Estado Actual y Mejora de la Utilización del Conocimiento Práctico en Diseño OO en la Industria Software

Javier Garzás [1] y Mario Piattini [2]
[1] Consultor Senior de ALTRAN SDB.
Madrid - España
jgarzas@altransdb.com / javier31175@terra.es
[2] Grupo Alarcos
Escuela Superior de Informática, Universidad de Castilla-La Mancha
Ronda de Calatrava, s/n. 13071
Ciudad Real – España

RESUMEN

Aparte del concepto de patrón, en el diseño de micro arquitecturas OO existen otros muchos núcleos de conocimiento, como los principios, reglas, heurísticas, etc., menos conocidos y pobremente caracterizados. En este trabajo presentamos un estudio estadístico sobre la difusión del conocimiento en diseño OO en la industria software y un estudio empírico sobre la validez y beneficios de disponer de una caracterización del mismo.

Palabras clave

Orientación a Objetos (OO), Micro Arquitecturas, Diseño, Principios, Patrones, Malos Olores, Heurísticas y Ontología

1 INTRODUCCIÓN

"It is time to stop hiding the enormous depth and breadth of our field"

(Denning P. J., The Profession of IT, Communications of the ACM, Noviembre 2003)

Hace ya más de 20 años desde que [1] comentó como "an expert in a field must know about 50,000 chunks of information, where a chunk is any cluster of knowledge sufficiently familiar that it can be remembered rather than derived", añadiendo como en áreas maduras esto suele llevar 10 años. La experiencia en un área es lo que distingue a los expertos, y su captura, comunicación y asimilación suponen un desafío. Pero aún hoy se reafirma como desde el nacimiento de la ingeniería del software sigue pesando la falta de estructuración y clasificación del conocimiento y experiencia en esta disciplina [2] [3].

Obviamente, todos reconocemos que en los últimos años se ha avanzado mucho en este sentido (claro ejemplo es el proyecto SWEBOK [4]). Así, en la actualidad disponemos de numerosos "chunk" de información definidos sobre la ingeniería del software, conocimiento de una u otra forma, y que está formado, entre otros, por estándares, metodologías, métricas, técnicas, lenguajes, conocimiento relacionado con procesos, conceptos, etc.

Pero no todo el conocimiento en ingeniería del software es estudiado, es tan accesible o está difundido de igual forma. Podemos sorprendernos a este respecto si realizamos el siguiente ejercicio... ¿cuál es y dónde está el enorme conocimiento práctico en Diseño de micro arquitecturas[1] Orientadas a Objetos (objetivo de este trabajo), basado en la experiencia acumulada en el desarrollo de sistemas software durante todos estos años y aplicable en la mayor parte de proyectos? Conocimiento que se ha acumulado con la experiencia de trabajar frente a las que [5] denomina complejidades inherentes del software, y que también este denomina "Esencial" y que está diferenciado del "Accidental" (más tecnológico y pasajero, como el conocimiento en un determinado lenguaje) que suele estar sujeto a "los tres años de vida" que menciona [2]. Conocimiento que está dentro del "generalmente aceptado" o "prácticas aplicables a la mayoría de los proyectos la mayoría de las veces, y de las que hay un amplio consenso sobre su valor y utilidad" [4].

[1] No existe definición de micro arquitectura ni en el estándar IEEE 610.12 ni en el proyecto SWEBOK. Cuando en este trabajo se hace referencia al concepto "micro – arquitectura OO" se refiere a la arquitectura que se centra en los objetos y clases, su estructuración y la relación existente entre estos, sin entrar en aspectos internos de sus métodos, más relativos a la codificación de los mismos.

