Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Analysing Agreement among Different Evaluators in God Class and Feature Envy Detection

KHALID ALKHARABSHEH[1], SADI ALAWADI[2], YANIA CRESPO[3], ESPERANZA MANSO[3], JOSE A. TABOADA[4]

[1] Department of Software Engineering, Prince Abdullah bin Ghazi Faculty of Information and Communication Technology, Al-Balqa Applied University (BAU), As salt, Jordan (e-mail: khalidkh@bau.edu.jo)
[2] Department of Information Technology, Uppsala University, Box 337, 75105, Uppsala, Sweden (e-mail: sadi.alawadi@it.uu.se)
[3] Departamento de Informática. Escuela de Ingeniería Informática. Campus Miguel Delibes, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011. Spain (e-mail: yania@infor.uva.es, manso@infor.uva.es)
[4] CiTIUS, Centro Singular de Investigación en Tecnoloxías da Información, Universidad de Santiago de Compostela, Santiago de Compostela 15782. Spain (e-mail: joseangel.taboada@usc.es)

Corresponding author: Khalid Alkharabsheh (e-mail: khalidkh@bau.edu.jo), Sadi Alawadi (e-mail: sadi.alawadi@it.uu.se).

ABSTRACT The automatic detection of *Design Smells* has evolved in parallel to the evolution of automatic *refactoring* tools. There was a huge rise in research activity regarding *Design Smell* detection from 2010 to the present. However, it should be noted that the adoption of *Design Smell* detection in real software development practice is not comparable to the adoption of automatic *refactoring* tools. On the basis of the assumption that it is the objectiveness of a refactoring operation as opposed to the subjectivity in definition and identification of *Design Smells* that makes the difference, in this paper, the lack of agreement between different evaluators when detecting *Design Smells* is empirically studied. To do so, a series of experiments and studies were designed and conducted to analyse the concordance in *Design Smell* detection of different persons and tools, including a comparison between them. This work focuses on two well known *Design Smells*: *God Class* and *Feature Envy*. Concordance analysis is based on the Kappa statistic for inter-rater agreement (particularly *Kappa-Fleiss*). The results obtained show that there is no agreement in detection in general, and, in those cases where a certain agreement appears, it is considered to be a fair or poor degree of agreement, according to a *Kappa-Fleiss* interpretation scale. This seems to confirm that there is a subjective component which makes the raters evaluate the presence of *Design Smells* differently. The study also raises the question of a lack of training and experience regarding *Design Smells*.

INDEX TERMS Design smell, Survey, Empirical Study , Experiment, Inter-rater Agreement, Kappa-Fleiss

I. INTRODUCTION

Ward Cunningham introduced the term "Technical Debt" as a metaphor in an experience report presented at OOPSLA'92. According to Fowler, the comparison to a financial debt suggests that the technical debt incurs interest payments in the form of the extra effort that has to be made in a software project in the form of maintenance activities of any kind (corrective, adaptive, and 7 perfective).

[1], speaks about design heuristics and insists on the need to identify and correct the non compliance of the said heuristics. Kent Beck, in the late 90's, coined the term *Code Smell* and wrote the term for the first time in Cunningham's Wiki[1]. Meanwhile, [2] (including Beck, Brand and Opdyke) popularised the terms *refactoring* and *Bad Smells* beyond an academic context with their seminal book. This popularisation started in the year 2000. [3] used the term *AntiPattern* and, in parallel, [4] and [5] published their works on *Design Flaws*. The same term was later used by [6]. [7] defined the term *Disharmonies* and elaborated a detection method based on metrics. In [8], the term *Design Smell* was proposed as a unifying concept for these many related terms. We adopted this term in a Systematic Literature Review on *Design Smell* Detection [9].

Design Smells are problems in software structure that do not produce compilation or runtime errors, but which negatively affect such software quality factors as reusability, stability, understandability, and maintainability as defined in

[1] http://c2.com/cgi/wiki?CodeSmell

[8, 9, 10].

Cunningham, in 2008, relaunched the Technical Debt concept in a series of conferences across the world. This concept is correlated with the presence of *Smells*. [11] state that a poor structure in source code or design (*i.e.* the definition of *Design Smell*) is one of the most important factors contributing to Technical Debt.

Tools that assist *Design Smell* detection began to emerge as a result of the essential role in improving the software structures(code and design) and their quality and controlling technical debt and its correlation with *Design Smells*. The first *Design Smell* detection tool to appear was reported in 2002 (jCOSMO: [12]). There has been a continuous rise in the appearance of new detection tools since 2004. After 2010 research activity regarding *Design Smell* detection has experienced a rapid and huge growth. Different approaches and techniques have been proposed with respect to the identification and correction of *Design Smells* ranging from manual, semi-automated to fully automated [9, 13, 14, 15, 16, 17, 18, 19, 20]. In addition, most of the approaches were mapped into detection tools with different capabilities and deal with different programming languages. These tools can be dedicated (standalone) or integrated (as plug-ins in development environments, or as automatic ant, maven, sonarqube tasks). Despite the diversity of detection tools in the literature that should improve the detection of *Design Smells*, the adoption of these techniques remains a huge challenge for the software industry, in contrast to the adoption of refactoring tools in the development and maintenance processes, because the current techniques suffer from high false positive and false negative rates (low precision and recall) and very few recommend solutions (*refactoring*) for the detected *Design Smells*.

The authors of [17, 21, 22] achieved tool comparisons based on a set of particular *Design Smells*. Their works show that different tools obtained different results when they analysed the same software. Nevertheless, even when results are different, there can still be some concordance or agreement. [23] and [15] compare detection tools and perform agreement analysis. The authors of [13, 18, 19, 24, 25, 26, 27, 28] focused their attention on the role of subjectivity and how it influences *Design Smell* detection. Subjectivity may be related to the persons (their background, experience, ...), to the workplace (the organisation), or to the software (its domain, dimensions, complexity, ...).

