Una Propuesta de Marco de Trabajo Orientado al Dominio del Procesamiento Transaccional.

Hernán Enrique Zbucki[1], Claudia Pons[1]

[1] Universidad Nacional de La Plata, Facultad de Informática
La Plata, Buenos Aires, Argentina
{hernan.zbucki}@gmail.com
{cpons}@lifia.info.unlp.edu.ar

Resumen

La ingeniería de software establece que la construcción de programas debe ser encarado de la misma forma que los ingenieros construyen otros sistemas complejos. Los sistemas de procesamiento transaccional no son la excepción. Para lidiar con algunos de los desafíos de construir estas soluciones, se desarrolló una propuesta de marco de trabajo, que propone la construcción de una base de conceptos comunes, obtenidos del análisis de soluciones preexistentes, y de experiencias del equipo que desarrolla esta propuesta. Esta recolección de factores comunes se hace de forma iterativa, y se capitaliza en elementos del *framework* que aquí se introduce. Ese marco de trabajo tendrá como objetivos aumentar la reusabilidad, disminuir los costos de mantenimiento, y fomentar la comunicación entre los desarrolladores. Para tal fin, se pretende implementar una metodología MDD (DSM) que permita facilitar el uso del marco de trabajo a otros usuarios ajenos al desarrollo de esta propuesta. La implementación específica DSM será a través de un *lenguaje de dominio específico* que concentre en *elementos de dominio*, la experiencia concentrada en el *framework*.

Palabras clave: Servidores concurrentes transaccionales, IoC, refactorización, patrones, DSM

Contexto

El proyecto de I/D/I que se presenta en este artículo está enmarcado en el ámbito de una empresa multinacional (PointPay .Inc, www.pointpay.net), que inicialmente definió las problemáticas encontradas en su proceso de construcción de sistemas de procesamiento transaccionales. A su vez, uno de los autores de este trabajo (Hernán Zbucki), alumno de la Maestría de Ingeniería de Software de la Universidad Nacional de La Plata, evaluó los requerimientos que presentó la compañía, y en conjunto con la Dra. Claudia Pons, de LIFIA, desarrollaron esta propuesta de marco de trabajo. Actualmente, ahora dentro del marco de dicha universidad, se está trabajando para formalizar el *framework* a través de un *lenguaje específico de dominio* (DSL). Mediante la implementación de una metodología DSM (*Domain Specific Modeling*), se pretende cumplir los objetivos planteados de forma tanto corporativa (requerimientos iniciales) como

académica (metodología formal para la solución de los requerimientos planteados).

Introducción

Los sistemas de procesamiento de transacciones son unos de los más usados y de los más sofisticados dentro de los sistemas de información de alta concurrencia. Particularmente un *sistema de autorización de tarjetas* debe manejar varios tipos de transacciones simultáneas, y en gran volumen por unidad de tiempo, tales como compras, autenticaciones, transferencias, devoluciones, balances, promociones, etc. [1]. Más allá del dominio de aplicación final, los puntos más delicados de un sistema transaccional (particularmente los de tarjetas de crédito/débito), son la *performance* y la *seguridad*. Respecto a la *performance*, el factor primordial es el tiempo que lleva autorizar y completar una transacción de punta a punta, mientras que en el aspecto de *seguridad*, el tema más importante es la prevención del fraude y la confidencialidad de la información sensible. [2] Ambos requerimientos están ligados: Los procesos de verificación, autenticación y criptografía, que sirven en pos de la seguridad de cualquier transacción, reducen el rendimiento y el tiempo de respuesta. Además se suma el efecto del aumento del volumen de transacciones a medida que se expande la red transaccional. Esto genera una situación donde se denota un consumo incremental de recursos computacionales, ya sea el porcentaje de uso de los procesadores de los servidores, o los tiempos de acceso a mecanismos de persistencia.

Otra problemática común en estas soluciones transaccionales es la *frecuencia de actualización del sistema* (referido a nuevas reglas de negocio, que generan cambios en los algoritmos), y la *capacidad de mantenimiento* de los componentes dentro de ellos. Los gobiernos, los bancos y otras entidades interesadas crean, y/o modifican normas y reglas de negocio recurrentemente, para satisfacer uno o varios objetivos corporativos. La motivación a estos cambios está ligado a la competencia entre compañías y bancos, de modo que son presionadas para ofrecer nuevos servicios o modificar los servicios existentes con frecuencia. [1] Estas situaciones causan constantes revisiones de los sistemas de autorización, aumentando la complejidad de mantenimiento de los mismos.

