



PROYECTO OSMIUS



LICENCIA CREATIVE COMMONS
Reconocimiento-No Comercial-Sin obra derivada 2.0 Spain

Esta licencia permite:

- Copiar, distribuir, exhibir e interpretar este texto.
Siempre que se cumplan las siguientes condiciones.



Autoría-Atribución: Deberá respetarse la autoría del texto y de su traducción. El nombre del autor/a y del traductor/a deberá aparecer reflejado en todo caso.



No comercial: No puede usarse este trabajo con fines comerciales.



Sin obra derivada: No se puede alterar, transformar, modificar o reconstruir este texto.

- Se deberá establecer claramente los términos de esta licencia para cualquier uso o distribución del texto.
- Se podrá prescindir de cualquiera de estas condiciones si se obtiene permiso expreso del autor/a.

Este libro tiene una licencia Creative Commons Reconocimiento-No Comercial-Sin obra derivada 2.0 Spain. Para ver una copia de esta licencia visite <http://creativecommons.org/licenses/by-nc-nd/2.0/es/legalcode.es> o envíe una carta a Creative Commons, 559 Nathan Abbot Way, Stanford, California 94305, USA.

©2005, de la edición de Cein.



Definición de Proyecto

Introducción

Este proyecto consiste en el desarrollo e investigación de un framework software de código abierto cuyo nombre es **Osmius**.

Osmius es una distribución software de código abierto para la recolección de eventos desde sistemas, aplicaciones e instancias de todo tipo en red.

No ha sido diseñado como un producto sino como una solución a un subconjunto de problemas dentro de la gestión de los Sistemas de Información.

Osmius se basa en las últimas técnicas de programación orientada a objetos: Patrones de Diseño y FrameWorks (Marcos de Trabajo).

Mediante estas técnicas Osmius consigue:

- Reutilización Efectiva de Código.
- Reutilización Efectiva de Algoritmos.
- Código Multiplataforma.
- Adaptabilidad a nuevas necesidades y/o plataformas tanto hardware como software.

Se utiliza extensamente el Framework creado por el Profesor Douglas Schmidt de la Universidad de Vandervilt, **ADAPTIVE Communication Environment (ACE)**.

ACE es ampliamente utilizado tanto para aplicaciones comerciales como en entornos de investigación universitaria o empresarial. Ver apéndice a final del documento.

La arquitectura de Osmius permite adaptarse y cubrir la funcionalidad completa de entornos completos de:

- **Supervisión de Sistemas:**
Definición de eventos y alarmas. Presentación de los datos para Operadores y Administradores, Documentación de Eventos, ejecución de acciones automáticas, etc.

Correlación Inteligente de eventos para la Identificación real de problemas. Reglas parametrizables.



Visión orientada a Servicio y/o Proceso de Negocio. Gestión de Acuerdos de Nivel de Servicio (ANS), Informes de alto nivel. Cuadro de Mando Integrado.

Escalado de Incidencias.

Envío de alarmas a Móvil, SMS, Correo Electrónico y resto de Sistemas Comerciales de Supervisión.

▪ **Planificador de Tareas Distribuido:**

Definición de Tareas en entornos heterogéneos, definición de Cadenas de Tareas con precondiciones y poscondiciones definibles por el usuario, recuperación automática en tiempo de ejecución.

Seguimiento en tiempo real de tareas, informes de estado, duraciones, caminos óptimos.

Gestión de Alarmas y fallos.

▪ **Sistemas de Supervisión, Control y Adquisición de Datos (SCADA):**

Definición de valores y métodos de recogida de datos con certificado de autenticidad.

Capa de Presentación sobre mapas CAD.

Seguimiento en tiempo Real. Capacidad de modificaciones sobre los actuadores.