These ideas regarding subjectivity led us to design a series of experiments to evaluate empirically to what extent *Design Smell* detection tools agree, to what extent persons agree when deciding the presence of a *Design Smell* in software, and to what extent persons and tools agree. The experiments were conducted with different types of evaluators to detect two types of *Design Smells*: *God Class* and *Feature Envy* in a set of open source projects.

The first experiment is related to tool evaluation, with a set of detection tools involved in analysing a single software project. The second experiment is related to expert evaluation using a questionnaire (online survey), in which persons evaluate the presence of *Design Smells* in a sample of classes from the same project used in first experiment. The survey questions allow to assess the impact of subjective indicators related to the human context (background, experience, etc.) to be analysed. Mixing data from the first and second experiments allowed us to study concordance between persons and tools. All concordance analyses were based on the Kappa statistic ([29]) for inter-rater agreement, particularly *Kappa-Fleiss* (see [30]) using R.

With the intention of circumventing some threats to the validity, such as the selected detection tools or the analysed software project, a third experiment was conducted as a replication of the first, introducing new *Design Smell* detection tools and a dataset assembled with more than $12,000$ classes from 24 open source projects. In addition to the *Kappa-Fleiss* analysis, due to the dataset dimensions in this experiment, Formal Concept Analysis (FCA) (see [31]) was applied to study the relationships between the detections accomplished by the tools. Finally, mixing data from the second and third experiments enabled us to study the concordance between persons and the new set of tools.

The rest of this paper is organised as follows: Section II describes the conceptual framework, the *Design SmellsGod Class* and *Feature Envy* are described, as well as the tools involved in the experiments, and the *Kappa-Fleiss* and FCA analysis are briefly explained. Section III describes the problem to be solved and Section IV explains the design of the experiments. Then, Section V analyses the results of the experiments and Section VI presents the discussion. Next, section VIII discusses the main threats to the validity. Section VII presents some related work. Finally, Section IX presents our conclusions and the direction we intend to take in future work.

II. BACKGROUND

If the readers are aware of these topics, they can jump to the next section.

A. THE GOD CLASS AND FEATURE ENVY DESIGN SMELLS

This study focuses on two different kinds of *Design Smells*: *God Class* and *Feature Envy*. On the one hand, *God Class* is an intra-class *Design Smell*, *i.e.*, it is sufficient to observe the single class to detect the smell and its scope is the class. It is the class which suffers from the smell. On the other hand, *Feature Envy* is an inter-class *Design Smell*, *i.e.* it requires observation of the interaction of the class with other classes to detect the smell. In terms of scope, its scope is on a method level. It is a method which suffers from the smell. A tool detecting *God Class* should look at the class as a whole. A tool detecting *Feature Envy* should look inside the method details in order to observe how interaction with other classes is achieved. When metrics are involved in the detection, different sets of metrics are relevant to detect each of them.

[1] defined a *God Class* as the class of an object controlling too many objects in the system, and which has grown beyond all logic to become The Class That Does Everything. In good Object-Oriented Designs, the logic of the system is uniformly distributed across multiple classes. A *God Class* has too many instance variables and too much code. The greater the amount of code present, the greater the danger of duplicated code. Sometimes a *God Class* is the result of a wrong application of the Mediator Design Pattern. A *God Class* can be absorbed by the *Large Class* bad smell defined by [2]. It is also known as the antipattern *"The Blob"*, as defined in [3].

Feature Envy is part of [2]'s catalogue. [32] classifies *Feature Envy* in the category named "inter-class" and the subcategory "responsibility". *Feature Envy*'s scope is method level, as mentioned before. A method suffering from *Feature Envy* seems to be more concerned with manipulating data from other classes than from its own class. It is related with class responsibility because it is evidence of having assigned a method (a responsibility) to the wrong class. As exceptions, Fowler *et al.* explain that there are some Design Patterns that break this rule, such as Visitor and Strategy.

According to the classification of *Design Smells* developed by [33], 7 *God Class* belongs to the group named *"Bloaters"*, while *Feature Envy* belongs to the group named *"Couplers"*.

B. DESIGN SMELL DETECTION TOOLS

As mentioned above, the first *Design Smell* detection tool reported was jCOSMO in 2002. In 2011, as a result of a research project conducted by our group, two Technical Reports were published (see [8, 34]). These reports analysed and classified some of the *Design Smell* detection and correction tools proposed up to that point. Once the tools were studied, revised and classified, we discarded from the input list those tools that only check code style rules and also those tools that only carry out correction activities (based on *refactoring*) but not detection. We thus obtained a set of *Design Smell* detection tools available in 2010 such as: Analyst4j, Cultivate (from jTransformer suite), DÉCOR, iPlasma, inCode, inFusion, jDeodorant, an emerging PMD, Reek, RevJava, SA4J and Together.

From 2010 to the present, new studies have been conducted to organise the knowledge on *Design Smell* detection tools. [35] carried out a review of *Code Smell* detection techniques and tools from 2000 until 2015. As can be seen, a new group of tools emerge including: Stench Blossom, ConcernReCS, SourceMiner, BSDT, JCodeCanine, GrouMiner, CodeVizard, JSNose, Hist-Inspect, SVMDetect, PTIDEJ suite (containing DÉCOR and its evolution DETEX), BLOP, and an evolution of the previously emergent PMD with a new set of rules for *Design Smell* detection. Moreover, there is a set of research prototypes without any particular name that implement the techniques reported by their authors in different publications (see for example [36, 37, 38]).

In section IV, the set of tools selected for each experiment are briefly described.

C. KAPPA-FLEISS

A large number of situations rely on many people collecting research data and evaluating them. The question of consistency, or agreement among the individuals, emerges due to the variability between human observers. Perfect agreement is seldom achieved. The extent of agreement among evaluators is called "inter-rater reliability". Traditionally, this was measured as a percent of agreement, calculated as the number of agreement scores divided by the total number of scores. There are a number of appropriate statistical tests which can be used to measure the agreements between different evaluators, such as Cohen's kappa, Scott's pi, and Fleiss' Kappa. In 1955, William A. Scott proposed the pi coefficient to determine the agreement of two raters assigning items to nominal categories. In 1960, Jacob Cohen criticised the use of percent agreement due to its inability to take into account the possibility of the agreement occurring by chance.

