

# Sobre el grado de acuerdo entre evaluadores en la detección de *Design Smells*

Khalid AlKharabsheh<sup>\*1</sup>, Yania Crespo<sup>2</sup>, Esperanza Manso<sup>2</sup>, and José Ángel Taboada<sup>1</sup>

<sup>1</sup>{khalid, joseangel.taboada}@usc.es, CíTIUS

Centro Singular de Investigación en Tecnologías de la Información

Universidad de Santiago de Compostela, Santiago de Compostela 15782

<sup>2</sup>{yania, manso}@infor.uva.es, Dpto. Informática, Universidad de Valladolid

Escuela de Ingeniería Informática. Campus Miguel Delibes, Valladolid 47011

**Resumen** La detección automática de *Design Smells* ha evolucionado a la par que las herramientas automáticas de *refactoring*. Sin embargo se constata que su grado de adopción por la industria de desarrollo del software no ha sido comparable con la forma en la que se han adoptado las herramientas de *refactoring*. En este trabajo partimos de la suposición de que la diferencia reside en la objetividad y pragmatismo de una operación de *refactoring* comparada con el grado de subjetividad que suponemos en la definición e identificación de *Design Smells*. Para estudiar este problema se diseñó una encuesta difundida vía correo electrónico en la que se obtuvieron 92 respuestas de personas tanto de la academia como de la industria acerca de la presencia de *Design Smells* en 5 clases de un proyecto de código abierto. El estudio se ha realizado centrándose en la detección de dos tipos de *Design Smells*: *God Class* y *Feature Envy*. Se ha realizado un análisis en el que se valora el grado de acuerdo en la identificación de estos *Design Smells* entre los encuestados mediante el estadístico *Kappa-Fleiss*. En el trabajo se analizan la interrelación entre diferentes factores como la experiencia del evaluador, su contexto de trabajo, etc. Los resultados obtenidos muestran que no existe acuerdo en general o que es muy pobre en los casos que sí existe algún acuerdo. Estos resultados sostienen que, por una parte, falta formación en *Design Smells* y, por otra parte, hay una componente subjetiva que hace a distintos sujetos evaluar de forma distinta la presencia o no del *Design Smell*.

**Keywords:** Detección de *Design Smells*, encuesta, acuerdo entre evaluadores, *Kappa-Fleiss*, calidad, evolución, mantenimiento

## 1. Introducción

Un *Design Smell* es un problema encontrado en la estructura del software (código, diseño) que no produce errores de compilación o de ejecución, pero afecta negativamente los factores de calidad del software. *Design Smell* es un término unificador que agrupa en un único concepto a *Code Smells*, *Bad Smells*,

\* Khalid AlKharabsheh agradece la financiación del programa Erasmus Mundus

*Disharmonies*, *Antipatterns*, *Design flaws*, *Technical debt*, entre otros términos conocidos en el estado del arte.

La detección de *Design Smells* ha ido experimentado un auge en resultados de investigación publicados. La primera definición de *Design Smells* data del año 2000. La primera herramienta se reporta en 2002. A partir de 2004 comienza un crecimiento continuado, proliferando la aparición de nuevas herramientas para la detección automática de *Design Smells*. Particularmente a partir de 2009-2010 se produce un incremento notable en la actividad en este ámbito.

Nos encontramos actualmente con todo tipo de herramientas para detectar *Design Smells*: herramientas dedicadas, integradas como plug-ins en entornos de desarrollo, aplicando técnicas diferentes, para varios lenguajes de programación, etc.. Sin embargo, a pesar de este auge, no se ha experimentado una adopción de estas herramientas por parte de la industria. La detección automática de *Design Smells* ha evolucionado a la par que las herramientas automáticas de *refactoring*. Teniendo en cuenta que la presencia de *Design Smells* sirve como base para la detección de oportunidades de *textitrefactoring*, es llamativo que su grado de adopción por parte de los desarrolladores en la producción y mantenimiento del software no sea comparable con la forma en la que se han adoptado las herramientas de *refactoring*. En este trabajo partimos de la suposición de que la diferencia reside en la objetividad y pragmatismo de una operación de *refactoring* comparada con el grado de subjetividad que suponemos en la definición e identificación de *Design Smells*.

Partiendo de estudios comparativos sobre herramientas de detección de *Design Smells*, se aprecia que no hay buen acuerdo entre ellas a la hora de detectar la presencia de un *Design Smell*. En trabajos previos [5], nuestro grupo ha explorado la cuestión de modelar la subjetividad en la detección de *Design Smells*, introduciendo información semántica sobre la intención de diseño del desarrollador, así como teniendo en cuenta históricos de datos de las organizaciones y la retroalimentación de falsos negativos y falsos positivos detectados por expertos. Estas ideas relativas a lo subjetivo de la detección de *Design Smells* nos conducen a la necesidad de realizar un estudio para comprobar cómo coinciden o no las personas, desarrolladores e investigadores al decidir la presencia de un *Design Smell* en el software. En consecuencia, se elaboró el experimento que aquí se presenta.

Para estudiar este problema se diseñó una encuesta difundida vía correo electrónico en la que se obtuvieron 92 respuestas de personas tanto de la academia como de la industria acerca de la presencia de *Design Smells* en 5 clases de un proyecto de código abierto. El estudio se ha realizado centrándose en la detección de dos tipos de *Design Smells*: *God Class* y *Feature Envy*. La selección se basa en que son dos de los *Design Smells* más populares en la literatura y en que además se trata de *Design Smells* de diferente naturaleza en cuanto al ámbito y al efecto en el software. Se ha realizado un análisis en el que se valora el grado de acuerdo en la identificación de *Design Smells* entre los encuestados mediante el estadístico *Kappa-Fleiss* [3]. En el trabajo se analizan la interrelación entre diferentes factores como la experiencia del evaluador, su contexto de trabajo, etc. Los resultados obtenidos muestran que no existe acuerdo en general y que es muy pobre en los casos en los que sí existe algún acuerdo. A partir de los resultados obtenidos se concluye con recomendaciones a tener en cuenta en

las nuevas tendencias para la detección automática de *Design Smells* que pueden influir positivamente en la adopción por la industria de técnicas y herramientas para la detección de *Design Smells*.

