SOFTWARE INDUSTRIAL FLEXIBLE

Daniel Díaz, Leandro Muñoz, Daniel Sirerol, Sandra Oviedo, Francisco Ibañez
LISI- Instituto de Informática – Dpto. de Informática
FCEFyN - Universidad Nacional de San Juan
CUIM – Av. Ignacio de la Roza 590 (O), Rivadavia – J5402DCS San Juan
{ddiaz, lmuñoz, soviedo, dsirerol, fibanez }@iinfo.unsj.edu.ar

Resumen

En este trabajo se pretende investigar y proponer técnicas, métodos y tecnologías que permitan el desarrollo de software flexible en ambientes industriales. El objetivo es generar métodos y técnicas para facilitar el desarrollo de software flexible en ambientes industriales. Las áreas de investigación son los sistemas de scheduling de producción, la generación de software para plataformas de hardware abiertas y la innovación.

Palabras clave: Software Flexible, Production Scheduling, Software Product Lines.

Contexto

Este trabajo se ejecuta en el LISI, Laboratorio Integrado de Sistemas Inteligentes, del Instituto de Informática de la Universidad Nacional de San Juan. El laboratorio en sus comienzos se dedicó al desafío de incorporar nuevas técnicas de la Inteligencia Artificial a la industria, allí nació la idea de aplicar tecnologías de Inteligencia Artificial a la solución de problemas de scheduling, y así, diversos productos fueron transferidos a empresas. Con el tiempo, la vinculación con empresas hizo necesario adquirir nuevas capacidades y conocimientos, por ello el LISI incursionó en el campo de la Logística con el objetivo de construir puentes entre la logística y las tecnologías de la información. En los últimos años el LISI ha abordado la difícil tarea de la innovación. El desafío actual del Laboratorio no es solo de generar y transferir conocimiento a las empresas, sino contagiar el espíritu innovador y establecer la cultura de la innovación en nuestra comunidad.

El trabajo actual guarda relación con esta línea, y surge como necesidad de generar conocimientos al proyecto marco “SIF - Propuesta para el Desarrollo de Software Industrial Flexible. Caso: Scheduling de Producción”.

Introducción

Desde hace tiempo se conoce que uno de los costos más significativos del ciclo de vida del software industrial está relacionado con la evolución del software para satisfacer las necesidades cambiantes del negocio. En el desarrollo de software industrial la flexibilidad es una herramienta que ayuda a reducir este costo. El término flexibilidad es una característica que puede poseer una entidad para adaptarse o cambiar ante contextos cambiantes. El concepto de flexibilidad en el software es usado de diferentes maneras, en el contexto de la ingeniería de software, dando lugar a términos alternativos estrechamente relacionados tales como mantenibilidad, adaptabilidad, modificabilidad y capacidad de evolución.

Por otro lado, la innovación está requiriendo desarrollar software flexible. Cada vez más la economía está pasando de una economía basada en el conocimiento a una economía basada en la creatividad y la innovación [1-3]. En este contexto es que el software está siendo usado cada vez más como el instrumento para hacer innovación.

Desarrollo flexible es la capacidad de responder rápidamente a las nuevas necesidades del mercado y las peticiones del cliente. Se trata de aumentar la

velocidad a la que las innovaciones y las ideas son llevadas al mercado. Las empresas sienten la necesidad creciente de ofrecer productos a tiempo. En este contexto y desde la perspectiva del desarrollo de software es necesario introducir el concepto y las técnicas necesarias para llevar a cabo el desarrollo flexible. Desde este punto de vista se puede definir al desarrollo de software flexible como el proceso que permite el desarrollo flexible.

La problemática del software industrial flexible es muy amplia es por ello que nuestra actividad de investigación ha sido acotada y dividida en 3 áreas de investigación.

La primera aborda los sistemas de scheduling de producción y tiene como objetivo la construcción de software flexible para este dominio.

La segunda trata con las plataformas de hardware abiertas y el objetivo es la construcción de software flexible para dispositivos electrónicos basados en plataformas de hardware abiertas.

La tercera es la innovación, aquí la investigación está centrada en el estudio de la técnicas de ingeniería de la innovación aplicada al desarrollo de software flexible. Otras tareas relacionadas con esta temática están referidas a contagiar el espíritu innovador y establecer la cultura de la innovación en nuestra comunidad universitaria.