[29] introduced his Kappa statistic (for two raters), developed to account for the possibility that raters actually guess on at least some variables due to uncertainty. Kappa measures the degree of agreement (or concordance) of the nominal or ordinal assessments made by appraisers when assessing the same samples. Kappa can range from -1 to $+1$. The higher the value of Kappa, the stronger the agreement.

Fleiss' Kappa is an adaptation of Scotts pi and Cohen's Kappa for 3 or more raters (see [30]). In this study, we use Fleiss' Kappa (*Kappa-Fleiss*) because we have more than two appraisers.

Kappa-Fleiss allows the degree of agreement of r raters on k evaluated objects to be measured. Table 1 shows the interpretation of this coefficient.

TABLE 1: Interpretation of the *Kappa-Fleiss* values used in this paper.

Kappa-Fleiss value	Degree of Agreement
$k < 0.20$	Poor
$0.21 \leq k < 0.40$	Fair (Weak)
$0.41 \leqslant k < 0.60$	Moderate
$0.61 \leqslant k < 0.80$	Substantial (Good)
$0.81 \leq k \leq 1.00$	Almost perfect (Very Good)

D. FORMAL CONCEPT ANALYSIS

Formal Concept Analysis (FCA) (see [31]) is a formal technique that allows the underlying structure in a set of data to be automatically extracted. The basic technique in FCA consists of the elaboration of an incidence matrix, denominated formal context in terms of the theory. Starting from this formal context, a Galois lattice is obtained in an algorithmic way, and the lattice is represented by means of its corresponding Hasse diagram, which contains the original information in its entirety, but organised in a way that shows the underlying structure of the data. The rows of the incidence matrix represent objects, while their columns are attributes, and the

incidence represents the presence of an attribute in a given object. The terms object and attribute are the usual terms in FCA theory, and have no relationship with the homonymic terms of Object Orientation.

The nodes of the lattice, so called formal concepts, are then formed by a pair of sets of objects and attributes that mutually determine each other. The lattice constitutes a partial order relation which is determined by the inclusion relationship between the set of objects, as well as by the contention relationship between the set of attributes. Therefore, applying this formal tool requires the way in which the incidence matrix is built to be defined, determining what will be interpreted as objects and attributes, respectively, and the form in which the lattice should be interpreted in the problem being modelled. Different incidence matrices allow several structures in the original data to be made clear.

III. PROBLEM STATEMENT, GOALS AND RESEARCH QUESTIONS

As mentioned before, in the state of the art, several tools have been proposed to detect *Design Smells*. However, despite the boom of research in this field, the adoption of these tools in industry is poor compared to that of refactoring tools. This comparison in adoption is particularly relevant if we consider that *Design Smell* detection is closely related to refactoring, as refactoring opportunities indicators. Refactoring tools have been adopted in the industry as an automated mechanism for high-level code editing, as well as by agile practices and Test Driven Development (TDD) as part of the *Red-Green-Refactor* cycle that embraces the change.

In this work, the supposition we have raised is that the absence of adoption in *Design Smell* detection tools is related to the lack of agreement between them and the subjectivity of deciding the presence of *Design Smells*, which seems to be intrinsic to the problem. The research community has not taken into account in depth the impact of subjective assessment on the design of *Design Smell* tools.

A. RESEARCH QUESTIONS

To address the problem being studied, we introduce the following research questions to examine the degree of agreement between the different types of evaluators: tools, experts, and tools vs. experts. In our opinion there is a set of factors related to the subjectivity assessment that affects *Design Smell* detection. These include the degree of evaluator experience, their background regarding training, knowledge and developing activities, and their work context which includes the geographical areas were the evaluators are from and/or work. We aim to identify what types of interrelation can be found between these factors, and their influence on *Design Smell* detection.

- RQ1 Is there any agreement between the selected tools in *God Class* and *Feature EnvyDesign Smell* detection?
- RQ2 Which of the selected tools coincide more closely when detecting in *God Class* and *Feature EnvyDesign Smell* detection?
- RQ3 What is the degree of agreement between human evaluators in *Design Smell* detection?
- RQ4 What is the degree of agreement between human evaluators and tools in *Design Smell* detection?
- RQ5 Which tools coincide more with human evaluators in *Design Smell* detection?
- RQ6 How does the degree of experience affect *Design Smell* detection?
  - RQ6a Is the degree of agreement between human evaluators higher when the group of evaluators has more experience?
  - RQ6b Is the degree of agreement with detection tools higher when the group of evaluators has more experience?
- RQ7 How does the background (regarding training, experience and knowledge) and context of evaluators affect *Design Smell* detection?
  - RQ7a Does the work context, geographical area where the developers are from or whether the context is industrial or academic have any effect?
  - RQ7b Does the evaluator's background (regarding expertise in object-oriented programming, in code reviewing or his/her knowledge level on *Design Smells*) affect the degree of agreement?

To this end, the following null hypotheses have been formulated :

Hypothesis 1: There is no agreement when detecting *God Class* and *Feature EnvyDesign Smells*.

This hypothesis in general, can be separated into the following secondary hypotheses:

Hypothesis 1.a: There is no agreement between detection tools when they detect *God Class* and *Feature EnvyDesign Smells*.

Hypothesis 1.b: There is no agreement between human evaluators when they detect *God Class* and *Feature EnvyDesign Smells*.

Hypothesis 1.c: There is no agreement between human evaluators and detection tools when they detect *God Class* and *Feature EnvyDesign Smells*.

As can be seen, the first hypothesis 1.a relates to RQ1 and RQ2, while the second hypothesis 1.b relates to RQ3, RQ6, RQ6a, RQ7, and hypothesis 1.c to research questions RQ4, RQ5, RQ6b.