El resto del artículo se organiza de la siguiente forma. En la Sección 2 se describe el marco conceptual para este trabajo. Se tratan cuestiones relativas a la detección de los defectos de diseño que nos ocupan. En la Sección 3 se detallan los objetivos del trabajo y las preguntas de investigación que nos planteamos responder. Se describe el diseño de la encuesta realizada de forma que permita obtener datos para responder a las preguntas de investigación. La Sección 4 presenta los estudios realizados a partir de las respuestas de los evaluadores, los resultados obtenidos y el análisis de los mismos. En la Sección 5 se describen brevemente algunos trabajos relacionados mientras que la Sección 6 presenta un resumen de las conclusiones obtenidas y las líneas de trabajo a partir del mismo.

## 2. Detección de los *Design Smells God Class* y *Feature Envy*

El estudio se ha realizado centrándose en la detección de dos tipos de *Design Smells*, *God Class* y *Feature Envy*. Se trata de dos tipos diferentes. *God Class* es un *Design Smell* intra clase, sólo se necesita observar el código de dicha clase para detectarlo. Por su parte *Feature Envy* es un *Design Smell* inter clase, para detectarlo es relevante observar la interacción de dicha clase con otras clases relacionadas.

En los buenos diseños orientados a objetos la lógica del sistema está distribuida uniformemente entre las clases de los niveles superiores. En [9] se define una *God Class* como un objeto que controla demasiados objetos en el sistema y que ha crecido más allá de toda lógica para convertirse en la clase que lo hace todo. En una *God Class* hay demasiadas variables de instancia y demasiado código. Donde hay demasiado código hay peligro de que surja código duplicado. A veces se corresponde con una mala aplicación del patrón de diseño Mediador. Este problema de diseño puede ser parcialmente asimilado con el defecto *Large Class* definido por Fowler [4]. También es conocido como el antipatrón “*The Blob*” en [2].

*Feature Envy* se recoge en el catálogo de Fowler [4] y se clasifica por Wake [11] en la categoría que denomina “entre clases” y la subcategoría “de responsabilidad”. Un método tiene el defecto *Feature Envy* si parece estar más preocupado en manipular datos de otras clases que de la suya propia. Está relacionado con la responsabilidad de la clase, ya que contiene métodos que deberían estar en otra clase. Como excepciones Fowler indica que existen varios patrones de diseño que rompen esta regla, como el Visitante y el Estrategia.

Se trata entonces de dos *Design Smells* de ámbito diferente. *God Class* es de ámbito de clase. Para detectarlo hay que revisar la clase como un todo. *Feature Envy* por su parte es de ámbito de método, para detectarlo basta con revisar en el interior de un método. De acuerdo con Tiberghien et al. [10] en su clasificación de *Design Smells*, *God Class* pertenece al grupo de los “*Bloaters*” (hinchadores) mientras que *Feature Envy* pertenece al grupo de los “*Couplers*” (acopladores).

### 3. Objetivos y preguntas de investigación

El estudio previo que hemos realizado, sobre el grado de acuerdo entre herramientas al detectar *Design Smell* en una colección de clases Java [1], nos ha dado pie para profundizar en el escenario de detección de *Design Smell* por personas (evaluadores), estudiando el papel que juega la subjetividad en dicha detección.

Utilizando el patrón **GQM**, ampliamente extendido en la investigación en ingeniería del software para definir los objetivos [12], el objetivo de este estudio cuasi experimental se define como:

**Analizar** Una colección de clases  
**Con el propósito de** evaluarlas  
**Con respecto al** grado de acuerdo entre evaluadores para detectar los *Design Smells God Class y Feature Envy*  
**Desde el punto de vista de** profesores, investigadores y desarrolladores de software  
**En el contexto** académico de los investigadores que realizan el estudio

De este objetivo general derivamos la siguiente hipótesis de trabajo, y cuestiones a contestar.

**Hipótesis 1** *No existe acuerdo entre los evaluadores en la detección de los Design Smells God Class y Feature Envy*

*Esta hipótesis general se desglosa en las siguientes hipótesis secundarias:*

**Hipótesis 1.a** *No existe acuerdo entre los evaluadores humanos en la detección de Design Smells.*

**Hipótesis 1.b** *No existe acuerdo entre evaluadores humanos y herramientas en la detección de Design Smells.*

Queremos conocer qué factores pueden afectar el acuerdo o el desacuerdo en la detección de *Design Smells*, qué interrelación puede existir entre estos factores y qué impacto tiene esto en detección. Pensamos que puede haber factores que afecten de alguna forma a la detección de *Design Smells* cuando se trata de evaluadores humanos, tales como: el grado de experiencia de los desarrolladores, sus antecedentes (en términos de formación, conocimientos y actividades desarrolladas), el contexto de trabajo, incluso el área geográfica en la que se han formado y/o trabajan.

