Interfaz JAUS para Herramientas de Desarrollo en el campo de la Robótica

BAZÁN, FEDERICO A. JOST, MAURICIO G. MICOLINI, ORLANDO
bazanfedericoa@gmail.com|mauriciojost@gmail.com|omicolini@compuar.com
Laboratorio de Arquitectura de Computadoras

MATHE, LADISLAO
mathe@ieee.org
Grupo de Robótica y Sistemas Integrados

Facultad de Ciencias Exactas, Físicas y Naturales
Universidad Nacional de Córdoba, Argentina

7 de julio de 2010

Abstract

Software modularization presents important advantages when it comes to robotic development. That is based on the definition of well-known limits for each module that make the parallel development easier. Furthermore, that way of working allows the reutilization of one module in more systems.

There is a paradigm known as Components Based Software which follows the mentioned idea and needs definitions that include interfaces, responsibilities and certain communication rules between the members of the whole system. Precisely, the JAUS standard defines both a model of components with their interfaces, and an architecture, in the specific context of robotics. There are several SDKs, but they either lack a multi-platform support, or work with platforms whose features do not satisfy the needs of the developers, such as agile development of proof of concepts, robotic specific development resources (existent projects, community support, GUI, among others).

This work extends the possibilities of the implementation of a component, and allows the developer to take advantage of the benefits of the chosen platform. As a result, projects based on widely used tools used in robotics (such as MATLAB, Simulink and LabVIEW) can be easily integrated in a short period of time.

Keywords: JAUS, OpenJAUS, I-JAUS, MATLAB, Simulink, LabVIEW, manipulator, robotic arm software, SDK, toolbox, component.

1. Objetivos

El presente trabajo tiene por objetivos : ampliar el conjunto de HDRs (Herramientas de Desarrollo de Robótica) soportadas para la implementación de componentes JAUS y facilitar la integración de los mismos.

2. Marco Teórico

JAUS (*Joint Architecture for Unmanned Systems [1]*) es un estándar impulsado por el Departamento de Defensa de los EEUU (OUSD por sus siglas en inglés) destinado al software de vehículos no tripulados. JAUS define una arquitectura de componentes. Esto significa que establece reglas y restricciones para la comunicación entre componentes, funcionalidades de cada uno, sus interfaces, y el grupo al cual pertenecen.

3. Planteo del Problema

En el campo de la robótica de manipuladores, el desarrollo de software posee un alto grado de reusabilidad: existen bloques de uso recurrente tales como interfaz a sensores, bloques de control a lazo cerrado, bloques de manejo de servos, entre otros. Por esta razón, el software basado en componentes constituye una alternativa conveniente.

JAUS materializa las ventajas de los componentes en el campo de la robótica.

Existen distintas SDK (*Software Development Kit*) de JAUS que facilitan la implementación de un componente. Son principalmente las siguientes: OpenJAUS[1], Jaus++[2], RESquared[3], RI-JAUS[4].

Para desarrollar un componente haciendo uso de estas herramientas es necesario programar en lenguajes C/C++. Sin embargo, existen lenguajes de más alto nivel que brindan ventajas al desarrollo de robots como: facilidades para el desarrollo de GUIs, soporte para una variedad de periféricos tales como dispositivos DAQ y cámaras, *toolboxes* de modelado y simulación, etc. Estos entornos constituyen habitualmente la mejor opción para pruebas de concepto, e incluso para versiones finales del software de un robot. MATLAB, Simulink y LabVIEW son las principales HDR que brindan estas ventajas. En particular existe una alternativa diseñada por Ruél R. Faruque [4], la misma está limitada solo a la implementación de componentes JAUS bajo LabVIEW.

En este contexto, el usuario es capaz de escoger en qué plataforma desarrollar sus componentes, y luego hacerlos interactuar. Sin embargo sólo dispondrá de lenguajes como C/C++ y LabVIEW. Esto representa una seria limitación ya que existe una numerosa cantidad de desarrollos para MATLAB que podrían ser reutilizados en esta área. Es entonces importante la posibilidad de incluir a MATLAB entre las alternativas de implementación de componentes JAUS.

