GENERACION AUTOMÁTICA DE SOFTWARE PARA APLICACIONES DE NEGOCIOS: UN *FRAMEWORK* EMPRESARIAL

M. S. Manuel Prieto de Hoyos
Profesor e Investigador de las carreras de Informática
Facultad de Ingeniería y Arquitectura
Universidad Regiomontana
mprieto@mail.ur.mx

Resumen

Se presenta una revisión del proceso de generación manual de software, los principales factores que afectan su producción, la influencia de los estándares de programación y el uso de herramientas de apoyo en el proceso de fabricar software. Mediante la clasificación funcional de los objetos de software que intervienen en las aplicaciones empresariales, se propone una herramienta que proporciona un *framework* que permite la generación automática de aplicaciones estándares por medio de las cuales es posible generar sistemas completos.

1. Introducción.

1.1. Proceso de creación del software.

La humanidad a construido puentes y caminos por milenios, los programadores han construido programas por no mas de 70 años. Evidentemente esa experiencia se refleja en los resultados y vemos obras civiles que se desvían de su presupuesto en un 1% y proyectos de sistemas que lo hacen hasta en un 500%. Sin mencionar los que son cancelados, abandonados o simplemente suspendidos. Estas palabras o algunas parecidas las podemos encontrar en libros de texto, artículos especializados, conferencias y convenciones, son la verdad incomoda de nuestras últimas décadas.

Uno de los avances más importantes en la productividad de la fabricación de software es sin duda el uso de estándares de codificación. Sea cual fuere el lenguaje que se utilice para programar, cuando se utilizan estándares de codificación, las variables más riesgosas se ponen bajo control. Por ejemplo se puede disminuir la individualización del software y aumentar el grado de institucionalidad del mismo. La utilización de estándares es sin duda un paso firme hacia la concepción de la actividad de programación como una actividad de ingeniería en contraposición con una actividad artesanal.

Como una consecuencia del uso de estándares se produjo la detección y el uso de patrones y posteriormente aparecieron los *frameworks* . En estos nuevos enfoques el desarrollo de software se ve como una evolución, en otras palabras, el punto de partida para desarrollar un software siempre es otro software y un nuevo desarrollo nunca se inicia desde cero (Kobayashi,1999).

1.2. Patrones y *Frameworks.*

Una forma de resolver un problema es aplicando una solución que ya se ha utilizado anteriormente y que se ha documentado y guardado para uso posterior. Esa es la idea de los patrones de software. Actualmente se cuenta con catálogos de patrones de software disponibles para resolver diversos tipos de problemas.

Los patrones son una colección de objetos de software que resuelven un determinado problema en un determinado dominio (Gaertner, 1999). Es un concepto importado de la arquitectura (Alexander, et. al. 1977) y aplicado en la ingeniería de software. A diferencia de los objetos, el objetivo de un patrón no es modelar un elemento sino proporcionar una solución probada que ha sido utilizada anteriormente en contextos similares. Debido a lo anterior los patrones facilitan la arquitectura del software y permiten construir los sistemas más rápidamente. Los catálogos de patrones son colecciones completas de soluciones debidamente documentadas y listas para aplicarse.

Los patrones evolucionan cuando se aplican a otros contextos diferentes al que les dio origen (Kobayashi,1999). Un patrón puede evolucionar con intervención de un programador, o bien su evolución puede ser producida por una herramienta que permita el cambio en el patrón y que lo ajuste a un nuevo contexto, esto es lo que propone Kobayashi (2001).

El término *Framework* se refiere a una nueva estrategia de solución para requerimientos de proceso de información de las organizaciones, toma el punto de vista de los objetos no de los procesos y utiliza la teoría de objetos para soportar la solución propuesta. Un *framework* consiste en una colección de clases, entendiendo por clase la definición genérica de un objeto. Dicha colección permite la re utilización de las clases, para ello el *framework* cuenta con el soporte de interfases para su uso. Al utilizar los *frameworks* por medio de sus interfases se generan aplicaciones apegadas al contexto que resuelven problemas específicos.