Para estudiar este problema se diseñó una encuesta online, cuyo objetivo principal era permitir la detección de los *Design Smells God Class y Feature Envy* por sujetos (evaluadores humanos). Las preguntas de investigación que nos planteamos, y que han dirigido tanto el diseño de la encuesta como la explotación de los resultados, son las siguientes:

**RQ1** ¿Existe acuerdo entre los evaluadores humanos al detectar *Design Smells*?

**RQ2** ¿Existe acuerdo entre los evaluadores humanos y las herramientas de detección de *Design Smells*?

**RQ3** ¿Qué herramientas coincide más con los evaluadores humanos al detectar *Design Smells*?

- RQ4** ¿Cómo afecta el grado de experiencia en la detección de *Design Smells*?
- RQ4a** ¿El grado de acuerdo entre evaluadores humanos es mayor en el grupo de evaluadores con mayor experiencia?
- RQ4b** ¿El grado de acuerdo con las herramientas de detección es mayor cuando el evaluador humano tiene mayor experiencia?
- RQ5** ¿Cómo afectan los antecedentes (en términos de formación, experiencia y conocimientos) y el contexto de los evaluadores en la detección de *Design Smells*?
- RQ5a** ¿Afecta el contexto de trabajo? ¿el área geográfica donde se forma o desarrolla la persona? ¿si se trata de un contexto académico o de la industria?
- RQ5b** ¿Afectan los antecedentes del evaluador? ¿experiencia en programación orientada a objetos, en lecturas o revisiones de código? ¿nivel de conocimiento de *Design Smells*?

### 3.1. Diseño del experimento

La encuesta y las clases a evaluar como candidatas a presentar *Design Smells*, se han elegido para que los sujetos no necesiten demasiado tiempo para completar las tareas. Puesto que la tarea de detección de *Design Smells* es consumidora de tiempo, solamente se van a proporcionar a los encuestados 5 clases que serán evaluadas por los sujetos para identificar en ellas posibles *Design Smells*. Como se ha mencionado anteriormente, se preguntará por la detección de *God Class* y *Feature Envy* en las clases seleccionadas. La elección de *God Class* y *Feature Envy* para el estudio es interesante ya que se trata de dos tipos de *Design Smell* diferentes como se explicó en la sección 2, lo que podría ser útil a la hora de extraer conclusiones de los resultados obtenidos.

Para facilitar la tarea del evaluador se suministrará a los encuestados una forma rápida de recordar la definición de dichos *Design Smells*. En el diseño de la encuesta, además de las preguntas directamente relacionadas con la detección de los DS, hay otra colección de preguntas que tiene como objetivo elaborar el perfil del sujeto encuestado, con el fin de comprobar si los factores del perfil influyen en el acuerdo en la detección de *Design Smells*, tal y como hemos explicado en las cuestiones de investigación.

La selección de las 5 clases debe cumplir los siguientes requisitos:

- se debe poder leer la clase separadamente y a la vez permitir ver la clase en su contexto para ayudar al evaluador en su tarea.
- las clases elegidas deben ser un subconjunto de las clases que se utilizan en el primer experimento de comparación entre herramientas que se presenta en [1].
- en el criterio de elección de las clases debemos incluir clases que sean candidatas a padecer *God Class*, *Feature Envy*, ambos o ninguno de acuerdo a algún oráculo de referencia.

Como fuente de las clases a evaluar en la detección de *Design Smells* se eligió un proyecto Apache de código abierto que se encuentra disponible en un repositorio público. De esta manera las clases mostradas a los evaluadores humanos

pueden ser consultadas muy legiblemente en su contexto proporcionando el enlace al repositorio de código. El proyecto Apache Lucene es uno de los más frecuentemente usados en las publicaciones sobre detección de *Design Smells* según un estudio realizado por estos autores. En concreto se ha utilizado la versión 3.1.0. El código de este proyecto se puede consultar en <http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.lucene/lucene-core/3.1.0>. Concretamente las clases elegidas para este experimento fueron:

- `org.apache.lucene.search.TopDocs`,
- `org.apache.lucene.queryParser.TokenMgrError`,
- `org.apache.lucene.util.ReaderUtil`,
- `org.apache.lucene.analysis.CharArraySet` y
- `org.apache.lucene.index.FieldsWriter`

Los detalles del diseño de la encuesta realizada para obtener respuestas de los evaluadores humanos se pueden consultar en <http://www.infor.uva.es/~yانيا/designsmells/#survey> donde se ha alojado un modelo de la encuesta. Con la intención de disponer de respuestas de evaluadores con diferentes perfiles y llegar al grupo de personas que podrían considerarse expertos en el tema de *Design Smells*, se recopilaron los e-mails de contacto de los autores de artículos sobre *Design Smells*. Se escribió directamente a dichos autores solicitando su participación en el estudio. Adicionalmente, los canales de difusión de la encuesta fueron diversos, desde el colegio de Ingenieros de Informática, linkedin y colegas de diferentes universidades; además se solicitó la difusión en un grupo de empresas de la región de Bélgica, Holanda y Alemania, a través de un colega con contactos profesionales en dichos países. Aunque desconocemos la ratio sujetos\_contactados/sujetos\_incluidos, consideramos que disponer de 92 respuestas válidas es un buen resultado, comparado con los resultados de trabajos similares (ver Sección 5). El tiempo en que la encuesta estuvo abierta para recibir respuestas fue de 3 meses.

#### 4. Estudio y análisis de las preguntas de investigación

Los sujetos que respondieron a la encuesta online fueron 93, de los que seleccionamos 92, pues una de ellas fue no válida. Así pues, dispondremos de 92 sujetos como evaluadores humanos de las 5 clases seleccionadas, que las clasificarán con *Design Smell* (*God Class* o *Feature Envy*) o sin *Design Smell*.