IV. DESIGN OF THE EXPERIMENTS

We examined the degree of agreement between different evaluators (tools or humans) using the *Kappa-Fleiss* coefficient, interpreting the results as usual in Software Engineering (see Section II for *Kappa-Fleiss* explanation and Table 1 for interpretation of values). The R tool was used in our experiments to compute this coefficient, particularly

the `irr` package. This package contains `kappam.fleiss()`. In addition, the function obtains the results of a hypothesis test that allow us to accept (or reject) whether there is no agreement between evaluators.

Two initial experiments were conducted to identify the agreement between the different evaluators (tools, human) on detecting the popular *Design Smells*: *God Class* (GC) and *Feature Envy* (FE). Figure 1 presents an overview of the experiments and studies over the scheduled time.

As a first stage, we performed the tool experiment (E1), which investigates the degree of agreement between the selected detection tools. Then, we performed the human experiment (E2), which investigates the degree of agreement between human evaluators. After that, using the available data from the experiments E1 and E2, we conducted the tool-human study (S1) to evaluate the degree of agreement between tools and human evaluators. Later on, based on the lessons learned from E1, we decided to replicate the same experiment, but with a new set of detection tools and a large set of projects to detect only the *God ClassDesign Smell*. The replicated study is referred to as R1. Finally, based on the available data from E2 and R1, a new study (S2) was conducted to evaluate the degree of tool-human agreement on *God Class* detection. The conducted experiments and studies will be described in more detail in the following subsections.

A. TOOLS EXPERIMENT (E1) DESIGN

The design of the first experiment E1 was formulated from the data collected by the technical reports of the GIRO group in the University of Valladolid (check [8, 34]). The reports evaluate a comprehensive set of detection and correction *Design Smell* tools. On the basis of these reports, we discarded the correction tools from our selection and only focused on the 24 available *Design Smell* detection tools. Through the study of the characteristics of the detection tools, approximately 50% were selected that support *Design Smell* detection in Java source code. Then, a filter was applied on these tools in order to determine which of them can analyse projects that have been developed with the same version of Java and, at the same time, can detect a common group of *Design Smells*.

In the tool selection process, some new information was found that adds new suppositions to the lack of detection tool adoption in industry: there is no *Design Smell* corpus common to several tools.

In this study, we focused on *God Class* and *Feature EnvyDesign Smells* because they are the most cited smells in the literature according to our systematic mapping of *Design Smell* detection [9].

The selected smells can be detected by several tools and permit a comparison between them using the same selected tools. In addition, selecting *God Class* and *Feature Envy* is very interesting as they are two different types and present different perspectives (see section II).

Following these criteria for tools to be included in the experiment:

- can detect *Feature Envy* and *God Class*,
- can work with Java projects of the same Java version
- produce textual reports (output format)
- should be available for our team;

five third-party detection tools were selected (inFusion, inCode, iPlasma, JDeodorant, Together), while another tool (JSmellSensor: [40]), developed by our team as part of a PhD Thesis (see [41]) based on automatic learning, which is based on initial knowledge taking the inCode and JDeodorant tools as experts, also fulfils the criteria and was included as well.

Table 2 summarises the most important characteristics of the selected tools; such as the version, whether the tools are open source and free or not, the supported languages, the term used to describe *Design Smells*, the ability to refactor after detecting a smell, the way to run the tool (execution environment), their ability to generate metrics, the type of input source, the output format, and the ability to work with Command Line Interface (CLI) or the need to use it through a Graphical User Interface (GUI).

The set of classes to be evaluated by the selected tools was obtained from the Apache Lucene version 3.1.0. This is an open source project, written in Java (version 1.5), includes 15 packages, 533 classes, 3,997 methods, and 45,416 line of code, as summarised in Table 3. The Apache Lucene project is one of the most frequently used in *Design Smell* detection publications, according to our the study presented in [9]'s systematic mapping of *Design Smell* detection previously mentioned , particularly the version 3.1.0, the one selected in this experiment. The source code of the project with the same version is freely available on public repository and can be obtained from the URL: http://grepcode.com/snapshot/repo1.maven.org/maven2/org.apache.lucene/lucene-core/3.1.0. Figure 2 summarises the main characteristics of the first experiment.

B. HUMANS EXPERIMENT (E2) DESIGN

In this experiment, a web-based survey was designed to detect *God Class* and *Feature EnvyDesign Smells* by subjects (human evaluators). The survey included a sample of classes given as candidates for the presence of *Design Smells* and asked the evaluator whether he/she could detect *God Class* or *Feature Envy*, both or neither of them. Since it is assumed that the task of *Design Smell* detection is time-consuming, only five classes were supplied to the respondents to be evaluated in order to identify the possible *Design Smells*. Thus, the subjects did not require much time to complete the survey.

To facilitate the evaluator's task, we supplied the respondents with a quick way to remember the definition of such *Design Smells*. In the survey design, in addition to the direct questions relating to *Design Smell* detection, there was another group of questions that aims to make up the profile of the respondent subject, to check if the profile factors influence the degree of agreement of *Design Smell* detection or not.