Johnson y Foote (1988) definen un *framework* como “aplicaciones semi completas que pueden especializarse para producir software a la medida”. Pocos estudios se han publicado sobre la ventajas de los *frameworks* en la producción de software, sin embargo un caso de estudio analizado con rigurosidad científica se presenta en Morisio, Romano y Stamelos, I. (Septiembre, 2002) demostrando las ventajas del uso de esta estrategia.

Gaetner(1999), compara los objetos, patrones y *frameworks* utilizando las siguientes propiedades:

- Independencia del lenguaje de programación. Los *frameworks* y los objetos son dependientes en forma absoluta del lenguaje de programación.
- Dependencia del dominio. Entendiendo por dominio el contexto del problema, viéndolo así, sólo los patrones de diseño son independientes del dominio.

- *Granularidad*. Los *frameworks* muestran una alta *granularidad* que representa la capacidad de encontrar aplicaciones que se forman de solo una parte del software.
- Flexibilidad. Objetos, patrones y frameworks presentan esta propiedad.
- Cobertura. Los *frameworks* son los que presentan mejor cobertura de los problemas
- Facilidades de Integración. Los que pueden integrarse más fácilmente a otras aplicaciones son los patrones, ya que son soluciones no terminadas, mientras que los frameworks son soluciones de software.
- Re uso. Los frameworks son los que pueden re utilizarse más ampliamente brindado con ello el ahorro más representativo.

1.3. Factores que afectan la productividad de los programadores

Existen una gran cantidad de factores que pueden alterar la productividad de un programador por ejemplo: el medio ambiente, su auto concepción del trabajo, su propio temperamento y muchos otros. Aquí solo nos enfocaremos a aquellos factores que tienen una naturaleza tecnológica y que por lo tanto son controlables por la organización.

Utilizando la base de datos de proyectos realizados de *International Software Benchmarking Standards Group*, mejor conocida como, ISBSG por sus siglas en inglés, Jiang y Comstock (2007) identificaron, utilizando un estudio estadístico riguroso, los factores que afectan la productividad del software. La información que estudiaron contiene 3024 proyectos realizados de la década de los 90's al 2005. La base de datos pertenece a la edición 9 publicada en el 2005 y es reconocida como una de las más grandes y mas importantes a nivel mundial.

Los factores encontrados por Jiang y Comstock (2007) son los siguientes:

- Promedio del tamaño del equipo de trabajo.
- El lenguaje de programación y el ambiente de desarrollo integrado, IDE por sus siglas en inglés.
- La plataforma de desarrollo, una o más tecnologías involucradas, multiplataforma, etc.
- Técnicas de desarrollo, uso de estándares, objetos, patrones, frameworks.

1.4. Apoyo a los factores que afectan la productividad en la programación.

Para producir un aumento en la producción de software de los programadores es necesario apoyar los factores que afectan la productividad en la programación. Para ello se ha desarrollado una herramienta generadora de soluciones de software la cual, utilizando los principios de los *frameworks*, proporciona al programador la posibilidad de generar soluciones basadas en un conjunto de objetos que se han identificado previamente en forma estadística en los sistemas de tipo empresarial utilizados por empresas medianas y pequeñas en México (Prieto, 2007). La herramienta denominada OSMA por las siglas de su nombre: Objetos de Software para la Manufactura y la Administración, apoya los factores arriba citados de la siguiente manera:.

- Promedio del tamaño del equipo de trabajo. Esto lo hace OSMA por medio de una facilidad que permite la integración de equipos de alto desempeño, basada en la aplicación de perfiles.
- El lenguaje de programación y el ambiente de desarrollo integrado, IDE por sus siglas en inglés. El programador que usa OSMA puede elegir cualquier IDE orientado a objetos que le permita la máxima reutilización del código.
- La plataforma de desarrollo. OSMA utiliza una plataforma de base de datos multiusuario.
- Técnicas de desarrollo, uso de estándares, objetos, patrones, frameworks. OSMA automáticamente aplica estándares de desarrollo y utiliza los conceptos de objetos, patrones y frameworks, incluyéndolos en el software generado.

En las pruebas piloto la herramienta ha mostrado ahorros de tiempo de mas de un 90%.

