectividad. CORBA se emplea de modo natural en los lenguajes orientados a objetos, si bien también se transporta a lenguajes como C que no están orientados a objetos.

Se ha elegido **omniORB** porque proporciona una implementación de CORBA para C++ y Python, sus tiempos de respuesta son bajos y porque incorpora todos los servicios necesarios para nuestra aplicación como por ejemplo el servicio de nombres. omniORB cumple los requisitos mencionados en la introducción y además proporciona entre otros:

- multihilos (un hilo por conexión)
- SSL (Secure Shell Layer)
- servicio de nombres
- soporte multiplataforma (Windows, Linux, Solaris, Mac...)

ominORB no es una implementación de CORBA en tiempo real, sin embargo, a la vista de los resultados obtenidos en diversas comparativas con otras implementaciones puede decirse que sus tiempos de respuesta son excelentes, ya que en algunos casos incluso superan a los de implementaciones en tiempo real.

Se hizo un estudio de varias implementaciones existentes de CORBA, entre ellas ORBit, MICO, JacORB, ORBacus, TAO y omniORB. Basándonos en el análisis de nuestras necesidades, en la decisión tomada en proyectos de gran envergadura como ALMA (donde se eligen omniORB para Python, JacORB para Java y TAO para C++), y dado que en nuestro caso no emplearemos Java en una primera aproximación, y que no existen requerimientos de tiempo real, la elección natural es omniORB.

6. Ingeniería de software: CVS

Una herramienta fundamental para la gestión de grandes proyectos de software con varios programadores trabajando simultáneamente con el mismo código es el **CVS** (Concurrent Versions System). El CVS es un sistema que permite almacenar en un repositorio central el código fuente, la documentación y una gran variedad de archivos, gestionando las diferentes versiones a lo largo del tiempo. El CVS mantiene todas las versiones del código fuente, las fechas de modificación, los comentarios realizados y el autor de los cambios, manteniendo un árbol de directorios complejo. El CVS mantiene una única copia del código en el repositorio y permite

consultas desde una cuenta anónima a través de la red, si se configura adecuadamente, y la modificación del código si el usuario está autorizado a través de una contraseña. Dispone de herramientas auxiliares que permiten ver el código, las diferencias entre versiones y otras datos de interés a través de un servidor HTTP.

El CVS permite trabajar con diferentes ramas de modo que se pueda separar el código en desarrollo de aquel que es estable y que sólo se modifica para corregir errores. También permite marcar directorios y archivos etiquetandolos para producir imágenes instántaneas que sirven para distribuir el código cuando se considera que es estable.

El OAN dispone de dos repositorios CVS, uno en la sede de Alcalá y otro en Yebes. El situado en Yebes es el que se emplea para el código relacionado con el radiotelescopio de 40M. En la actualidad el acceso al CVS a través de la web está restringido a la red local.

7. Documentación

La documentación que de lugar a informes técnicos, notas internas o cualquier otro tipo de documentación debe estar en formato PDF (Portable Document Format). Dada la hetereogeneidad de los sistemas operativos no se ha consensuado un formato único de documentación como "Word" de Microsoft, LaTeX, o OpenOffice, por lo que en este momento tan sólo se requiere el formato final en PDF.

La documentación que se genere automáticamente a partir de los comentarios del código fuente se puede obtener empleando la aplicación **kdoc** de KDE. Esta aplicación requiere que los comentarios estén en los archivos de cabecera (".h") en un formato especial muy similar al que emplea "javadoc". Para más información sobre el formato de la documentación se puede consultar [2]. La documentación generada puede estar en HTML, LaTeX, man, texinfo y docbook.

8. Bibliografía

Referencias

[1] C. http://www.linuxdevices.com/articles/AT8906594941

[2] Sirtaj S. Kang http://www.ph.unimelb.edu.au/ ssk/kde/kdoc/

[3] http://www.mysql.com/information/benchmarks.html

[4] Michele Zamparelli http://www.eso.org:8082/development/computing/docs/memos