Los desarrolladores dedicados al dominio en cuestión están bajo un dilema importante ya que deben mantener un equilibrio delicado entre la calidad, la expectativa de vida del procesador transaccional y los costos de producción presupuestados. Para mantener el sistema funcionando, y a la vez, cumpliendo con las nuevas reglas de negocio o de servicio, sufren evoluciones constantes que muchas veces incurren negativamente en la calidad de la solución integral.

Motivación

La ingeniería de software establece que la construcción de programas debe ser encarado de la misma forma que los ingenieros construyen otros sistemas complejos, como puentes, edificios, barcos y aviones. La idea básica consiste en observar el sistema de software a construir como un producto complejo y a su proceso de construcción como un trabajo ingenieril. Es decir, un proceso planificado basado en *metodologías formales apoyadas por el uso de herramientas.* [3] Sin embargo, los enfoques actuales de construcción de software no son suficientes para tratar los

inconvenientes relacionados a los efectos de cambios de tecnología y a los efectos que surgen de la necesidad de cambios en los requerimientos de forma recurrente. [4]

En este trabajo se plantea entonces la construcción de una propuesta para hacer ingeniería sobre las soluciones transaccionales preexistentes, a través de la definición de un *marco de trabajo* que intente funcionar como *herramienta* para reducir los efectos adversos descritos anteriormente, inherentes al proceso de creación y mantenimiento de servidores transaccionales.

Descripción del Trabajo Realizado

El trabajo concreto realizado hasta la actualidad consta de la construcción de la primera versión totalmente funcional del *framework,* una versión de evaluación del *lenguaje específico de dominio*, y otra versión de evaluación de la herramienta de transformación automática de modelos del DSL a código fuente.

La construcción del *framework* comenzó con la recolección iterativa de factores comunes y *artefactos de software* que se repetían entre distintas soluciones analizadas. Si bien las soluciones variaban dados sus requerimientos específicos, muchos comportamientos eran comunes. Por ejemplo, una de las primeras lecciones aprendidas de este análisis fue conocer el *mecanismo de crecimiento* (a nivel constructivo) de un sistema transaccional: un procesador de transacciones crece a través del agregado de nuevas transacciones, o la sofisticación de transacciones existentes. Como contraparte, muchos sistemas analizados estaban orientados al flujo de las operaciones, sin distinguir de forma coherente los flujos operativos de cada transacción. Esto causaba que al querer modificar un comportamiento de una transacción dada, se ponía en riesgo la modificación indeseada.

Por lo que el primer paso fue detectar los *elementos repetibles* entre sistemas transaccionales, y el orden de interacción de estos elementos en la *secuencia de trabajo*. Dada la explicación del párrafo anterior, el *framework* y los sistemas generados de éste, deberían ser *orientados a la transacción*, en vez del *flujo de la operación*. De este modo, con la implementación de un patrón del tipo *Strategy*, se podrían definir las transacciones como unidades lógicas de construcción. Acompañando esta idea con la proposición de una *secuencia de trabajo genérica de procesamiento*, cada *transacción* podría resolverse de forma independiente de otras, sin afectar las eventuales modificaciones realizadas en una en particular, y por ende aumentando la confiabilidad del sistema contra nuevos cambios.

La *secuencia de trabajo genérica* tiene como objetivo que cualquier transacción, más allá de su finalidad, pueda realizarse con pasos definidos con posibilidad de sobrecarga. La definición de esta secuencia requirió mucho trabajo de revisión de sistemas preexistentes, y de trabajo iterativo de corrección, de modo de conseguir un conjunto de pasos y un orden que puedan eventualmente satisfacer a múltiples sistemas transaccionales desconocidos de antemano. Dado entonces los elementos de dominio base del *framework*, una clase de usuario de un sistema dado podría heredar del elemento *transacción,* y sobrecargar solo aquellos pasos que necesite personalizar según el requerimiento dado para la transacción real. Más aún, dada la jerarquía interna de clases dentro del marco de trabajo, varios de estos pasos pueden tener

comportamientos por defecto que faciliten la construcción de sistemas por reusabilidad de componentes. Este concepto de sobrecarga de pasos dentro de una jerarquía de clases, bajo una secuencia de ejecución pre-ordenada se formalizó implementando diferentes técnicas de *Inversión de Control* (IoC) y usando el patrón de diseño *Template Method*.