In order to select the five classes the following requirements were taken into account:

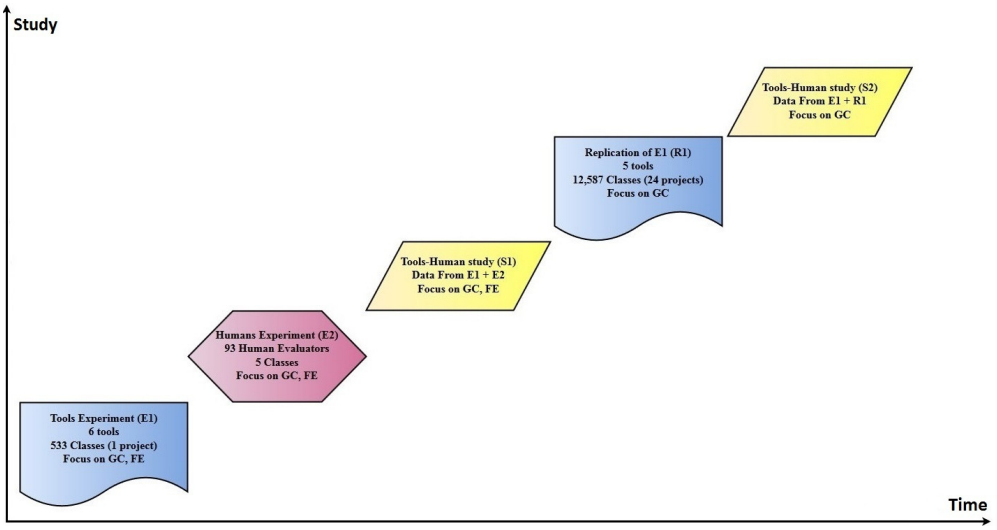


FIGURE 1: Experiments and studies overview.

TABLE 2: Characteristics of the selected detection tools in experiment E1.

Tool	inFusion	inCode	iPlasma	JDeodorant	Together	JSmellSensor
Version	1.8.4	2.0.7	6.1	5.0.13	12.6.0	1.1.0
Supported Java Version	1.6	1.6	1.5.02	1.7.0_22 or more	1.5.0_22 or 1.6.0	1.5.0
Available	No	No	Yes	Yes	Yes	Yes
Free	No	No	Yes	Yes	No	No
Type	Proprietary	Proprietary	Open Source	Open Source	Proprietary	Proprietary
Supported languages	Java, C++, C	Java, C++, C	Java, C++	Java	Java, C++, C#	Java
Smell term	Design Flaw	Design Flaw	Disharmony	Bad Smell	Design Flaw	Bad Smell
Design Smells	BM, CD, DC, FE, GC, IC, MTM, RB, SD, SS, SB, ID, ED, B, BO, DaC, MsC, DH, SC, TB, UD	DC, DaC, DupC, FE, GC, GM, MC, RB, SS, MsC, SC	BC, BM, DC, DisC, FE, GC, IC, SS, RB, TB, LM, LPL, SG	FE, GC, LM, TC	GP, SS, RB, FE, GC, GM, DupC, MC, MsC, DC, ISP, DaC	FE, GC, DC
Refactoring	No	No	No	Yes	No	No
Environment	Standalone	Eclipse Plug-in, Standalone	Standalone	Eclipse Plug-in	Eclipse Plug-in	Eclipse Plug-in
Generates metrics	Yes	Yes	Yes	No	Yes	No
Input data	Source code	Source code	Source code	Source code	Source code, UML model	Source code
Output data	Textual, Visual	Textual, Visual	Textual, Visual	Textual	Textual	Textual
Command Line Interface (CLI)	No	No	No	No	No	Yes

Design Smell acronym used in table:
FE = Feature Envy, **GC** = God Class, **DC** = Data Class, **GP** = God Package, **SS** = Shotgun Surgery, **RB** = Refused Bequest, **GM** = God Method, **DupC** = Duplicate code, **MC** = Misplaced class, **MsC** = Message chain, **ISP** = Interface Segregation Principle violation, **DaC** = Data Clump, **LM** = Long Method, **TC** = Type Checking, **BC** = Brain Class, **BM** = Brain Method, **DisC** = Dispersed Coupling, **IC** = Intensive Coupling, **TB** = Tradition Breaker, **LPL** = Long Parameter List, **SG** = Speculative Generality, **SC** = Schizophrenic class, **CD** = Cyclic Dependencies, **MTM** = Missing Template Method, **SD** = Sibling Duplication, **SB** = SAPBreakers, **ID** = Internal Duplication, **ED** = External Duplication, **B** = Blob, **BO** = Blob Operation, **DH** = Distorted Hierarchy, **UD** = Unstable Dependencies.

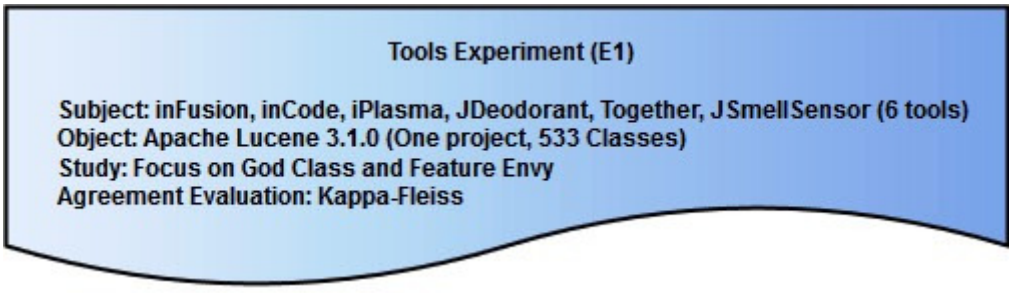


FIGURE 2: Description of the first tool experiment (E1).

- It should be possible to read the class separately and at the same time allow it to be seen in its context so as to assist the evaluator in his/her task.
- The chosen classes should be a subset of the classes used in the first experiment (E1) regarding tool comparison.
- As the criterion of class selection, classes that are candidates to suffer from *God Class*, *Feature Envy*, both or neither, according to some reference.

TABLE 3: Characterization of Apache Lucene project (version 3.1.0).

Metric Acronym	Metric Name	Value
NOP	Number of Packages	15
NOC	Number of Classes	533
NOM	Number of Methods	3,997
LOC	Lines of Code	45,416

As mentioned above, the source code of the classes to be evaluated are available in a public repository. In this way, the respondents will be given a link to show the class in its context. Specifically, the selected classes were:

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

To obtain responses from subjects with different profiles and to reach the group of persons who could be considered experts in the *Design Smell* domain, we collected the emails of the authors of articles related to *Design Smells*. We then contacted them directly, requesting their participation in the study. The design details of the survey conducted to obtain the responses of human evaluators are available on https://www.infor.uva.es/~yania/designsmells/#survey.