▪ **Otros:**

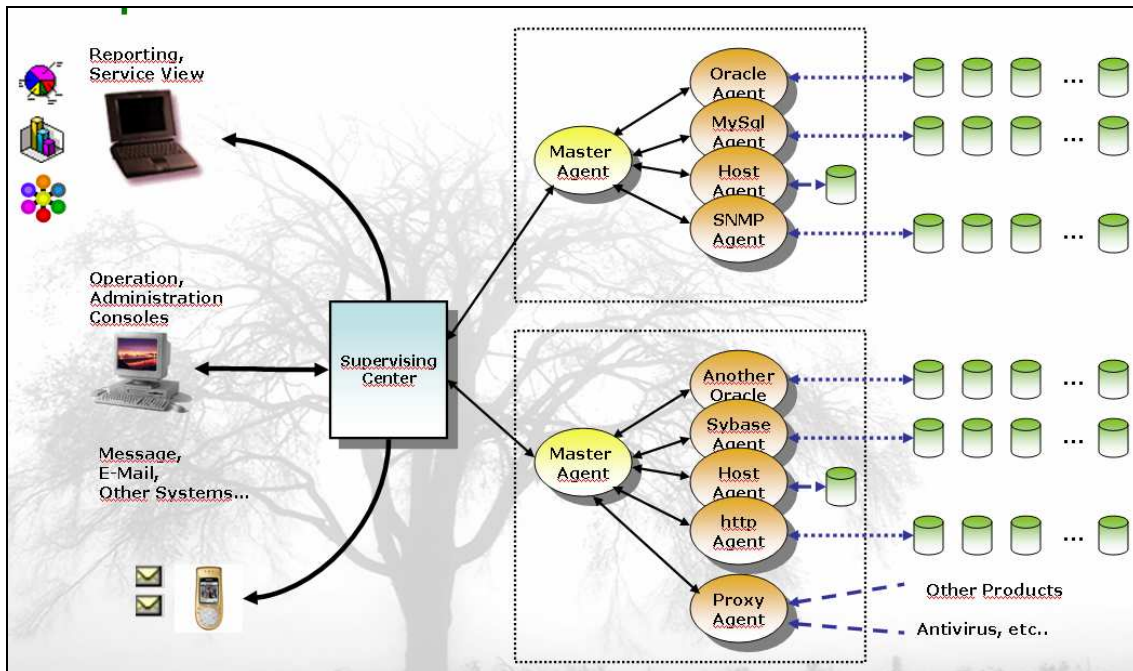
La filosofía de Osmius permite cubrir las funcionalidades demandadas por cualquier sistema que necesite recolectar datos basados en la ejecución de eventos a través de una red. Ejemplos de uso serían:

- Domótica: Lectura y actuación sobre sistemas domésticos.
- Video-Vigilancia.
- ...

Osmius cuenta con un motor inteligente basado en reglas para analizar las causas reales de los problemas en función de los eventos recibidos y para proponer nuevas reglas, y de consolas adaptadas a las diferentes funcionalidades a cubrir.



Ilustración 1 - Arquitectura de Osmius para Supervisión



Funcionalmente sus principales características son:

- **Actuador - Recolector:**
 Proporciona un motor para aplicaciones que necesitan de una estructura de agentes distribuidos en red, encargados de realizar acciones sobre determinadas instancias en red y devolver su resultado a una aplicación central.
- **Funciones Modulares:**
 La arquitectura de Osmius permite añadir nuevos módulos según aparezcan nuevos tipos de instancia en el mercado (nueva válvula de control de gasoducto por ejemplo), o interese conectarse a un nuevo software.



Descripción y Razones de uso de ACE

ACE se corresponde con las siglas de ADAPTIVE Communication Environment, y es un marco de trabajo – Framework – orientado a objeto, de código abierto y disponible gratuitamente, que implementa muchos de los patrones centrales para el desarrollo de software de comunicación.

ACE está orientado a desarrolladores de aplicaciones y servicios de alto rendimiento en red y tiempo real. Simplifica el desarrollo de este tipo de aplicaciones haciendo uso de la programación orientada a objetos, servicios y componentes.

Hoy en día desarrollar software debe tener como uno de sus principales objetivos ser independiente de plataformas tanto hardware como software y sistemas operativos, y no verse atado a ningún fabricante o proveedor en concreto. Las soluciones que hay en el mercado son:

- **Sun Java Virtual Machine (JVM)**, que proporciona una máquina virtual responsable de interpretar el código Java y traducirlo al lenguaje de la máquina real en la que reside. De esta manera las aplicaciones ven de forma transparente el sistema operativo subyacente.
- **Microsoft Common Lenguaje Runtime (CLR)**, que es la infraestructura sobre la que están construidos los servicios Web de Microsoft .NET. Es similar a JVM
- **ADAPTIVE Communication Environment (ACE)**, disponible de forma gratuita, con acceso a su código y altamente portable entre plataformas. ACE es un entorno y un conjunto de herramientas escritas en C++, que separa el sistemas operativo de la aplicación a través de la capa de Adaptación al Sistema Operativo.