El grado de acuerdo entre los diferentes evaluadores lo examinamos con el coeficiente *Kappa-Fleiss* que permite comparar el grado de concordancia de  $r$  evaluadores en  $k$  objetos evaluados. La interpretación de este coeficiente se muestra en la Tabla 1. Hemos utilizado la herramienta R con la que se obtiene este coeficiente de concordancia usando el paquete `irr`, que contiene la función `kappam.fleiss()`. Esta función además nos proporciona un test de hipótesis para aceptar (o rechazar) que no hay concordancia entre los evaluadores.

Se obtuvieron respuestas de diferentes países. Bélgica, Canadá, India, Estados Unidos, Holanda, República Checa, Israel, Pakistán, Corea del Sur, Líbano, Malasia, Nueva Zelanda, Marruecos, Reino Unido, Emiratos Árabes, Jordania, Arabia Saudí, Suiza y España. Como la dispersión de los 92 sujetos por los 20

Valor de kappa, grado de concordancia	
$k < 0,20$	Pobre
$0,21 \leq k < 0,40$	Débil
$0,41 \leq k < 0,60$	Moderada
$0,61 \leq k < 0,80$	Buena
$0,81 \leq k \leq 1,00$	Muy buena

Tabla 1: Interpretación de los valores del coeficiente *Kappa-Fleiss* utilizada en el estudio.

Europa	Asia	América	África	Oceanía
35	13	22	7	15

Tabla 2: Distribución de sujetos por continente.

países no deja suficientes sujetos/pais para focalizar el estudio por países, decidimos analizar por áreas geográficas basadas en continentes. De esta manera se decidió diferenciar entre: Europa, Asia, América, África y Oceanía. La distribución del número de respuestas por continente se muestra en la Tabla 2. Se puede ver que en Europa es donde se localiza el mayor número de sujetos, y en África donde menos.

Por otra parte, la distribución de los sujetos por actividad laboral (Tabla 3) presenta la mayor frecuencia de sujetos en el Grupo 1 (Professor, Lecturer, Instructor) y la menor en el Grupo 4 (Software Architect). Cuando analizamos las respuestas obtenidas en el Grupo “Other”, encontramos “System admin”, “Quality Consultant”, “Engineer” y algún que otro “Msc Student” o “Student”. Esto nos motivó a dirigir los análisis posteriores clasificando esta variable en dos grupos: Industria y Academia, agrupando el Grupo 1 con el Grupo 2 como Academia, el grupo 3 y el grupo 4 como Industria y del Grupo 5, repartiendo a los estudiantes como Academia y al resto de respuestas en dicho Grupo 5 como Industria.

Mirando la distribución conjunta, el grupo con mayor número de sujetos fue: (Industria, entre 5 y 10 años de experiencia). El de menor frecuencia fue: (Academia, menos de 5 años de experiencia). Al considerar las distribuciones marginales, hubo más sujetos procedentes de la Industria que de la Academia. Además, el grupo de sujetos con entre 5 y 10 años de experiencia fue el más frecuente, y el de más de 10 años el que menos.

En la Tabla 5 se muestran tres distribuciones de frecuencias correspondientes a considerar el grado de formación condicionado por 3 factores determinados por los conocimientos sobre Escribir código Orientado a Objetos (OO), Revisar código OO y Experiencia en *Design Smells* (en la encuesta utilizamos el término *Code Smell* por ser más extendido entre los desarrolladores). Se puede ver que el mayor número de sujetos se concentra en aquellos cuyo grado de experiencia es Intermedio, sea cual sea el otro factor.

En las Tablas 6 y 7 se muestran los resultados obtenidos en la encuesta sobre el esfuerzo de los sujetos al detectar los *Design Smells*. Este esfuerzo se evalúa de dos formas. Por una parte en la Tabla 6 se presenta la distribución de frecuencias

Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5
35	13	22	7	15

Grupo 1 Professor, Lecturer, Instructor  
 Grupo 2 Researcher  
 Grupo 3 Programmer  
 Grupo 4 Software Architect  
 Grupo 5 Other

Tabla 3: Distribución de sujetos por actividad laboral.

Activities/Experience	1	2	3	Total	%
Academia	11	13	16	40	43,5%
Industria	17	24	11	52	56,5%

1 Menos de 5 años de experiencia  
 2 Más de 5 y menos de 10 años  
 3 Más de 10 años de experiencia

Tabla 4: Distribución conjunta de los sujetos por experiencia\*actividad profesional.

del tiempo empleado en detectar los *Design Smells*. A los encuestados se les informó de que necesitarían una media de 25 minutos para realizar dicha tarea, y aproximadamente un 80% tardaron menos de 25 minutos. La media de tiempo por clase fue de 6 minutos, lo cual confirma que detectar *Design Smell* sin ayuda de herramientas en proyectos grandes, es inviable. En cuanto a la Tabla 7, la mayoría de los sujetos (al menos un 79%) consideran que tanto la revisión del código desarrollado por terceros, como la detección de los *Design Smells* tienen una dificultad de media a difícil.

Con el fin de contestar a las cuestiones de esta investigación, vamos a utilizar el coeficiente *Kappa-Fleiss*, y el test de hipótesis relativo a la no concordancia, con un nivel de significación de 0,05. La hipótesis nula se define como:

$H_0$  No existe acuerdo entre evaluadores ( $kappa = 0$ )

Esta hipótesis se subdivide de la siguiente forma:

$H_0^{GC}$  No existe acuerdo entre evaluadores el detectar *God Class*

$H_0^{FE}$  No existe acuerdo entre evaluadores el detectar *Feature Envy*

Estudiaremos primero los test para la Hipótesis 1a, relativa a la concordancia entre sujetos. Y después la Hipótesis 1b, relativa a la concordancia entre sujetos y herramientas. Y en cada caso, el estudio se hará para cada *Design Smell* considerado, *God Class* y *Feature Envy*: Para hacer este análisis se han realizado más de 100 estudios con el coeficiente *Kappa-Fleiss*, con diferentes combinaciones relativas al perfil de los evaluadores. Vamos a agrupar las preguntas de investigación en dos grupos. En el primer grupo, las que afectan a una comparación entre evaluadores humanos, y en el segundo grupo, las que afectan a una comparación entre los evaluadores humanos y las herramientas.