El marco de trabajo a su vez necesita definir sus elementos de dominio, en pos de la formalización a través de un DSL: Para tal fin se conceptualizaron por ejemplo, los *motores transaccionales de entrada* que permiten la recepción de requerimientos y envío de respuestas transaccionales (datos) hacia los POS (*Point of Sale)* que interactúan con el sistema. También se idearon los *motores de salida* que permiten el intercambio de tramas de datos con otros nodos externos (otros autorizadores por ejemplo). Se concretó el elemento *manejador de transacción* que contiene los métodos y delegados para crear una secuencia genérica de trabajo que pueda ser sobrecargada para la personalización del *handler* de transacción. También se definieron elementos de dominio que representan puntos de acceso a datos, cuando es necesario realizar una operación contra un archivo, o una base de datos, por ejemplo. Por cada uno de los elementos de dominio generados, se analizó la posibilidad de refactorización a patrones (siempre que fuera posible). El objetivo de esta tarea fue sacar el mayor provecho de los patrones de modo que los elementos de dominio puedan ser lo más sólidos posibles a nivel arquitectura (escalabilidad), y que puedan brindar las funcionalidades planeadas de una forma eficiente y entendible para otros desarrolladores.

Una vez definida la secuencia de trabajo, los elementos de dominio, y la jerarquía entre ellos, se propuso formalizar el conocimiento acumulado en un *lenguaje de dominio específico*. Utilizando las herramientas *DSL Tools* provistas por el IDE *Visual Studio,* se ha generado un metamodelo con la representación gráfica de todos estos elementos de dominio, con propiedades de dominio que permiten su personalización desde el mismo modelo. Se definieron un conjunto de pre-validaciones y post-validaciones por cada elemento, y las relaciones entre estos, para poder interconectarlos.

Por último se investigó y se desarrolló usando el lenguaje *T4* una herramienta de transformación que interpreta las instancias del metamodelo del DSL, y las transforma a código fuente. A futuro, estamos considerando poder transformar los modelos generados por este DSL en otros modelos, como por ejemplo diagramas de clase o de secuencia.

Líneas de Investigación, Desarrollo e Innovación

Para este proyecto de I/D/I, se investigaron los siguientes tópicos:

- Metodologías DSM, dentro de la rama MDD.
- Construcción de DSL's
- Lenguajes para la construcción de herramientas que permitan la transformación de modelos a código fuente.
- *Refactorización a Patrones de Diseño*, que permitan optimizar los elementos del marco de trabajo, en el ámbito de sistemas transaccionales.
- Técnicas de *Inversión de Control*.

Resultados y Objetivos

PointPay Inc., para la cual se construyó la primera versión del *framework*, valoró la definición de un marco de trabajo que se enfoque a la reusabilidad de la mayor cantidad de componentes posibles, la reducción de costos de mantenimiento, y la convergencia de criterios de diseño entre desarrolladores.

Con estos tres objetivos en vista, se realizó un trabajo retrospectivo, que requirió la revisión y re-desarrollo iterativo de componentes preexistentes, el agregado de valor ingenieril a cada iteración de re-trabajo, la recolección de factores comunes del dominio, la capitalización de esos factores en elementos base del marco de trabajo, la refactorización a patrones conocidos, y la re-educación de los integrantes de los equipos de desarrollo, al trabajo cooperativo entre subsidiarias, a través del uso del marco de trabajo bajo definición. Este trabajo retrospectivo, que comenzó a mediados de Mayo de 2012, generó la primera versión de *Transaction Kernel* (nombre del marco de trabajo). Actualmente esta revisión se sigue realizando de forma trimestral al día de la fecha. A futuro se formalizará el *framework* en un *lenguaje de dominio específico* y un conjunto de herramientas que permitan la transformación automática de modelos del DSL a código fuente, u otros modelos.

Formación de Recursos Humanos

El equipo está conformado por el Ing. Hernán Zbucki, la Dra. Claudia Pons, y un grupo de alumnos de grado de la Facultad de Informática de la UNLP. Actualmente este trabajo forma parte de la Tesis de Maestría de Ing. de Software del Ing. Zbucki (en desarrollo), y de dos Tesinas de Grado, también en curso.

Referencias

1. Ming, W. Y.: Multi-threading technique for authorization of credit card system using .NET and JAVA, Kuala Lumpur, 2007.
2. Kang, K., Lee, J., Kim, B., Kim, M., Seo, C., Yu, S.: Re-engineering a credit card authorization system for maintainability and reusability of components–a case study. In: Reuse of Off-the-Shelf Components, pp. 156--169. Springer Berlin Heidelberg (2006)
3. Pons, C., Giandini, R., Perez, G.: Desarrollo de software dirigido por modelos. Conceptos teóricos y su aplicación práctica. EDULP, La Plata, Argentina (2010)
4. Singh, Y., Sood, M.: Models and Transformations in MDA. In: First International Conference on Computational Intelligence, Communication Systems and Networks. (2009)