Las principales diferencias entre ACE, JVM y CRL .NET son:

- ACE siempre genera un código compilado para la plataforma destino en lugar de un código intermedio que debe interpretarse después, eliminando un nivel de direccionamiento y optimizando el rendimiento en ejecución.
- ACE es Código Abierto, por lo que es posible adaptarlo o usar sólo un subconjunto en el que estemos interesados.
- ACE es capaz de ejecutarse en MÁS sistemas operativos y hardware que JVM y CLR.

Ejemplos de plataformas en las que corre ACE actualmente son:



- Para PC:
 - Todas los Windows de 32 y 64 bits.
 - WinCE.
 - Linux RedHat, Debian y Suse.
 - Macintosh

- UNIX:

Solaris	AIX
SGI IRIX	UnixWare
HP-UX	FreeBSD, NEtBSD,...

- Sistemas de Tiempo Real:

VxWorks	LynxOS
OS/9	Pharlap TNT
Chorus	QNX Neutrino
Integrity	RTEMS

- Grandes Sistemas:
 - OpenVMS
 - MVS OpenEdition
 - Tandem

Los beneficios resumidos de usar ACE son:

Aumento de la Portabilidad:

ACE está preparado para generar aplicaciones en la gran mayoría de Sistemas Operativos. No tenemos que preocuparnos por quedarnos atrapados con un solo sistema operativo.

Aumento en la Calidad del Software:

Los componentes de ACE están diseñados utilizando muchos patrones de diseño. Los patrones de diseño son parejas de problema-solución probados y contrastados por muchos desarrollos y años de experiencia. Todo esto mejora cualidades como la flexibilidad, la modularidad, la reusabilidad y la solidez frente a errores del software.

Mejora de la Eficiencia:

ACE ha sido cuidadosamente diseñado para soportar un amplio espectro de requerimientos de aplicaciones en cuanto a su Calidad de Servicio (QoS). Esto incluye requisitos de bajas latencias para aplicaciones muy sensibles a retrasos en las comunicaciones en red, de alto rendimiento para aplicaciones que demandan un fuerte uso del ancho de banda, y las necesidades de predicción en aplicaciones de tiempo real.



ACE sigue evolucionando a día de hoy y sus objetivos a futuro son:

- Incremento de Solidez
- Nuevas Funcionalidades. Se añaden constantemente.
- Optimizaciones orientadas a rendimiento.
- Adición de nuevas plataformas. Sistemas Operativos Nuevos
- Mejora de Documentación.
- Mejorar la Integración. CORBA.

ACE consta a día de hoy con más de 3.200.000 líneas de código, con más de 3.500 clases, mantenidas por más de 40 desarrolladores habituales y es usado por empresas y en proyectos comerciales y actuales.

Para más información:

<http://www.cs.wustl.edu/~schmidt/ACE-overview.html>

La curva de aprendizaje de Osmius está inclinada y precisa por parte del desarrollador unos conocimientos previos profundos en programación orientada a objeto, además de cierta atracción por los estándares y el código "bien hecho". Superada la curva de aprendizaje los beneficios obtenidos son sustanciosos.



Qué aporta Osmius a la comunidad científica y a la investigación universitaria

El desarrollo del software Osmius aquí descrito posibilita un marco de trabajo donde encuadrar diferentes tipos de colaboración con las universidades. Por un lado, las líneas de investigación que se abren en las que los investigadores pueden desarrollar rápidamente sobre la plataforma potentes herramientas operativas que comprueben sus hipótesis y testeen sus resultados, consiguiendo además productos finales utilizables y, por otro lado, la enseñanza a través de Osmius de toda la tecnología teórica y práctica que le subyace.

Los nuevos paradigmas de programación suelen aparecer para paliar los defectos de los ya existentes y aunque el paradigma orientado a objetos es el más extendido y admitido, sobre él están creciendo otros nuevos que suplen sus carencias, sobre todo, a la hora de desarrollar y mantener grandes proyectos. Empresas punteras como BEA, IBM y Oracle están auspiciando la programación orientada a Aspectos OAP (Mezini M., 2005) y se intenta unirla (Reddy y Kulkarni, 2005) con los metalenguajes o las arquitecturas conducidas por modelos MDA (Gokhale et al, 2002, Kleppe A., 2003) mientras, convive la programación orientación a servicios y a componentes.