In addition, the diffusion channels of the survey were diverse, from the Spanish professional association of Informatics Engineering, LinkedIn, ResearchGate, to colleagues from different universities. Also, a group of software companies in Belgium, the Netherlands, and Germany was contacted to participate in the study through a colleague who had professional relations with those companies. Despite the fact that we have no knowledge about the ratio of subjects contacted/subjects included, we consider 93 responses to be a good result. The survey remained open to the receipt of responses for three months. Figure 3 summarises the main characteristics of the second experiment, which concerned subjects, objects, the focus of study and the statistical test. From the 93 respondents, we considered 92 as valid answers and therefore we discarded the invalid one. In the rest of the paper the experiments and studies mentioned 92 subjects as human evaluators.

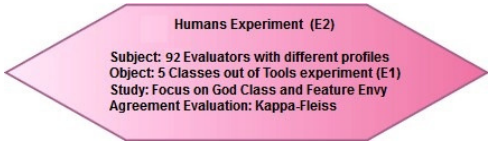


FIGURE 3: Description of humans experiment (E2).

C. TOOL-HUMAN JOINT STUDY (S1) DESIGN

In order to evaluate the degree of agreement between detection tools and human experts, a new study (S1) was carried out, using the information obtained from E1 and E2 as shown in Figure 1.

As can be seen in Figure 4, the set of subjects was made up of the six detection tools used in E1 in addition to the 92 human evaluators obtained from E2. The set of objects was made up of the previously five selected classes in E2, which belong to the same project used in E1. Therefore, from E1 and E2, we have the necessary information on how evaluators (tools and humans) detect *God Class* and *Feature Envy* on the above mentioned five classes.

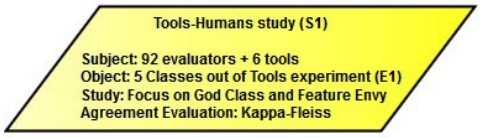


FIGURE 4: Description of human-tool experiment (S1).

D. TOOL EXPERIMENT REPLICATION (R1) DESIGN

Having analysed the results of experiments E1, E2, and S1, a replication study (R1) was designed as can be seen in Figure 1. R1 maintains the same objectives and research questions as in E1 as Figure 5 shows. According to the analysis of the E1 results (see Section V-A), the criteria in this study for selecting the detection tools were based on those most cited in the current state of the art [7, 12, 17, 21, 22, 24, 25, 26, 42, 43]. The selected tools were iPlasma, JDeodorant, Together, DÉCOR and PMD, of which three were the same as those used in the first experiment E1. Table 4 presents the characteristics of the two tools DÉCOR and PMD, included for the first time in R1. The reasons for maintaining iPlasma but removing inFusion and inCode from the experiments can be found in Section VI-A.

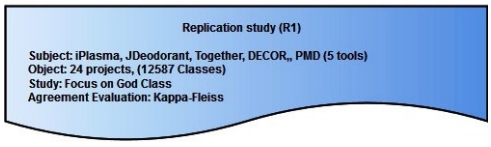


FIGURE 5: Description of replication of E1 experiment (R1).

During the preparation of R1, we decided to work with a large set of classes from different projects. The selected projects were chosen randomly from the SourceForge repository, taking into account different characteristics in terms of project size, domain and status. The selected objects are made up of 12,587 classes from 24 open source projects written in Java. Due to the large size of the dataset in R1, we focused solely on *God Class* detection.

Table 5 shows a description of the main characteristics of the selected projects in the R1 study, such as the name, version, number of classes (NOC) and total lines of code (TLOC). All the information collected on these 24 projects and their classes is part of another study under revision

TABLE 4: Characteristics of the new selected detection tools in R1, as replication of E1.

Tool	PMD	DECOR
Version	5.3.2	1.0
Supported Java Version	1.4 or more	1.5.0
Free	Yes	Yes
Type	Open source	Open source
Supported language	Java, C, C++, C#, PHP, etc.	Java
Smell term	Design Flaw	Code smell, Antipattern
Design smells	LC, LM, LPL	LC, LzC, LM, LPL, RB, SG, MsC, SS, DupC, C, DC, NP, GV, CC, PC, LC, DivC, B, SpC, FD, SAK
Refactoring	No	No
Environment	Eclipse Plug-in, Standalone	Standalone
Generates metrics	No	No
Input data	Source code	Source code
Output data	Textual	Textual
Command line (CLI)	Yes	No

LC = Large Class, **LM** = Long Method, **LPL** = Long Parameter List, **LzC** = Lazy Class, **RB** = Refused Bequest, **SG** = Speculative Generality, **MsC** = Message Chains, **SS** = Shotgun Surgery, **DupC** = Duplicate Code, **C** = Comments, **DC** = Data Class, **NP** = No Polymorphism, **GV** = Global Variable, **CC** = Controller Class, **PC** = Procedural Class, **LC** = Low Cohesion, **DivC** = Divergent Change, **B** = Blob, **SpC** = Spaghetti Code, **FD** = Functional Decomposition, **SAK** = Swiss Army Knife

using the five selected detection tools. The dataset prepared is available for researchers at the CiTiUS site[2].

TABLE 5: Characterization of the selected projects showing project name, version, number of classes and total lines of code.