Podemos pensar que este Conocimiento Esencial del que hablamos puede estar en parte en las librerías, componentes, marcos de trabajo (frameworks), en forma de código, etc., pero estos no son mecanismos para obtener diseños a lo largo del ciclo de vida, son más "Conocimiento Accidental" y muchas veces producto del "Conocimiento Esencial" basado en la Experiencia, aplicable para generar a estos y muchos otros; el conocimiento práctico en Diseño de micro arquitecturas Orientadas a Objetos es la pieza fundamental de la reutilización de experiencias de diseño probadas durante años [6].

PRINCIPIOS
The Dependency Inversion Principle (DIP) "Depend upon Abstractions. Do not depend upon concretions" [7]
HEURÍSTICAS
"If two or more classes only share common interface (i.e. messages, not methods), then they should inherit from a common base class only if they will be used polymorphically." [8]
MEJORES PRÁCTICAS
"See objects as bundles of behavior, not bundles of data." [9]
MALOS OLORES
Refused bequest Subclasses that do not use what they inherit [10]

Tabla 1. Heterogeneidad entre núcleos de conocimiento.

En este punto podemos destacar uno de los principales núcleos de Conocimiento Esencial basado en la Experiencia sobre Diseño de Micro Arquitecturas OO: los patrones de diseño. Pero estos son sólo una parte, la visible del iceberg. Simplifiquemos y supongamos que queremos especializamos como ingenieros software en "Diseño Orientado a Objetos". Utilizando, por ejemplo, proyectos como SWEBOK podemos saber de manera más exacta y unificada qué es el área de Diseño, cómo se subdivide, sus principales referencias bibliográficas, etc., y hacernos con "razonable facilidad" con un buen conocimiento teórico. Si ahora continuamos evolucionando nos encontramos con la necesidad de estudiar la experiencia práctica de otros expertos en el área, y entonces nos encontramos con el concepto de patrón.

Pero tras estudiar las referencias de patrones en esta área (y salvando la complejidad de encontrar otros catálogos y de que cada uno tenga una estructura propia) intuimos que aún faltan cosas. Y en efecto, para un tema tan estudiado como el de los patrones aún no hemos logrado un método eficiente para que se sepan utilizar de manera adecuada, como, por ejemplo, se puso de manifiesto en el taller de trabajo (Workshop) "Beyond Design: Patterns (mis)used", especialmente dedicado al tema, y al que pudimos asistir como invitados en la OOPSLA 2001, donde autores como [11] manifestaron como "We got more and more aware that a good description of the proposed solution is necessary, but useless for the reader if the problem and the forces that drive the relationship between problem and solution are not covered properly".

Y no sólo eso, con relación a lo anterior existen desde hace tiempo otros muchos elementos de calidad, cualidades estructurales y características que un diseño debe presentar para soportar extensibilidad y reutilización [12]. Así, si continuamos indagando, buscando otros elementos que pudieran contener otro tipo de conocimiento sobre el diseño de micro arquitecturas aparte del que describen los patrones entraremos entonces en un "caos sin orden", en el que muchas de nuestras respuestas aparecerán entre: Principios, Heurísticas, Lecciones Aprendidas, Defectos, Prácticas, Experiencias, Malos Olores (Bad Smells), Refactorizaciones, Antipatrones, Patrones, etc. En la Tabla 1 podemos ver un ejemplo de la heterogeneidad a la hora de encontrar núcleos de conocimiento, su distinta terminología y tratamiento.

Principios, refactorizaciones, mejores prácticas y otros similares son una evolución en la forma de aplicar, estructurar, etc., el conocimiento en diseño, por lo que el valorar su incorporación en los procesos de ingeniería del software supone un paso en nuestrog perfeccionamiento y evolución.

2 APROXIMACIÓN ESTADÍSTICA AL ESTADO DEL CONOCIMIENTO EN DISEÑO OO EN LOS PROFESIONALES

Con el objetivo de observar el estado en que se encuentra el conocimiento práctico acumulado en el diseño de micro arquitecturas OO (patrones, principios, malos olores, etc.) en los profesionales de empresa realizamos una aproximación estadística sobre lo difundido del mismo[2].