Por otro lado, un mismo grupo de trabajo puede desarrollar dos componentes en diferentes entornos haciendo uso de diferentes SDK JAUS, los cuales pueden incluso no basarse en un mismo paradigma. Esta multiplicidad de entornos y SDK, representaría un dificultad extra para los programadores. Entonces, queda claro que existe una ventaja si todos los entornos utilizan el mismo SDK, puesto que el desarrollador

[1] http://www.openjaus.com/

[2] http://sourceforge.net/projects/active-ist/

[3] http://www.resquared.com/JAUS-SDK.html

[4] http://www.repinvariant.com/dist/ri-jaus/0.9.0beta/ri-jaus.html

tendría que familiarizarse con uno solo.

Este trabajo se basa en dichas ideas para ofrecer una solución.

4. Planteo de la Solución

Se plantea entonces la implementación de una interfaz multiplataforma que permita el desarrollo de componentes JAUS desde cualquiera de las HDR mencionadas. Este escenario permite continuar desarrollando soluciones en las HDR habituales, e integrarlas mediante el estándar sin un mayor esfuerzo.

La solución consta de una librería de enlace dinámico (DLL) desarrollada en lenguaje C, a la cual se ha denominado Interfaz JAUS (I-JAUS[5]). Esta interfaz hace uso de una SDK denominada OpenJAUS. Su justificación puede ser vista en [6].

I-JAUS pone a disposición del desarrollador un conjunto de llamadas, tal cual son las principales definidas por OpenJAUS, las cuales pueden ser invocadas desde MATLAB o LabVIEW. Las mismas son:

- Inicializar un componente OpenJAUS.
- Agregar servicios al componente.
- Arrancar el componente.
- Actualizar sus mensajes salientes.
- Enviar dichos mensajes salientes.
- Obtener mensajes entrantes.
- Finalizar el componente OpenJAUS.

La Figura 1 muestra un componente desarrollado con I-JAUS que forma parte de un sistema JAUS.

[5]I-JAUS http://code.google.com/p/i-jaus/

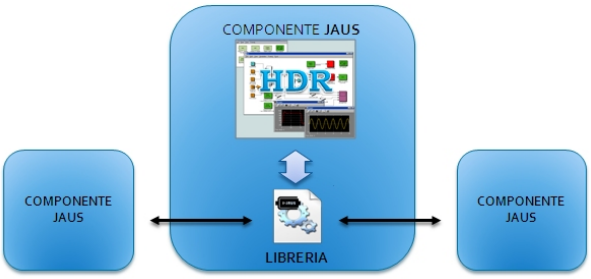

Figura 1: Componente JAUS implementado en HDR.

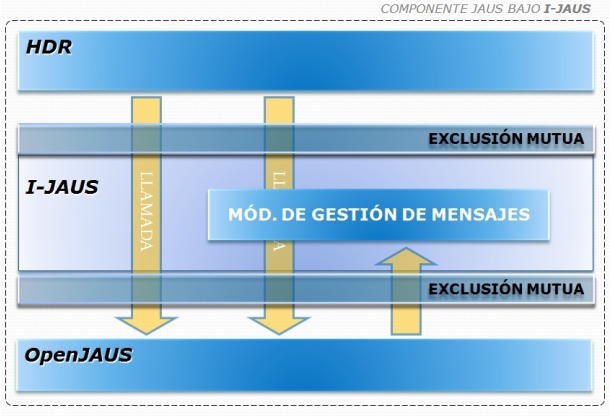

Figura 2: Arquitectura de I-JAUS.

La Figura 2 muestra la arquitectura de capas de un componente desarrollado mediante I-JAUS. Esta interfaz funciona como mediadora entre las solicitudes de la HDR y el componente OpenJAUS.

4.1. Problema de concurrencia

OpenJAUS permite definir una estructura de usuario para cada componente. La misma está destinada al almacenamiento de mensajes y toda aquella información que se considere necesaria. Además, un componente cuenta con estados y funciones asociadas a cada uno de ellos. Al poner en marcha un componente OpenJAUS,

es iniciado un único hilo encargado tanto del procesamiento asociado a cada estado como del procesamiento de mensajes. Al ejecutar un componente desde el hilo de la HDR, es puesto en marcha un nuevo hilo *OpenJAUS*. Este nuevo hilo tiene por objetivo atender la llegada asincrónica de mensajes. Esto está hecho para:

- Desacoplar el procesamiento de los mensajes entrantes hecho por OpenJAUS (recepción asíncrona) y el procesamiento del código de usuario sobre la HDR. Lo que permite menos latencia al generar llamadas desde la HDR misma.
- La librería maneja internamente llamadas bloqueantes, y con este nuevo hilo se evita el bloqueo de la HDR. Recíprocamente, la HDR podría bloquearse sin que esto bloquee al hilo de OpenJAUS (y no se reciban mensajes por ello).