Scheduling de producción.

El termino scheduling se atribuye al problema de asignar recursos limitados a tareas en el tiempo con el objeto de optimizar uno o más objetivos [4]. Los problemas de scheduling aparecen en una gran cantidad de dominios. Nuestra investigación solo contempla los problemas de scheduling en la industria, los cuales se conocen como production scheduling o programación de la producción.

En la industria, un sistema de scheduling es el corazón del sistema de planificación y control de la producción. Según [5], un sistema de scheduling se compone de diferentes módulos: (1) el módulo de base datos y base de conocimiento, (2) el módulo de motor de scheduling y (3) el módulo de interface de usuario.

El objetivo de esta área de investigación es la construcción de sistemas de scheduling flexibles. Un sistema de scheduling flexible es aquel que tiene la capacidad de responder rápidamente a las nuevas necesidades del mercado y las peticiones del cliente. Para alcanzar este objetivo se propone la construcción de sistemas de scheduling basada en motores de scheduling. Esencialmente, la idea es construir motores de scheduling que permitan integrarse fácilmente con la interface de usuario y las bases de datos de los sistemas integrados de gestión con que dispone la empresa.

El trabajo de investigación específicamente consiste en desarrollar una metodología que permita construir generadores de motores de scheduling en dominios específicos. Cualquier generador de motores de scheduling debe ser capaz de construir un motor customizable para una instancia particular del dominio bajo estudio (planta industrial).

La generación de motores de scheduling esta basada en la ingeniería de líneas de productos de software. La ingeniería de líneas de productos de software es un paradigma para desarrollar aplicaciones de software (sistemas y productos de software intensivos) usando una plataforma de componentes reusables y de customización en masa [6]. La ingeniería de línea de productos de software tiene dos procesos [7-9].

La Ingeniería de Dominio: Es el proceso responsable de establecer la plataforma reusable y por lo tanto definir las similitudes y las variabilidades de la línea de producto.

La Ingeniería de Aplicación: Es el proceso responsable de derivar aplicaciones de la línea de producto, a partir de la

plataforma establecida en la ingeniería de dominio.

El proceso de construcción del generador de motores de scheduling está basado en la ingeniería de domino. Mientras la construcción de un motor de scheduling está basada en la ingeniería de aplicación. Los métodos actuales de ingeniería de línea de productos no son aplicables directamente a la construcción de motores de scheduling y este es el desafío que plantea la investigación.

Actualmente la investigación se centra en el proceso de construcción del generador de motores de scheduling. Este proceso tiene 3 subprocesos: *Análisis de Dominio, Diseño de Dominio e Implementación de Dominio.* Los trabajos de investigación abarcan los dos primeros.

Para el *Análisis de Dominio* se propone una metodología para capturar y modelar motores de scheduling de un dominio. Esta metodología consiste en 2 actividades. La primera, *Definición del Dominio,* presenta técnicas de Ingeniería de Software, Ingeniería del Valor e Ingeniería de la Innovación. En esta actividad el objetivo es determinar las necesidades existentes y futuras del dominio con el objetivo de identificar los requerimientos. La segunda, *Modelado del Dominio*, presenta un método que permite modelar los requerimientos de un generador. La salida más importante del *Análisis de Dominio* es el modelo de características. El modelo de características incluye las características similares y variables de los miembros de una línea de producto de software, como así también dependencias y restricciones entre las características. El principal propósito es describir todas las posibles configuraciones (instancias) de una línea de producto de software [10].

El punto fuerte de la metodología es que partiendo de la nada se llega a un modelo de características de una manera práctica y sistemática.

Para el *Diseño de Dominio,* la investigación está centrada en el diseño de arquitecturas de software flexibles y evolucionables. Una arquitectura de un programa o sistema de computación es la estructura o estructuras de sistemas, la cual comprende los elementos de software, las propiedades externamente visible de estos elementos, y las relaciones entre ellos [11].