2. El problema de la productividad de los fabricantes de software.

2.1. Incertidumbre y su efecto en los presupuestos y costos de la programación.

Cuándo los sistemas no pueden ser estimados en tiempo y costo debido a que la actividad de desarrollo es totalmente dependiente del programador que la realiza, el productor de software se ve imposibilitado para llevar a cabo sus presupuestos y salir adelante en los negocios que involucran un desarrollo de software. Para sortear este problema, lo que las empresas productoras de software han hecho en forma tradicional es asignar a su propio personal de desarrollo a sus clientes y cobrar tarifas mensuales fijas.

El problema real es la incertidumbre que el desarrollo representa. Otra solución ha sido sobre evaluar el costo para dejar un margen de protección. Lo que ha producido en el mercado un efecto de encarecimiento de los productos de software y en innumerables ocasiones el fracaso de la comercialización de los mismos.

2.2. Necesidad de una herramienta generadora de soluciones.

La demanda de servicios y de productos de software no ha disminuido, al contrario, su aumento es notable. En 1999 se estimaba que las 1000 firmas más importantes del mundo habían instalado un ERP (Economist, 1999), en 2004 el mercado de los ERPs fue de 30 billones de dólares estadounidenses a nivel mundial con tendencia a incrementarse (King, 2005). Para el 2006 se alcanzaron los 120 billones de dólares norteamericanos. En un mercado que muestra tal crecimiento, es muy importante el estudio y el apoyo a la solución de los problemas de los productores de software.

OSMA como herramienta contribuye a incrementar los factores productivos técnicos para lograr no solo la mejora de la producción y la certidumbre en los presupuestos, sino también la mejora de la calidad, habilitando a los programadores menos experimentados para que sean capaces de lograr metas destinadas anteriormente solo a unos cuantos.

3. Descripción de la Herramienta.

3.1. Estrategia General de generación del software.

La estrategia que utiliza OSMA, es conocida en el medio informático como *Framework*, reutiliza objetos de software brindando el beneficio de usar software ya existente previamente probado sin la necesidad de adaptarse 100% al mismo. En esta estrategia la reutilización parcial del software se complementa con las facilidades proporcionadas por la herramienta que agilizan la programación faltante, esto se hace en un ambiente de alta productividad contando con objetos utilitarios y de apoyo. La figura 1 muestra la relación de las tres estrategias más comunes:

- Desarrollar un sistema de información a la medida
- Utilizar un *framework* empresarial como OSMA.
- Utilizar un software construido como paquete, que en el caso de las empresas se denominan *Enterprise Resource Planning Systems* o ERPs por sus siglas en inglés

Utilizando la analogía de construir un edificio, tenemos que en la primera estrategia que es la del desarrollo de software a la medida, equivale a construir el edificio iniciando en cero y sin utilizar elementos preconstruidos o ayudas utilitarias como moldes y vistas preconstruidas haciendo todo en forma manual. En el otro extremo, la estrategia de los ERPs equivale a rentar o comprar un edificio ya hecho y adaptarlo a las necesidades específicas de los usuarios que lo van a habitar. Finalmente el uso de los frameworks equivale a construir utilizando ventanas, muros, vistas, techos, pisos y estructuras preconstruidas.

Figura 1 Alternativas de desarrollo e instalación de software. Fuente: elaboración propia

3.2. Componentes principales y elementos de la arquitectura de la solución generada.

Dentro de la herramienta OSMA se encuentra la colección de objetos que podrán usarse como base en la construcción de sistemas a manera de meta objetos. Estos son los componentes básicos que forman un sistema de información para empresas pequeñas y medianas. Poseen interfaces gráficas predefinidas y probadas que se adaptan a las diferentes necesidades de información, ya que son generados a partir de las instrucciones de SQL que generan sus bases de datos. Se ha utilizado MySQL como manejador de base de datos tanto para la herramienta como para las soluciones que genera.

Los principales objetos de OSMA son los siguientes:

- Menús
- Actualizador Unitario
- Actualizador Tabular
- Documentos
- Procesos
- Listados

La herramienta cuenta en forma adicional con las siguientes facilidades complementarias:

- Proceso completo de integración del equipo de trabajo.
- Generación de accesos seguros para los usuarios según sus perfiles.
- Integración con un reportador *Open Source* .
- Generación automática de ayudas a partir de archivos de texto.