**RQ1** ¿Existe acuerdo entre los evaluadores humanos al detectar *Design Smells*?

Se realiza un estudio del grado de acuerdo entre evaluadores (inter raters) con el estadístico *Kappa-Fleiss*. Los resultados de los p-valores y el coeficiente obtenido para cada caso (*God Class* y *Feature Envy*) se pueden ver en la Tabla 8. En el caso de *God Class* se obtiene un p-valor cercano al límite establecido para ser significativo (0,05) y el valor del coeficiente según la interpretación indica que el grado de acuerdo es Pobre. En el caso de *Feature Envy* el p-valor indica que no se puede rechazar la hipótesis nula por lo que se puede decir que no hay acuerdo al detectar, el grado de coincidencia es peor que una evaluación realizada al azar.

... grado de acuerdo entre evaluadores en la detección de *Design Smells* 9

Experiencia/Con	Escribir Código OO	%	Revisar Código OO	%	<i>Design Smells</i>	%
None	9	9,8 %	8	8,7 %	19	20,7 %
Principiante	14	15,2 %	17	18,5 %	26	28,3 %
Intermedio	44	47,8 %	46	50 %	35	38 %
Avanzado	25	27,2 %	21	22,8 %	12	13 %

Tabla 5: Distribución de frecuencias de la experiencia con las actividades Escribir Código OO, Revisar Código OO y *Design Smells*.

Porcentaje de evaluadores por intervalos de tiempo en minutos								
0-4	5-9	10-14	15-19	20-24	25-29	30-34	35-39	40-45
9,8 %	19,6 %	21,7 %	17,4 %	10,9 %	1,1 %	4,3 %	0 %	3,3 %

Tabla 6: Distribución de frecuencias del tiempo empleado en detectar los *Design Smells*.

**RQ5** ¿Cómo afecta el grado de experiencia en la detección de *Design Smells*?

**RQ5a** ¿El grado de acuerdo entre evaluadores humanos es mayor en el grupo de evaluadores con mayor experiencia?

Las primeras exploraciones sobre el acuerdo por grupos indicaron que la significación aumentaba si comparábamos el grupo de Inexperiencia (Menos de 5 años) con el grupo de Experiencia (Más de 5 años). Por esta razón, en lugar de establecer tres grupos de experiencia como se preguntaba en la encuesta, se dividieron los datos en dos grupos: menos de 5 años de experiencia y más de 5 años de experiencia. En la Tabla 9 se muestran los resultados del estudio del coeficiente *Kappa-Fleiss* y los p-valores obtenidos. Solamente se obtienen resultados significativos cuando el grupo de más de 5 años de experiencia detecta *God Class* pero el grado de acuerdo es muy pobre. En el resto de los casos no se puede rechazar la hipótesis nula por lo que no hay acuerdo entre los evaluadores.

**RQ6** ¿Cómo afectan los antecedentes (en términos de formación, experiencia y conocimientos) y el contexto de los evaluadores en la detección de *Design Smells*?

**RQ6a** ¿Afecta el contexto de trabajo? ¿el área geográfica donde se forma o desarrolla la persona? ¿si se trata de un contexto académico o de la industria?

Los estudios del grado de acuerdo al detectar los *Design Smells God Class* y *Feature Envy*, teniendo en cuenta el área geográfica, indican que este no varía de un área a otro, excepto para Europa. De hecho, todos los resultados (excepto en Europa) son no significativos, obteniéndose p-valores bastante grandes en todos los casos (por cada área, por cada *Design Smell*). El único caso en el que se obtiene un resultado significativo es en Europa al detectar *God Class*. El p-valor es próximo a cero y el valor de *Kappa-Fleiss* es de 0,18 muy próximo a lo que se interpreta como acuerdo “débil”.

Porcentaje de evaluadores por medición de la dificultad en la tarea						
Tarea/Dificultad	No contesta	Muy fácil	Fácil	Medio	Difficil	Muy difícil
Revisar el código	4,3 %	6,5 %	14,1 %	51,1 %	19,6 %	4,3 %
Detectar <i>Design Smells</i>	4,3 %	7,6 %	7,6 %	50 %	22,8 %	7,6 %

Tabla 7: Distribuciones de frecuencias de la dificultad encontrada al Revisar el código y al Detectar *Design Smells*.

$H_0$	p-value	Kappa-Fleiss	Conclusión
$H_0^{GC}$	0,0565	-0,00536	Cercano a significativo pero el grado de acuerdo al detectar <i>God Class</i> es muy pobre
$H_0^{FE}$	0,438	0,0132	No significativo. No hay acuerdo al detectar <i>Feature Envy</i>

Tabla 8: Resultados del estudio del grado de acuerdo entre evaluadores al detectar *God Class* y *Feature Envy*

Los test realizados para estudiar el efecto del entorno (Académico vs. Industria) en el grado de acuerdo al detectar los *Design Smells God Class* y *Feature Envy*, fueron todos no significativos, no había acuerdo. Por lo tanto no parece que el entorno influya en el grado de acuerdo entre sujetos al detectar *God Class* y *Feature Envy*.