Las arquitecturas tienen distintas propiedades o características de calidad que permiten clasificarlas, diferenciarlas y evaluarlas. Desde el punto de vista del software flexible la característica que consideramos es la evolucionabilidad. La evolucionabilidad ha surgido como un atributo que se sustenta en la capacidad de un sistema para adaptarse a los cambios en sus requerimientos a lo largo de su vida con el menor costo posible mientras se mantiene la integridad arquitectural [12]. La analizabilidad, la integridad arquitectural, la cambiabilidad, la extensibilidad, la portabilidad, la testeabilidad y atributos específicos del dominio son considerado como subcaracterísticas de evolucionabilidad en [13].

Un aspecto importante es medir cuan evolucionable es una arquitectura. En la actualidad, existen métodos que pueden estimar cómo una arquitectura cumple con ciertos requerimientos de calidad y creemos que algunos de estos métodos pueden ser adaptados para medir la evolucionabilidad. Por ejemplo, las técnicas de evaluación basadas en escenarios pueden ser fácilmente adaptadas para evaluar los diseños con respecto a la capacidad de evolución; SAAM [14] puede ser especializada para trabajar con escenarios de evolución y ATAM [15] se puede utilizar para medir el equilibrio entre la evolucionabilidad y otro tipo de atributos de calidad.

Plataformas de hardware abiertas

Las plataformas de hardware abiertas son utilizadas para la creación de prototipos basados en software y hardware flexibles. Dichas plataformas pueden tomar

información del entorno a través de sus pines de entrada de toda una gama de sensores y pueden afectar aquello que les rodea controlando luces, motores, etc. Las plataformas poseen un microcontrolador que se programa utilizando un lenguaje de programación específico y un entorno de desarrollo. Si bien los proyectos hechos con las plataformas de hardware abiertas pueden ejecutarse sin necesidad de estar conectados a un ordenador, tienen la posibilidad de hacerlo y comunicarse con diferentes tipos de software. Algunos ejemplos de plataformas de hardware abiertas son Arduino [16] y Netduino [17].

Esta área de investigación intenta potenciar las prestaciones de la plataforma de hardware abierta mediante la aplicación de la tecnología de línea de productos de software con el objeto de permitir una configuración rápida y flexible para la obtención de una amplia variedad de productos de un dominio de aplicación.

El caso de estudio está enfocado al dominio de los sistemas de seguridad (alarmas). El trabajo consiste en la realización de una línea de productos de sistemas de alarma. Con esta línea de producto se puede construir desde sistemas muy simples (un sensor y una central) a sistemas complejos (varias centrales ubicadas en lugares geográficamente distintos). El esquema de funcionamiento de construcción de una alarma es muy simple y consiste en los siguientes pasos.

1. Elicitación de los requerimientos del cliente con el objeto de establecer el producto requerido.
2. Mediante la tecnología de línea de productos de software se obtiene el código que configura la plataforma de hardware abierta con los requerimientos del cliente.
3. Transferencia del código generado a la plataforma de hardware abierta.
4. Aprestamiento de la alarma personalizada o customizada para ser entregada al cliente.

Nos referimos a personalizada cuando decimos que la alarma satisface las necesidades de un cliente en particular y customizada cuando la alarma satisface las necesidades comunes de un grupo o conjunto de clientes.

Innovación

Como se ha mencionado en algunos párrafos anteriores, la innovación se ha convertido en una necesidad absoluta para hacer frente a los desafíos globales y las tendencias del futuro. De esta necesidad surge que el software flexible es uno de los elementos necesario para la innovación. El desarrollo de un producto requiere de la ingeniería de la innovación. El software es un producto y por lo tanto, como cualquier producto, la ingeniería de la innovación se puede aplicar a la fase de concepción para determinar las necesidades actuales y futuras de los usuarios del producto. Por otro lado el software cada vez se esta integrando más en productos de consumo masivo tales como electrodomésticos, teléfonos, aparatos de gimnasia, automóviles, entre otros. En algunos productos constituye el mecanismo utilizado por la innovación para generar nuevos productos. Es por ello que consideramos que la innovación puede ayudar a la generación de software y recíprocamente el software puede ayudar o facilitar la innovación. Por lo tanto el desarrollo de software flexible puede valerse de los métodos de la ingeniería de innovación y recíprocamente la ingeniería de la innovación requiere de software flexible para la creación de productos innovadores. El estudio de esta sinergia es el objetivo de esta área e investigación.