[2] Destacar que los estudios estadísticos de este trabajo han sido organizados con independencia de las organizaciones de las que dependen los autores de este artículo.

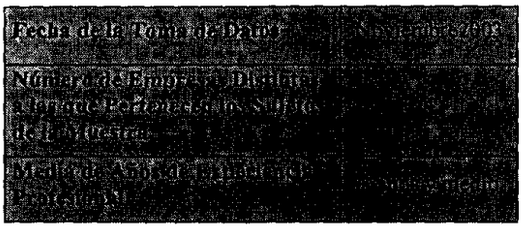

Tabla 2. Datos de la estadística sobre el conocimiento en diseño OO

De entre un numeroso grupo de profesionales hemos escogido minuciosamente una muestra de 15 individuos (atendiendo a sus años de experiencia en diseño, conocimientos en OO, etc.), de distintas empresas y con una media de 3 años y medio de experiencia (ver tabla 2), a los que les hicimos la siguiente pregunta:

"Si conoces algún(a) PATRON, PRINCIPIO, HEURÍSTICA, REFACTORIZACIÓN, BAD SMELL o similar de diseño OO escribe su nombre poniendo a su lado y entre paréntesis una T, si lo conoces sólo a nivel teórico, o P, si lo conoces a nivel práctico (lo has aplicado)"

El resumen de los resultados obtenidos es el que puede observarse en la gráfica de la figura 1. En esta se muestra el conocimiento teórico y práctico de patrones y reglas (concepto que explicamos en la sección 3 y bajo el que agrupamos a malos olores, principios, heurísticas, etc.). Se observa cómo existe un gran desconocimiento, prácticamente nulo, de las reglas por parte de los profesionales, tanto a nivel teórico como práctico, donde ambas gráficas coinciden. En el caso de los patrones la cosa mejora algo, pero, curiosamente, siendo este elemento mucho más popular, los datos frente a las reglas tampoco son muy diferentes.

No podemos afirmar, por el tamaño de la muestra, que la estadística sea del todo concluyente, pero si creemos que refleja el estado general en el que la experiencia práctica acumulada en el diseño de micro arquitecturas OO está asentada y difundida entre los profesionales, resultado que coincide con la sensación que encontramos a la hora de desarrollar proyectos de consultoría.

Claramente, el camino hacia la incorporación de estas tecnologías encuentra aún serios problemas. Las diferencias entre estos elementos no están claras, muchos son el mismo concepto denominado de distinta forma, contienen distinto tipo de conocimiento desde la experiencia, son conceptos difusos y no tienen una interpretación unificada, tampoco existe un método sistemático que los organice y que sea capaz de formar una guía de aplicación del mismo, de comparar sus componentes y clasificarlos.

Así, si bien hemos podido avanzar mucho en acumular conocimiento esencial sobre el diseño de micro arquitecturas basándonos en la experiencia acumulada durante años, hemos avanzado menos en la parte operativa, en su explotación y en su clasificación. Por lo general, las organizaciones fallan en aprender de su propia experiencia [3] y aún hoy, con el conocimiento adquirido en el diseño de sistemas OO, y conscientes de la problemática que puede suponer su mala realización, el problema con el que se enfrenta el diseñador es articular todo este conocimiento y aplicarlo de manera ordenada y eficiente, de forma que le pueda resultar verdaderamente útil.

3 CARACTERIZACIÓN DEL CONOCIMIENTO EN DISEÑO DE MICRO ARQUITECTURAS OO

Antes de establecer una caracterización y una base metodológica sobre el conocimiento en el diseño de micro arquitecturas OO es necesario precisar qué es cada uno de los elementos que componen este conocimiento y en qué se diferencian.

Ya hemos apuntado como a parte del conocido concepto de "patrón" existen muchos otros conceptos que agrupan un importante conocimiento sobre el diseño de micro arquitecturas OO, y entre estos se encuentran las "heurísticas", "mejores prácticas", "lecciones aprendidas", "bad smells", "refactorizaciones", etc., los cuales no pueden obviarse si pretendemos obtener el mayor beneficio de nuestra experiencia en diseño OO.