**RQ6b** ¿Afectan los antecedentes del evaluador? Los antecedentes del evaluador se miden respecto a su experiencia en programación orientada a objetos, en lecturas o revisiones de código, y respecto a su nivel de conocimiento de *Design Smells*

En la Tabla 10, los casos resaltados en gris con el p-valor  $\geq 0,05$  indican que no se puede rechazar la hipótesis nula por lo tanto se asume que no hay acuerdo entre los evaluadores. Sin embargo en los otros casos, se puede admitir que hay acuerdo, que se puede interpretar mirando el valor del coeficiente *Kappa-Fleiss* como acuerdo muy pobre. Como se puede ver el acuerdo se produce en el grupo de mayor experiencia cuando detecta *God Class* y, sin embargo, en el grupo de los más inexpertos cuando detectan *Feature Envy*.

Es curioso que se repite el mismo patrón para los tres tipos de actividad. Una posible explicación puede ser que en el *Design Smell God Class* se relaciona con los conceptos de tamaño y de complejidad ciclomática. Sin embargo, *Feature Envy* se relaciona con conceptos de cohesión y acoplamiento en código OO, con la violación de los patrones GRASP. Suponemos que los más jóvenes están formados en estos conceptos en sus cursos de Ingeniería del Software y Programación. Ellos están al tanto de estos problemas, de ahí que tengan algún acuerdo detectando *Feature Envy*, aunque pobre. Sin embargo, los más experimentados tienen mejor desarrollados los aspectos relativos al manejo del tamaño y la complejidad. Los principiantes tienden a producir métodos y clases más largas y complejas. La inexperiencia les lleva a creer que pueden manejar el tamaño que los expertos consideran demasiado grande.

**RQ2** ¿Existe acuerdo entre los evaluadores humanos y las herramientas de detección de *Design Smells*?

... grado de acuerdo entre evaluadores en la detección de *Design Smells* 11

$H_0$	p-value	Kappa-Fleiss	Conclusión
$H_0^{GC}$	Inex. 0,215	Inex. -0,028	Significativo cuando los más expertos detectan <i>God Class</i> pero el acuerdo es muy pobre
	Exp. 0,00204	Exp. 0,0307	
$H_0^{FE}$	Inex. 0,524	Inex. -0,0146	No significativo. No hay acuerdo al detectar <i>Feature Envy</i> en ningún caso
	Exp. 0,267	Exp. -0,0111	

Tabla 9: Resultados del grado de acuerdo entre evaluadores del mismo grupo de menos de 5 años vs más de 5 años de experiencia

	Inexpertos (None+Beginner)			con Experiencia (Intermediate+Expert)		
	DS	<i>p</i>	<i>k</i>	DS	<i>p</i>	<i>k</i>
Exp. Escribir código OO	<i>God Class</i>	0,579	0,015	<i>God Class</i>	0	0,117
	<i>Feature Envy</i>	0,008	0,078	<i>Feature Envy</i>	0,605	0,005
Exp. Revisar código OO	<i>God Class</i>	0,159	0,036	<i>God Class</i>	0,011	0,024
	<i>Feature Envy</i>	0,005	0,073	<i>Feature Envy</i>	0,399	0,008
Exp. con <i>Design Smells</i>	<i>God Class</i>	0,960	0,001	<i>God Class</i>	0,006	0,001
	<i>Feature Envy</i>	0,527	0,009	<i>Feature Envy</i>	0,101	0,022

Tabla 10: Resultados del estudio por grupos de sujetos, clasificados por experiencia, respecto de diferentes actividades (Escribir código OO, Revisar código OO y Detección de *Design Smells*)

Para contestar a esta pregunta, se compara el resultado de las personas encuestadas con las mismas 6 herramientas utilizadas en el primer experimento que se presenta en [1] comparando las herramientas de detección, en este caso: Together, JDeodorant, iPlasma, inFusion, inCode y JSensorSmell cuando detectan *God Class* y *Feature Envy* en las clases del proyecto Apache Lucene, en particular sus resultados sobre las 5 clases que fueron preguntadas en la encuesta.

Los tests realizados indican que no existe acuerdo entre evaluadores humanos y herramientas en ninguno de los dos casos (*God Class*, *Feature Envy*). Los p-valores que se obtienen son 0,111 y 0,51, respectivamente.

**RQ4** ¿Qué herramienta está más cerca de los evaluadores humanos al detectar *Design Smells*?

Cuando se estudia el grado de acuerdo entre los sujetos y cada herramienta por separado, para detectar *God Class* y *Feature Envy*, se obtuvieron en todos los casos resultados no significativos, esto es, no hubo acuerdo. Sin embargo, en el caso del *God Class*, los p-valores estaban próximos a ser significativos (cerca de 0,05) en las 6 herramientas. Pero aún en estos casos, los valores del coeficiente *Kappa-Fleiss* darían un acuerdo muy pobre, oscilan entre 0,0617 y 0,0776. Se concluye que ninguna de las herramientas está más cerca de los evaluadores humanos en general, pero todas se acercan más en el caso de *God Class* frente a *Feature Envy*.

**RQ5b** ¿El grado de acuerdo con las herramientas de detección es mayor cuando el evaluador humano tiene mayor experiencia?