Con la ejecución de dos hilos, es posible leer y modificar *simultáneamente* cualquier dato o mensaje contenido en la estructura de usuario de un componente. Esto genera situaciones propensas para la pérdida de integridad. Dicho problema es resuelto mediante un mecanismo de exclusión mutua, el cual es aplicado a cada una de las llamadas involucradas en el procesamiento de mensajes.

4.2. Tratamiento de mensajes entrantes

Los hilos de HDR y OpenJAUS poseen un flujo de datos asociado a los mensajes que se desea enviar o recibir. Este flujo de datos implica un acoplamiento entre dos entidades asincrónicas, que es resuelto por medio de una cola de mensajes.

En los casos para los que la capacidad de procesamiento de mensajes entrantes no es suficiente, se produciría una sobrecarga en la cola (potencial pérdida). Siendo que JAUS define en su mayoría componentes para conformar un sistema de control, la política de descarte debe asegurar la preservación de datos actualizados. Sin embargo, es importante aclarar que *JAUS* define un campo *Sequence Number* y un campo *ACK* en la cabecera de cada mensaje. Ambos, pueden ser utilizados en caso de ser necesario realizar un control de tráfico.

En la Figura 2, se presenta el Módulo de Gestión de Mensajes, que implementa la política mencionada. Este módulo fue diseñado de manera tal que un componente mantiene la última copia de cada tipo[6] de mensaje recibido. Además, soporta un mecanismo de prioridad dinámica basada en el orden de llegada de los tipos de mensajes. Esto evita la ocurrencia de inanición.

5. Pruebas

Como parte del proyecto I-JAUS, se han desarrollado un conjunto de experimentos que sirven de ejemplo del uso de la Interfaz. Estos muestran que es posible adaptar al estándar proyectos implementados en múltiples lenguajes.

Para controlar los manipuladores mediante comandos JAUS, se crearon componentes *Subsystem Commander (SSC)* que utilizan proyectos capaces de interactuar con el usuario mediante: mouse, teclado, GUI, joystick, voz, o seguimiento de ojos. Dichos SSC asumen el rol de *Operator Control Unit (OCU)*, y dan órdenes al componente que envuelve al manipulador.

5.1. Ejemplo de integración de componentes basados en múltiples HDR

El proyecto PUMA3D (MATLAB) ha sido envuelto en un componente *End Effector Pose Driver* (EEPD). Gracias a esto puede ser

[6]El tipo del mensaje está dado por su campo ID. Por ejemplo, dos instancias de mensajes *Set Joint Positions* corresponden al mismo tipo.

controlado mediante un OCU implementado en LabVIEW. Esto es mostrado en la Figura 3.

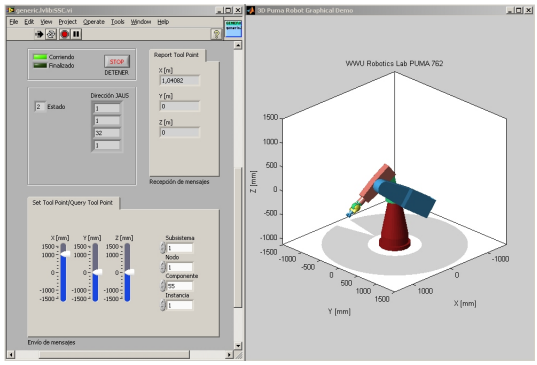

Figura 3: Componentes de LabVIEW (panel SSC) y MATLAB (PUMA3D en EEPD) interactuando.

El conjunto de demostraciones completo puede ser visto en la página de I-JAUS (http://i-jaus.googlecode.com).

6. Conclusiones

Entornos de desarrollo

Las pruebas han mostrado que la integración de componentes, implementados bajo diferentes entornos, permite una mayor agilidad en los tiempos de implementación. El desarrollador puede seleccionar la herramienta según sus funcionalidades, el soporte y los proyectos existentes que ofrezca como así también según sus propias habilidades en la misma.