3.3. Arquitectura y tiempo de desarrollo de las soluciones generadas.

La misma arquitectura que presenta OSMA es la que se utiliza en las soluciones generadas, debido a que una buena parte de OSMA esta hecha con la herramienta misma. Los diseños son sencillos y fáciles de operar. Sus ventanas y facilidades tienen una cantidad mínima de botones y controles operativos.

Una característica importante es que es posible adaptar los generadores a estándares especiales de las organizaciones, el costo de la solución generada aumentaría, pero por otro lado la organización contaría con generadores de soluciones adaptados a sus propios estándares.

El diseño de la solución generada es por capas, por lo que es posible cambiar el manejador de bases de datos con un mínimo esfuerzo.

El mantenimiento de un programa de la solución generada se compara con el tiempo de generar el programa de nuevo, ya que se ha encontrado que es más rápido volver a generar

un programa que cambiarlo. Los programas generados están en lenguaje Java y cambiarlos no difiere en nada de cambiar un programa hecho por un programador muy ordenado y que siempre sigue los estándares de programación.

Los tiempos de desarrollo obtenidos en la prueba piloto se muestran en la tabla 1. Se obtuvieron 9888 líneas de código generadas en 13.55 hrs. de trabajo de dos programadores participantes.

Objeto	# Objetos Desarrollados	# Líneas	Tiempo Total	Tiempo Promedio. por línea	Tiempo Promedio por Objeto
Actualizador	17	6,262	8	0.00127	0.47
Act. Tab.	2	1,109	0.95	0.00085	0.48
Menú	2	302	0.40	0.00132	0.20
Proceso	5	612	2.5	0.00400	0.50
Listado	2	309	1.35	0.00436	0.68

Tabla 1.-Resultados de la prueba Piloto de OSMA.

4. El futuro de la Herramienta.

Actualmente el desarrollo de la herramienta se encuentra en estado de pruebas masivas, medición y comprobación de resultados, búsqueda de áreas de oportunidad para detectar líneas de mejoras y de nuevas investigaciones.

Se están llevando a cabo mejoras en la portabilidad, las facilidades de instalación y la automatización de tareas, buscando siempre minimizar el trabajo y lograr una mayor productividad en la compleja tarea de construir sistemas de información. Es posible encontrar más información de Osma en la siguiente dirección de Internet: www.osmasoftware.com.

REFERENCIAS

Economist (June,1999). ERP RIP? . Economist, 00130613, 351(8125).

Gaertner, N. y Thirion , B. (1999), Working with Business Patterns & Frameworks: A Case Study for Fuzzy Logic Control. European Conference on Object-Oriented Programming (ECOOP) Workshops

Jiang, Z. y Comstock, C. (Enero 2007) The Factors Significant to Software Development Productivity. . Proceedings of World Academy of Science, Engineering and Technology Vol 21 ISSN 1307-6884Jiang, Z. y Comstock, C. (Enero 2007) The Factors Significant to Software Development

Jonson, R. y Foote, F. (Junio,1988). Designing Reusable Classes. Journal of Object Oriented Programming, 1 (5).

Johnson,R.,Foote,R. (Junio,1988). Designing Reusable Classes. Journal of Object Oriented Programming, 1 (5).

King, W. R. (2005). Ensuring ERP Implementation Success. Infromation System Management, 83.

Kobayashi, T. (Jun. 2001) Object-Oriented Modeling of Software Patterns and Support Tool, ECOOP2001 Workshop on Automating Object-Oriented Software Development Methods.Abrams, M.D. y Stein, P.G. (1973). Computer Hardware and Software. Reading,Massachusetts: Addison-Wesley Publishing Company.

Kobayashi, T. y Saeki M. (Dec. 1999) Software Development Based on Software Pattern Evolution, Proc. of Asia-Pacific Software Engineering Conference, pp.18-29

Morisio, M., Romano, D. y Stamelos, I. (September, 2002). Quality, Productivity, and Learning. IEEE Transactions on Software Engineering, 28 (9), 876-888.

Prieto, M.(2007). Identificación de Conglomerados (Clusters) para Implementar Objetos de Software, Veritas Seminario de Investigación de la Universidad Regiomontana Edición 2007, 127