Se ha realizado un estudio cruzado, de cada herramienta con los diferentes grupos de evaluadores basados en grado de experiencia en las actividades relevantes para la detección de *Design Smells*, área geográfica de procedencia, contexto de trabajo Academia o Industria, años de experiencia en sus actividades profesionales. Por razones de espacio no podemos exponer todos los datos obtenidos pero podemos resaltar que en cada herramienta se obtuvo un patrón similar, que describiremos a continuación:

Se obtienen resultados significativos que permiten rechazar la hipótesis nula aunque los coeficientes *Kappa-Fleiss* sean muy pequeños y por tanto el acuerdo pobre para un perfil de evaluador que se describe como sigue:

- Evaluadores expertos en la actividad de escribir código OO y/o en revisión de código OO cuando detectan *God Class*.
- Evaluadores principiantes en la actividad de escribir código OO y/o en revisión de código OO cuando detectan *Feature Envy*.
- Evaluadores expertos en *Design Smells* cuando detectan *God Class*.
- Evaluadores de Europa cuando detectan *God Class*.

## 5. Trabajos relacionados

No se cuenta con muchos estudios empíricos sobre detección de *Design Smells* por evaluadores humanos o que comparen los resultados de evaluadores humanos con la detección realizada por herramientas.

Mäntylä en [7] describe dos experimentos realizados entre los años 2003 y 2004, publicados posteriormente en un artículo que aparece en el año 2005. En el primero de ellos, realizó una encuesta preguntando a un grupo de evaluadores la presencia en el código de tres *Design Smells*. Se trataba de tres *Design Smells* de nivel de método: *Long Method*, *Long Parameter List* y *Feature Envy*. Se preguntaba además a los evaluadores si pensaban que el método debía ser refactorizado para eliminar los *Design Smells*. En el segundo experimento, no se preguntó por la presencia de los *Design Smells* sino directamente si el método debía ser refactorizado. En el primer experimento participaron 46 evaluadores y en el segundo 36. Los evaluadores eran estudiantes de Máster de los cuales poco más del 50 % tenían cierta experiencia desarrollando en la industria. El resultado del primer experimento reveló altos niveles de acuerdo en los evaluadores en los dos primeros *Design Smells* más simples y considerablemente débiles al detectar *Feature Envy* y al decidir si había que refactorizar el método. En el segundo experimento se obtuvo un acuerdo débil en la única pregunta sobre si había que refactorizar el método.

Mäntylä et al [6] en el año 2004 realizó un experimento en una pequeña empresa de desarrollo de software en Finlandia. De los 18 desarrolladores contestaron a su encuesta 12. En la encuesta preguntó a los desarrolladores por la presencia de 23 *Design Smells* en el código de varios módulos desarrollados en la propia empresa. Cada evaluador aportó la evaluación de una media de 3 módulos, cada módulo obtuvo una media de cerca de 4 evaluaciones. A pesar

de los pocos datos concluyeron que los desarrolladores líderes detectaban más *Design Smells* de tipo estructural mientras que los desarrolladores regulares detectaban más smells del tipo *Duplicate Code* o *Dead Code*. Encontraron mucha subjetividad en la evaluación. Para tres *Design Smells* como *Large Class*, *Long Parameter List* y *Duplicate Code* realizaron un estudio con métricas de código y concluyeron que la evaluación subjetiva de los desarrolladores no correlacionaba con las métricas de código.

En 2012 Yamashita y Moonen [13] realizaron un estudio teniendo en cuenta las opiniones de 2 expertos externos a una organización y de 6 desarrolladores obtenidos de un proceso de selección para encargarles el mantenimiento de dos sistemas. Con esto obtuvieron la opinión de evaluadores humanos sobre la mantenibilidad del sistema antes y durante su mantenimiento. Es un artículo muy interesante de leer. Se identificaron 13 factores importantes de cara a la mantenibilidad, en 9 de ellos coincidieron los expertos y los desarrolladores encargados del mantenimiento. Esta información la intentaron correlacionar con la detección de *Design Smells* como indicador de mantenibilidad. Obtuvieron correlaciones parciales con algunos *Design Smells*, analizaron de esos cuáles pueden ser detectados automáticamente por herramientas como: *God Class*, *God Method*, *Lazy Class*, *Message Chain*, *Long Parameter List*, *Duplicate Code*, *Switch statements*, *Feature Envy*, *Shotgun Surgery*, *ISP Violation* para dar una visión cuantitativa de aspectos cualitativos como la mantenibilidad.

Nuevamente Yamashita y Moonen en 2013 [14] diseñaron una encuesta para explorar si los *Design Smells* son importantes para los desarrolladores y en caso de que no, si se debe a irrelevancia del concepto, desconocimiento por parte de los desarrolladores o carencia de herramientas apropiadas para detectar *Design Smells* y eliminarlos. Encuestaron a 85 desarrolladores profesionales contratándolos para realizar la encuesta a través de un portal de mercado de trabajo para freelance. El 32% contestó que nunca habían oído hablar de *Design Smells* o término similar. El 22% contestó que lo había oído o visto en algún blog pero no estaba seguro de saber lo que era. Esto resulta en más del 50% con prácticamente ningún conocimiento sobre el tema. El 21% conoce el concepto pero no lo aplica. Solamente el 18% declara tener buenos o sólidos conocimientos sobre el tema y aplicarlo en sus actividades diarias. También elaboraron un ranking de los *Design Smells* más conocidos por los encuestados. Sólo 2 de los 85 encuestados respondieron que usaban alguna herramienta de detección de *Design Smells*. Concluyeron los autores de forma muy similar a nuestro estudio, se necesita formación y divulgación, y se necesitan herramientas con algunas características deseables entre las que destacamos estar lista para usar pero con reglas configurables.