En particular, la incorporación de MATLAB ofrece un entorno fácil de utilizar y una gran variedad de funciones matemáticas asociadas al control. Sin embargo, es importante destacar su interprete de comandos, el cual ha permitido menguar los tiempos de implementación de cada *test* realizado durante las pruebas de I-JAUS, en comparación con aquellas realizadas en C. Dicho interprete, permite escribir código en forma más dinámica, y luego reutilizar los comandos ejecutados mediante *scripts*, con un mínimo esfuerzo. Se puede afirmar entonces que, además de ser el entorno más utilizado en el ambiente estudiado, brinda facilidades que justifican aún más su elección.

Pruebas de concepto

Gracias a I-JAUS se pueden aprovechar una gran cantidad de trabajos desarrollados en MATLAB y LabVIEW, e integrarlos con el estándar JAUS de una manera muy sencilla y en un tiempo reducido. Por ejemplo, todas las demostraciones desarrolladas para este trabajo fueron realizadas en un tiempo inferior a dos semanas. Esta agilidad es consecuencia de dos puntos:

- La facilidad que presenta I-JAUS para ser integrado a un proyecto existente de alguna HDR en funcionamiento.
- El medio facilitado por I-JAUS, es decir el estándar JAUS, para que los proyectos se comuniquen entre sí.

Impacto

I-JAUS posee beneficios inmediatos para sus usuarios. De los observados, los más importantes son:

- División de un proyecto en bloques. Esto moviliza al grupo a finalizar componentes rigurosamente probados. Así es posible abstraerse de la implementación de partes consideradas fiables, para seguir experimentando en otras.
- Facilitación del aprendizaje de JAUS. Esto ocurre gracias a que MATLAB permite uso interactivo de un componente JAUS (desarrollado en I-JAUS) con su intérprete de comandos.
- Aumento de proyectos tenidos en cuenta. LabVIEW y MATLAB son cunas de desarrollos innovadores ahora integrables

mediante un estándar basado en componentes.

Aportes

El presente trabajo pone a disposición del usuario una librería de enlace dinámico (DLL) junto con la documentación necesaria para utilizarla tanto desde MATLAB como LabVIEW. Todo el material se distribuye libremente bajo licencia BSD[7].

Futuro

Aunque I-JAUS está actualmente basado en la versión 3.3 de los documentos *Reference Architecture*, su actualización es posible a efectos de acoplarse a las mejoras del estándar.

En vista de que el proyecto ha cumplido con el objetivo planteado, toma valor el hecho de extender las funcionalidades de I-JAUS, y así soportar el resto de los grupos de componentes del estándar.

Además de la Interfaz, este trabajo deja un conjunto de componentes implementados. Esto abre las puertas a cada usuario para que mejore dicha colección, cree nuevos componentes y los comparta con la comunidad.

Referencias

[1] Rowe, S. Wagner, C. (n.d.). *An Introduction to the Joint Architecture for Unmanned Systems*. EEUU: Cybernet Systems Corporation. Febrero, 1, 2010. http://www.openskies.net/papers/07F-SIW-089IntroductiontoJAUS.pdf

[2] *The Joint Architecture for Unmanned Systems. Reference Architecture Specification. Volume II, Parts 1, 2 and 3. Version 3.3*. (2007). EEUU: Jaus Working Group.

[3] *OpenJAUS*. Mayo, 30, 2009. http://openjaus.com/trac/openjaus

[4] Faruque, R. (2006). *A JAUS Toolkit for LabVIEW, and a Series of Implementation Case Studies with Recommendations to the SAE AS-4 Standards Committee*. EEUU: Virginia Polytechnic Institute and State University. Febrero, 1, 2010. http://scholar.lib.vt.edu/theses/available/etd-01142007-212355/unrestricted/jaus.pdf

[5] Sosa, O. (2004). *Design and Implementation of a Modular Manipulator Architecture*. EEUU: University of Florida.

[6] Bazán, F. Jost, M. (2009). *Interfaz JAUS para Herramientas de Desarrollo de Software de Robots*. Argentina: Universidad Nacional de Córdoba. Marzo, 6, 2010. Obtenido de: http://i-jaus.googlecode.com/files/ijaus_report.pdf

[7] I-JAUS http://code.google.com/p/i-jaus/