Por otro lado la importancia actual en la programación concurrente y distribuida tanto en granjas de servidores en Grid, como en clusters (Massie M. et al, 2004) y la calidad de servicio, QoS, en los sistemas en tiempo real (Gill D., et al, 2004) se relacionan directamente en el desarrollo y en la aplicabilidad de Osmius.

El estado del arte en las técnicas de inteligencia artificial para el aprendizaje automático se mueve entre la lógica difusa, las redes Bayesianas (Peral, 1988) y las redes neuronales. Técnicas supervisadas como los árboles de decisión (Qinlan, J. R, 1986), las redes neuronales (Bishop, C. M, 1996) y las redes Bayesianas (Heckerman, D. et al, 1995) y no supervisadas como los mapas autoorganizados (Kohonen, T. 1984).

Estas herramientas permiten la construcción de sistemas expertos que sirven de apoyo en las tareas de clasificación, predicción y toma de decisiones. El comportamiento de estos sistemas no está fijado por unas pautas rígidas sino que es dinámico y adaptativo permitiendo resolver problemas complejos.



En definitiva, el carácter multidisciplinar de Osmius y el desarrollo de una plataforma base completa y robusta de código libre que permite su utilización para obtener diferentes productos competitivos: monitorización, SCADAs, domótica, video vigilancia etc. crea un marco de trabajo extenso que abre las posibilidades de investigación y enseñanza en las universidades de temas tan dispares como la ingeniería del software, los paradigmas de programación, la inteligencia artificial, los sistemas distribuidos, los sistemas en tiempo real, las granjas de servidores, la calidad de servicio etc., sin perder el objetivo de compaginar investigaciones punteras con resultados “comerciales” en tiempos cortos de desarrollo y con equipos humanos pequeños.

Referencias

Cleva, M., López V., Montero J. 2005 “A functional tool for fuzzy first order logic evaluation”. Proceedings jmlins

López V., Garmendia L., Montero J. 2005. “Fuzzy specification and computing state fuzzy relations”. Proceedings NOV05

Scacchi, W. 2004. “When is Free/Open Source Software Development Faster, Better, and Cheaper than Software Engineering?” In Koch, S. (ed.), Free/Open Source Software Development. IDEA Publishing Group.

Stotts P. D., Lindsey M., Antley A.. 2002. “An informal formal method for systematic JUnit test case generation”. XP/Agile Universe pp 131-143

Feathers M. 2002. “Emergent Optimization in Test Driven Design” Object Mentor.

Mezeni M., 2005. Proceedings of the 4th international conference on Aspect-oriented software development, 2005 ACM Press

Reddy S., Kulkarni V. 2005. “Using Aspect Orientation To Restructure A Model-Driven Development Framework”. Proceedings of the 4th international conference on Aspect-oriented software development, 2005 ACM Press



Gokhale A., Balachandran Natarjan, Douglas C. Schmidt, Andrey Nechypurenko, Nanbor Wang, Jeff Gray, Sandeep Neema, Ted Bapty, and Jeff Parsons, 2002, "CoSMIC: An MDA Generative Tool for Distributed Real-time and Embedded Component Middleware and Applications", Proceedings of the OOPSLA 2002 Workshop on Generative Techniques in the Context of Model Driven Architecture, Seattle, WA, November 2002.

Kleppe, A., Jos Warmer, Wim Bast, 2003. "MDA Explained, the Model Driven Architecture: Practise and Promise" Addison-Wesley 2003.

Massie M., Brent N. Chun, David E. Culler, 2004. "The ganglia distributed monitoring system: design, implementation, and experience". Parallel Computing 30 (2004) 817-840

Gill D., Jeanna M. Gossett, Joseph P. Loyall, Douglas C. Schmidt, David Corman Richard E. Schantz, and Michael Atighetchi. "Integrated Adaptive QoS Management in Middleware: An Empirical Case Study," 2004. 24th IEEE International Conference on Distributed Computing Systems (ICDCS), May 23-26, 2004, Tokyo, Japan.

Pearl, J.. 1988. "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference". Morgan Kaufman, San Mateo.

Qinlan JR. 1986. "Induction of Decision. Trees", Machine Learning 1, 81-106 Kluwer. Academic Publishers

Heckerman, D., Geiger, D., and Chickering, D. (1995). "Learning Bayesian networks: The combination of knowledge and statistical data". Machine Learning, 20(3):197--243.

Kohonen, T. (1984) Self-Organization and Associative Memory. Springer Series in Information Sciences, 8, Heidelberg.