Project name	Version	NoC	TLOC
jAudio	1.0.4	416	117,615
Freemind	1.0.1	782	106,396
JasperReports	4.7.1	1797	350,690
SQuirreL SQL Client	3.7.1	1138	71,626
KeyStore Explorer	5.1	384	83,144
DigiExtractor	2.5.2	80	15,668
Angry IP Scanner	3.0	270	19,965
Plugfy	0.6	28	2,337
Matte	1.7	603	52,067
sMeta	1.0.3	222	30,843
xena	6.1.0	1975	61,526
pmd	4.3.x	800	82,885
checkstyle	6.2.0	277	41,104
JDistlib	0.3.8	78	32,081
JCLEC	4-base	311	37,575
Java graphplan	1.0	50	,1049
Mpxj	4.7	553	261,971
Apeiron	2.92	62	8,908
FullSync	0.10.2	169	24,323
OmegaT	3.1.8	716	121,909
Lucene	3.0.0	606	81,611
Gantt project	2.0.10	621	66,540
JFreechart	1.0.X	499	206,559
JHotDraw	5.2	151	17,807

E. NEW TOOL-HUMAN JOINT STUDY (S2) DESIGN

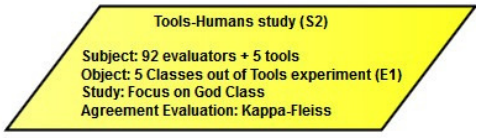


FIGURE 6: Description of human-tool experiment (S2).

The study of S2 was performed on the basis of the information available from E2 and R1. As can be seen in Figure 6 regarding S2, the set of subjects is made up of the five detection tools in R1 plus the 92 human evaluators of E2; while the set of objects is made up of the five selected classes in E2. In this study, our attention is focused on *God Class* detection.

[2]https://citius.usc.es/investigacion/datasets/project-nominal-information

V. RESULTS ANALYSIS

In this section, we introduce the results of our experiments and studies to answer the research questions. In order to organise the results presentation, we introduce a descriptive analysis of the experiments and the results that answer the research questions as follows: a description of the tools experiment (E1) and the replication study (R1), with the answers of RQ1 and RQ2. After that, the descriptive analysis of the human experiment (E2) and answers of RQ3, RQ6, RQ6a, and RQ7 concerning the comparison between human evaluators, followed by answers of RQ4, RQ5, and RQ6b which affect the comparison between tools and humans (Tool-Human studies S1, S2).

A. TOOL EXPERIMENT (E1) RESULTS

As mentioned above, we conducted an exploratory experiment E1 to detect *God Class* and *Feature Envy* first of all, then we replicate the experiment E1 as R1 to detect *God Class*, taking into account the lessons learned from the exploratory experiment E1.

Six tools were used in E1: inFusion, inCode, JDeodorant, Together, iPlasma, and JSmellSensor. They were used to examine 533 classes of the selected project (Apache Lucene 3.1.0). Table 6 shows the number of classes that each tool detected as *God Class*, *Feature Envy* or both (in the same class) in the Apache Lucene 3.1.0 project. At first glance, from the table, there seems to be no obvious agreement between the tools. Each tool obtained different results for the same smell. However, the tools were close to each other in the number of detected smells, with the exception of JDeodorant. Also, there is a consistency in the results of iPlasma, inFusion and inCode, as is discussed later in Section VI-A.

TABLE 6: Number of classes detected as God Class, Feature Envy, and both for each tool in the tool experiment E1.

Tool/Smell	*God Class*	*Feature Envy*	Both
inFusion v.1.8.4	6	3	1
inCode v.2.0.7	6	3	1
JDeodorant v.5.0.13	79	31	4
Together v.12.6.0	14	6	4
iPlasma v.6.1	12	16	2
JSmellSensor	13	8	1

RQ1 Is there any agreement between the selected tools in *God Class* and *Feature EnvyDesign Smell* detection?

To answer the research question RQ1, we recall the null hypothesis that we studied in two cases regarding each *Design Smell*, *God Class* (GC) and *Feature Envy* (FE):

H_0^{GC}: The detection tools do not agree in identifying *God Class* in the classes of the Apache Lucene 3.1.0 project.

H_0^{FE}: The detection tools do not agree in identifying *Feature Envy* in the classes of the Apache Lucene 3.1.0 project.

Both cases assume the null hypothesis to be "There is no agreement between tools". To reject the null hypothesis, we use the significance level $p - value \leq 0.05$. That is to say, if the obtained *p-value* is less than or equal to 0.05, the test is significant and the null hypothesis is rejected. Therefore, in this case, we use the interpretation of the *Kappa-Fleiss* (see Table 1) to identify the degree of agreement. On the other hand, if the *p-value* is greater than 0.05, we cannot reject the null hypothesis. Not rejecting the null hypothesis means we cannot reject the possibility of no agreement. Therefore, according to the data it can be interpreted as the tools not agreeing on detection.

Table 7 shows the obtained results regarding the agreement between detection tools in the first experiment E1. As can be seen, the test results are significant for both cases, *God Class* and *Feature Envy*. In both cases, the *p-values* are lower than the significance level, so H_0^{GC} and H_0^{FE} can be rejected. It could be said that there is agreement between detection tools when detecting the said *Design Smells*. Applying the interpretation of the *Kappa-Fleiss* coefficient, a **poor agreement** between tools is indicated in both cases.

TABLE 7: Results of *Kappa-Fleiss* tests when studying the degree of agreement between the 6 tools selected for E1 when the project Apache Lucene 3.1.0 is analysed.

Design Smell	p-value	*Kappa-Fleiss*	Interpretation
Feature Envy (FE)	0.00601	0.0709	Poor agreement
God Class (GC)	2.43e-13	0.189	Poor agreement

RQ2 Which of the selected tools coincide more closely when detecting in *God Class* and *Feature EnvyDesign Smell* detection?

To analyze RQ2, we have studied the degree of agreement between the possible pairs of the six selected tools ($C_{6,2} = 15$ comparisons), when detecting *God Class* and *Feature Envy*. Table 8 summarises the degree of agreement between each possible pair of tools. The grey cells are indicators of *p-values* greater than 0.05. In these cases, the null hypothesis cannot be rejected, so the tests are not significant. Therefore, it can be assumed that there is no agreement between these evaluators in which the agreement is worse than if it had been done randomly. In all the remaining cases, the null hypothesis is rejected and it is assumed there is a degree of agreement between evaluators, taking into account the *Kappa-Fleiss* value, according to the interpretation in Table 1.