Otra motivación del equipo de trabajo es desarrollar e instalar la innovación en el ámbito universitario, para lo cual se están llevando a cabo distintas acciones entre las que se destaca la organización de un campamento de innovación con el lema “Instalando la Cultura de la Innovación”, se trata de una actividad outdoor de 3 días para alumnos de 4to y 5to año de las diferentes carreras. Durante el campamento

están previstos ejercicios de creatividad e innovación que propicien el trabajo colaborativo con el objeto de explorar las capacidades creativas multidisciplinarias.

Resultados y Objetivos

- Marco Metodológico para Desarrollar Sistemas de Planificación Avanzada de la Producción, D. Díaz, F. Ibañez, R. Forradellas, V EnIDI 2011, Encuentro de Investigadores y Docentes de Ingeniería, San Rafael - Mendoza, Noviembre 2011.
- A Domain Analysis Approach for Building Customized Scheduling Engines, D. Díaz, F. Ibañez, R. Forradellas enviado al Journal Intelligent Manufacturing - Springer en proceso de corrección/ revisión.

Formación de Recursos Humanos

El plan de formación de recursos humanos considera:

- Tesis de doctorado en Ciencias de la Computación de Daniel Díaz.
- Tesis de grado de Lic. en Sistemas de Información de Daniel Sirerol.
- Dirección de la beca de Iniciación - UNSJ de Leandro Muñoz.

Otras actividades

- Dictado del curso titulado Dinámica de Sistemas, cursos de temáticas relacionadas con el proyecto.

Referencias

[1] K. Dehoff and D. Neely, "Innovation and product development: Clearing the new performance bar. ," in *Booz Allen Hamilton* http://www.boozallen.com/media/file/138077.pdf, 2004.

[2] B. Nussbaum, R. Berner, and D. Brady, "Get creative! How to build innovative companies.," in Business Week. vol. August 1 http://www.businessweek.com/magazine/content/05_31/b3945401.htm., 2005.

[3] C. Leadbeater, *We-think: Mass innovation, not mass production.* London: Profile Books., 2008.

[4] K. R. Barker, *Elements of sequencing and scheduling*. New York: John Wiley and Sons, 1974.

[5] B. P.-C. Yen and M. Pinedo, "On the design and development of scheduling systems," in *Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, 1994, pp. 197 - 204.

[6] K. Pohl, G. Böckle, and F. J. v. d. Linden, *Software Product Line Engineering: Foundations, Principles and Techniques*: Springer, 2005.

[7] D. M. Weiss and C. T. R. Lai, *Software Product-Line Engineering - A Family-Based Software Development Process*. Massachusetts: Addison-Wesley, 1999.

[8] G. Boeckle, P. Knauber, K. Pohl, and K. Schmid, *Software-Produktlinien: Methoden, Einführung und Praxis (in German)*. Heidelberg: dpunkt, 2004.

[9] F. van der Linden, "Software Product Families in Europe: The ESAPS and CAFÉ Projects," *IEEE Software*, vol. 19, pp. 41-49, 2002.

[10] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," in *Technical Report CMU/SEI-90TR-21* Pittsburgh, PA Software Engineering Institute, Carnegie Mellon University, 1990.

[11] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice 2nd ed*, 2003.

[12] D. Rowe, J. Leaney, and D. Lowe, "Defining systems evolvability-a taxonomy of change," in *International Conference and Workshops on Engineering of Computer-Based Systems (ECBS)*, 1998.

[13] H. P. Breivold, I. Crnkovic, and M. Larsson, "A systematic review of software architecture evolution research," *Inf. Softw. Technol.*, vol. 54, pp. 16-40, 2012.

[14] L. B. R. Kazman, G. Abowd, and M. Webb, "SAAM: A Method forAnalyzing the Properties of Software Architectures," in *Proceedings. 16th International Conference Software Engineering.*, 1994, pp. 81-90

[15] M. K. R. Kazman, M. Barbacci, H. Lipson, T. Longstaff, and S.J.Carriere, "The Architecture Tradeoff Analysis Method," in *Proc. Fourth International Conference Engineering of Complex Computer Systems*, 1998.

[16] Arduino, "website," http://www.arduino.cc/es/.

[17] Netduino, "website," http://netduino.com/.