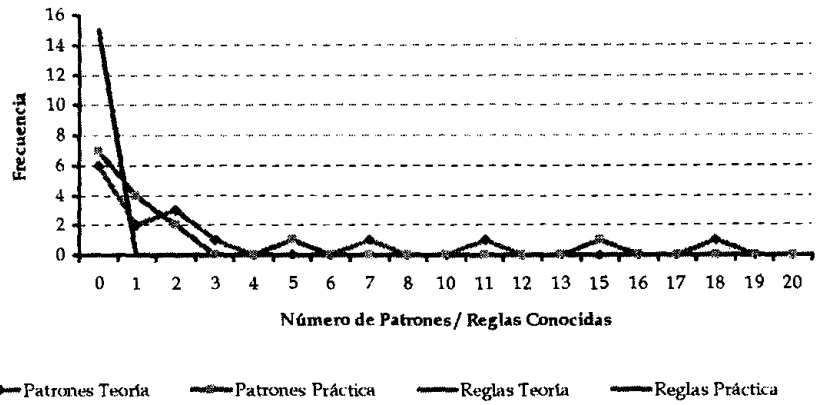

Fig. 1. Estadística sobre el conocimiento en diseño OO

El problema que encontramos es que salvando al concepto de Patrón, el resto de elementos que podríamos asociar al conocimiento en diseño de micro arquitecturas OO están pobremente clasificados, diferenciados, accesibles, etc. Estos problemas dificultan su utilización, comprensión y ampliación. Además, muy pocos autores han tratado esta problemática, y apenas hemos podido encontrar algún estudio, como el de [13], que apunta como "el conocimiento acumulado por la comunidad OO se divide en dos categorías: los principios de alto nivel, ampliamente aceptados, y los patrones, que documentan distintos tipos de conocimiento de diseño".

Con el objetivo de lograr una clasificación unificada del conocimiento, hemos observado como, si bien existen numerosos términos relacionados con lo que podríamos denominar conocimiento esencial acumulado de la experiencia en el diseño de micro arquitecturas, podemos hacer una primera agrupación en dos conjuntos:

- Por un lado tenemos los denominados Principios, Heurísticas, Patrones, Malos Olores (Bad smell), Lecciones Aprendidas, Mejores Prácticas, Buenos hábitos y similares, todos de carácter declarativo.

- Por otro, podemos encontrar otros conceptos como la refactorización. Este expresa una cantidad de conocimiento acumulado sobre cuál es la mejor manera de realizar cambios en el software, un método de realizar estos, transformación parametrizada de un programa preservando su funcionalidad. Así supone una operación o proceso, es decir, conocimiento operativo. Hay que resaltar que nos referimos a refactorizaciones en el ámbito de diseño (design refactoring), no de código.

Podemos hacer una primera clasificación y dividir el conocimiento esencial basado en la experiencia en el diseño de micro arquitecturas software OO en dos bloques: el conocimiento declarativo y operativo. El primero nos dice qué hacer y el segundo cómo, ambos según la experiencia.

Centrándonos exclusivamente en el conocimiento declarativo y sin tomar en cuenta a los patrones, quedándonos sólo con Principios, Heurísticas, Bad Smells, etc., hemos observado que en esencia estos siempre tienen la misma estructura común, y aunque diferentes autores los han denominado de distinta forma y existe una amplia heterogeneidad entre las formas de describir estos componentes del conocimiento, no existe diferencia sustancial entre los mismos: todos atienden a la estructura y forma de una Regla, en la que exponen una condición para la que de cumplirse se ofrece una recomendación, destacar la "recomendación" de la Regla, que no es una solución como la del Patrón.

Por otro lado, también hemos observado que existe una importante relación entre reglas y patrones [14], de tal forma que en muchas ocasiones los patrones son directamente implicados por las reglas pudiendo detallarse esta relación.