Palomba et al [8] publicaron en 2014 un estudio con 34 sujetos entre Estudiantes de Máster (15), Desarrolladores de Proyectos de Código abierto (10) y Desarrolladores trabajando en la industria del software (9). El objetivo del estudio era saber hasta qué punto los desarrolladores perciben los *Design Smells* como problemas de diseño que hay que solucionar y cuáles de estos *Design Smells* eran considerados más dañinos. Algunos *Design Smells* no fueron percibidos como problemas de diseño a solucionar. Los *Design Smells* relacionados con la complejidad y el tamaño como *God Class* son percibidos siempre como un problema, y en particular *God Class* fue el ganador en el ranking de los considerados

como más dañinos. El caso de *Feature Envy* es también muy interesante, es uno de los que más variabilidad presenta en las respuestas de los encuestados. Los autores concluyen en ese caso que se trata de posibles abusos/malos usos de los principios del diseño Orientado a Objetos.

## 6. Conclusiones y trabajo futuro

En este documento hemos presentado los resultados obtenidos al realizar una encuesta online, con el fin de que sujetos de diferentes perfiles y procedencias, identificaran los *Design Smells God Class* y *Feature Envy*, en 5 clases JAVA. A partir de esta información, hemos estudiado el grado de acuerdo entre dichos sujetos al detectar los *Design Smells* mencionados. Además, hemos estudiado el grado de acuerdo al detectar *Design Smells* entre sujetos y 6 herramientas, de las que se habían obtenido los datos en un estudio previo. La principal conclusión a la que llegamos es que hemos confirmado nuestra sospecha de que no existe acuerdo entre los desarrolladores en general y entre los desarrolladores y las herramientas, como consecuencia de lo anterior. En los casos en los que se produce algo de acuerdo, éste es débil o pobre, muy pobre en la mayoría de los casos.

Se detecta que existe un perfil de evaluador en el que se dan los mejores casos de acuerdo entre sí y con las herramientas. Este perfil nos indica que los desarrolladores experimentados coinciden mejor en cuestiones relativas a tamaño y complejidad, que los menos experimentados tienen más reciente el conocimiento sobre principios y patrones del diseño Orientado a Objetos y que se necesita mayor formación en *Design Smells*. En nuestro estudio el 49% de los encuestados se declara sin experiencia o principiante en cuanto a *Design Smells*. Solamente el 13% indica que se considera experto en la materia y esto considerando que hemos hecho un esfuerzo por alcanzar expertos en la materia mediante la elaboración de una lista de correo con los autores de artículos sobre detección de *Design Smells*.

Los resultados observados nos permiten elaborar algunas recomendaciones, para propiciar la adopción de técnicas de detección de *Design Smells* en la industria:

- Se debe incorporar formación sobre *Design Smells* en los estudios que preparan a los futuros desarrolladores de software e investigadores en Ingeniería del Software,
- Se debe contar con benchmarks consensuados, con el fin de asegurar que las herramientas cumplen unos mínimos que garanticen su utilidad para detectar *Design Smells*.
- Se necesita contar con la opinión de profesionales expertos en aquellas actividades relacionadas con la detección de *Design Smells*, como escribir código, revisar o leer código desarrollado por otros, y en conocimiento de *Design Smells* a la hora de validar los resultados de las herramientas de detección.

Como trabajo futuro, pensamos que es de interés realizar una réplica, con sujetos de perfil más homogéneo, y alta experiencia en los aspectos que son relevantes en la detección de *Design Smells*.

## Referencias

1. Alkharabsheh, K., Crespo, Y., Manso, E., Taboada, J.: Comparación de herramientas de detección de design smells. In: JISBD'2016, Salamanca, Septiembre (2016)
2. Brown, W.J., Malveau, R.C., McCormick III, H.W., Mowbray, T.J.: *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. John Wiley and Sons (March 1998), <http://www.antipatterns.com>
3. Fleiss, J.L.: *Statistical methods for rates and proportions*. New York: John Wiley, 2nd edn. (1981)
4. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: *Refactoring: Improving the Design of Existing Code*. Object Technology Series, Addison-Wesley (June 1999)
5. López, C., Manso, E., Crespo, Y.: Evaluación de la eficiencia de métodos de identificación del defecto de diseño God Class. In: XVII Jornadas de Ingeniería del Software y de Bases de Datos (JISBD 2012). Universidad de Almería (2012), <http://www.giro.infor.uva.es/Publications/2012/LMC12>
6. Mäntylä, M., Lassenius, C.: Subjective evaluation of software evolvability using code smells: An empirical study. *Empirical Software Engineering* 11(3), 395–431 (2006)
7. Mäntylä, M., Vanhanen, J., Lassenius, C.: Bad smells - humans as code critics. In: 20th International Conference on Software Maintenance (ICSM 2004), 11-17 September 2004, Chicago, IL, USA. pp. 399–408 (2004)
8. Palomba, F., Bavota, G., Penta, M.D., Oliveto, R., Lucia, A.D.: Do they really smell bad? A study on developers' perception of bad code smells. In: 30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014. pp. 101–110 (2014)
9. Riel, A.J.: *Object-Oriented Design Heuristics*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (April 1996)
10. Tiberghien, A., Moha, N., Mens, T., Mens, K.: Répertoire des défauts de conception. Tech. Rep. 1303, University of Montreal (2007)
11. Wake, W.C.: *Refactoring Workbook*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
12. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., Wesslén, A.: *Experimentation in Software Engineering: An Introduction*, International Series in Software Engineering, vol. 6. Springer (2000)
13. Yamashita, A.F., Moonen, L.: Do code smells reflect important maintainability aspects? In: 28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012. pp. 306–315 (2012)
14. Yamashita, A.F., Moonen, L.: Do developers care about code smells? an exploratory survey. In: 20th Working Conference on Reverse Engineering, WCRE 2013, Koblenz, Germany, October 14-17, 2013. pp. 242–251 (2013)