Con todo lo anterior hemos desarrollado una ontología del conocimiento, un método de aplicación,

un conjunto de métricas [15] y un catálogo unificado de reglas de conocimiento.

4 ESTUDIO EMPÍRICO DEL BENEFICIO DE LA CARACTERIZACIÓN DEL CONOCIMIENTO EN DISEÑO OO

Con el objetivo de comprobar los beneficios de poseer una caracterización del conocimiento en diseño de micro arquitecturas OO hemos desarrollado un estudio empírico, que resumimos a continuación. Este estudio toma como base los trabajos de [16] [17].

Para el experimento utilizamos tanto casos de estudio como un experimento para comprobar los resultados; ya que usar un ejercicio real sería excesivamente complejo, utilizamos un caso simplificado que muestra una micro arquitectura de diseño OO.

La hipótesis que pretendemos contrastar es que *"utilizar el conocimiento de manera caracterizada junto con un catálogo unificado de reglas de conocimiento hace que los profesionales obtengan micro arquitecturas de más calidad"*.

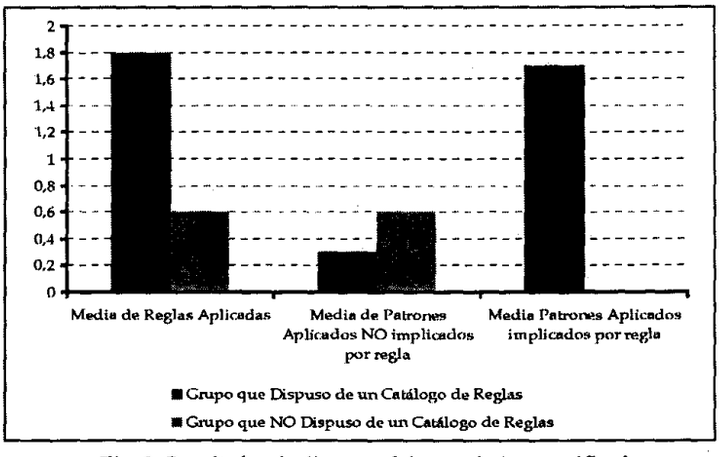

Fig. 2. Resultados de disponer del conocimiento unificado

Los sujetos del experimento son profesionales en activo, que dividimos en dos grupos que equilibramos en función de los años de experiencia de cada persona. Tomando un mismo caso de estudio, a los dos grupos les proporcionamos un resumen del catálogo de patrones de [18] y sólo a uno de ellos el catálogo de reglas unificado y una breve formación, de aproximadamente 25 minutos, sobre cómo utilizarlo.

Con todo lo anterior pedimos a los sujetos que en máximo de una hora realicen mejoras a la micro arquitectura propuesta. Esta micro arquitectura incumple 6 reglas básicas de diseño y puede mejorarse con la aplicación de 6 patrones. Dos de estos 6 patrones, si bien pueden aplicarse directamente, son recomendados al aplicar 2 de las reglas, y en el catálogo de reglas estas los recomiendan de manera explícita.

Una vez ejecutado el estudio obtenemos los resultados que se muestran en la figura 2. Respecto a la aplicación de reglas el grupo que disponía de un catálogo de reglas unificado obtuvo una media de 1,8 aplicaciones frente a 0,6 del grupo que no disponía del catálogo de reglas. En cuanto a la aplicación de patrones no implicados por regla (en los que ambos grupos estaban en igualdad de condiciones), el grupo con catálogo de reglas aplicó una media de 0,3 frente a 0,6 del otro grupo, destacar aquí el grupo sin catálogo obtiene mejores resultados. Por último, a la hora de detectar patrones que son implicados por reglas el grupo formado en reglas obtiene una media de 1,7 aplicaciones frente a 0 del otro grupo, donde se destaca una importante diferencia.

5. CONCLUSIONES

Los expertos han usado ideas de eficacia probada desde siempre. Conceptos como los principios, malos

olores, heurísticas, etc., resuelven problemas, capturan soluciones y no simplemente principios abstractos o estrategias teóricas, además de ser conceptos probados durante años. Pero aún hoy, se destaca la poca madurez que rodea a este tipo de técnicas en lo que concierne a su aplicación metodológica, la propia clasificación de estas, su nivel de aplicabilidad, la dificultad de saber cuándo y dónde aplicarlas, metodologías, etc.

Para facilitar esta tarea, hemos desarrollado una ontología, un método en base al conocimiento práctico acumulado y un catálogo unificado de reglas de conocimiento, con el objetivo implícito de no olvidar que *"a branch of engineering must take several basic steps in order to become an established profession, highlighting understanding the nature of knowledge. We as a discipline must ask how software engineers might acquire this knowledge."* [19].

6. AGRADECIMIENTOS

Esta investigación parte del proyecto MAS (TIC 2003-02737-C02-02).

BIBLIOGRAFÍA Y REFERENCIAS

[1] Redwine S.T. (1984). DOD-Related Software Technology Requirements, Practices, and Prospects for the Future. Tech. Report P-1788, Inst. Defense Analyses, Alexandria.

[2] McConnell S. (2003). Professional Software Development. Addison-Wesley.

[3] Backlund P. (2003). Knowledge Transfer in Information Systems Engineering. Tesis Doctoral, Department of Computer and Systems Sciences, Stockholm University/Royal Institute of Technology.

[4] Abran A., Bourque P., Dupuis R. y Moore J. (2001). Guide to the Software Engineering Body of Knowledge - SWEBOK, IEEE Press

[5] Brooks F. P. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. Computer, 20 (4), pp. 10-19.

[6] Ji K. y Chen S. (2001). Design Pattern Application - a component-based model for applying design patterns in software development. Informatica. Vol 25, No 4.

[7] Martin R. C. (1990). [illegible] Report. Enero – Noviembre.

[8] Riel A. J. (1996). Object-Oriented Design Heuristics. Addison-Wesley.

[9] Venners B. (2004). Interface Design Best Practices in Object-Oriented API Design in Java. www.artima.com/interfacedesign/contents.html

[10] Fowler M. (2000). Refactoring improving the design of existing code. Addison Wesley

[11] Schwanninger C. (2001). Patterns as Problems Indicators. OOPSLA 2001, Workshop Beyond Design: Patterns (mis)used.

[12] Opdyke W. (2000). Refactoring, reuse and reality. En Fowler M. (Ed.), Refactoring. Improving The Existing Code. Addison– Wesley.

[13] Priestley M. (2001). Practical Object Oriented Design with UML. McGrawHill

[14] Garzás J. y Piattini M. (2001). Principios y Patrones en el Diseño Orientado a Objetos. JISBD 2001, VI Jornadas de Ingeniería del Software y Bases de Datos, pp. 285-295.

[15] Garzás J. y Piattini M. (2002). Object Oriented Design Knowledge: Ontology and Measurement of Impact. En Bellahsene Z., Patel D. y Rolland C. (Eds.): Object-Oriented. Information Systems, OOIS 2002, Proceedings. Lecture Notes in Computer Science. Springer

[16] Wohlin C. (1999). Experimentation in Software Engineering – An Introduction published. Kluwer Academic Publishers.

[17] Prechelt L., Unger B., Philippsen M. y Tichy W. (1997). Two controlled Experiments Assessing the Usefulness of Design Patterns Information During Program Maintenance. Empirical Software Engineering.

[18] Gamma E., Helm R., Johnson R. y Vlissides J. (1995). Design patterns: Elements of Reusable Object Oriented Software. Addison-Wesley.

[19] Shaw M. (1990). Prospects for an Engineering Discipline of Software. IEEE Software 7(6), pp.15-24.