There is a very good agreement (perfect) in the inCode and inFusion pair, since the results were identical, as shown in Table 6. Also, this pair with iPlasma have a very good level of agreement. This result is interesting, since the agreement is very good, despite iPlasma detected a number of classes with *Design Smells* different from the number of classes detected by the pair of (inCode, inFusion) shown in Table 6.

In *God Class* detection, we can observe a **weak (or fair) agreement** has been obtained between the JSmellSensor tool and the group of inCode, inFusion, and iPlasma. It is interesting that the JSmellSensor results, after introducing extra knowledge in order to obtain better results in terms of false positives and false negatives, have been different from those of JDeodorant, but still remain close to inCode (and its related group), although the degree of agreement is weak (or fair) according to *Kappa-Fleiss* interpretation.

Regarding the degree of agreement between the pairs of tools when detecting *Feature Envy* (Table 8), the best agreement (very good or perfect) can be observed again between the pairs of inCode, inFusion, and iPlasma. However, in this case, we detect a weak (fair) agreement between Together and this group of tools. The remaining tools (JSmellSensor, JDeodorant) are distant from this group and from each other. In the case of JSmellSensor, it is particularly interesting that, having inCode and JDeodorant as experts for supplying the initial data, and once the learning algorithm had been trained, the tool returns quite different results from both when detecting *Feature Envy*.

In the light of these results, we decided to perform a new experiment (R1) to study the agreement between tools, as a replication of E1. R1 does not include the entire group formed by inFusion, inCode, and iPlasma; we selected iPlasma as being representative of them.

Before concluding E1, we evaluated the agreement of the remaining three tools (JDeodorant, Together, and JSmellSensor) and obtained *p-values* of 0.782 and 0.95 for the *God Class* and *Feature Envy* detection, respectively. Therefore, the null hypothesis cannot be rejected, and we concluded that there is no agreement between these tools, because the concordance in both cases is worse than if it had been done randomly.

B. E1 REPLICATION (R1) RESULTS

Taking the following facts into account: *i)* the results of E1 show several *p-values* indicating non-significant results;

TABLE 8: Summary of degree of concordance (agreement) between possible pairs of tools when detecting GC and FE in E1.

Pair agreement		inCode	inFusion	iPlasma	JDeodorant	JSmellSensor
Together	GC	**p: 0.528**	**p: 0.528**	**p: 0.528**	**p: 0.211**	**p: 1**
	FE	p : 0.00937 K : 0.26 weak	p : 0.00937 K : 0.26 weak	p : 0.00937 K : 0.26 weak	**p:1**	**p:0.561**
InCode	GC		p : 0 K : 1 very good	p : 0 K : 1 very good	**p:0.965**	p : 0.000442 K : 0.351 weak
	FE		p : 0 K : 1 very good	p : 0 K : 1 very good	**p:0.417**	**p:0.757**
inFusion	GC			p : 0 K : 1 very good	**p:0.965**	p : 0.000442 K : 0.351 weak
	FE			p : 0 K : 1 very good	**p:0.417**	**p:0.757**
iPlasma	GC				**p: 0.965**	p : 0.000442 K : 0.351 weak
	FE				**p:0.417**	**p:0.757**
JDeodorant	GC					**p: 0.211**
	FE					**p: 0.91**

ii) the finding of the coincidental group formed by inCode, inFusion and iPlasma; and *iii)* the experimental character of our JSmellSensor tool; we decided to include in this replication R1 JDeodorant, Together and iPlasma, as well as the other tools (DÉCOR and PMD) presented in Table 4, which were, at the moment of designing R1, some of the most cited tools in the state of the art.

The replication study (R1) of the tools experiment (E1) maintains the same goals and research questions as E1. We decided to work with a large dataset in order to overcome the cases of non-significant results (several p-values greater than 0.05). As mentioned, due to the large size of the dataset in the experiment, which is formed by 12587 classes from 24 projects, we decided to focus only on *God Class*. The characterisation of this dataset is shown in Table 5. The dataset includes the results of detecting *God Class* with the five tools included in this experiment.

Table 9 shows the number of *God Class* detected with each tool. The differences in the number of *God Class* detected can be easily appreciated. Regarding the number of detected classes, the numbers of PMD and iPlasma are close to each other, but the important fact here is the list of classes reported. Due to the dimensions of the dataset, in order to analyse possible coincidences and inclusion relationships between tools, *i.e.* whether a subset of classes detected by a tool is contained in a subset of classes detected by another tool, we performed an FCA study (see Section II-D). We have used the tool FcaBedrock (see [44]) to create the formal context. Once the context had been created, the tool ConExp[3] was used to obtain the Galois lattice.

Figure 7 shows the result of constructing the Galois lattice from a formal context, where the objects are classes and the attributes are tools. The context is defined by indicating that an object (a class in the dataset) has an attribute for each tool, (e.g. PMD) which detects such a class as *God Class* (e.g. PMD detects a class as *God Class*, then the class is marked as having the attribute PMD in the formal context).

[3] http://conexp.sourceforge.net/

TABLE 9: Number of *God Class* detected by each tool in the replication study R1 out of a total of 12587 classes from the 24 projects.

ToolsNoC	GC	No GC
Together	159	12428
DÉCOR	321	12266
PMD	559	12028
iPlasma	564	12023
JDeodorant	1235	11352

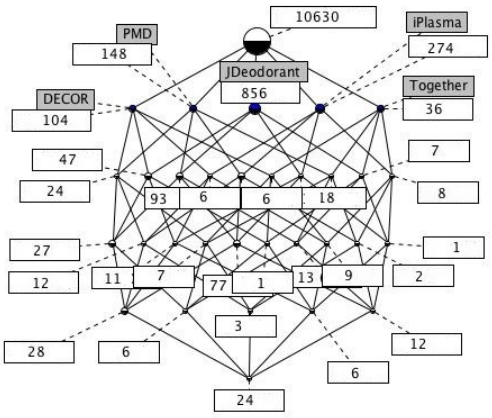


FIGURE 7: Galois lattice obtained by Formal Concept Analysis (FCA).

The construction of a Galois lattice guarantees that each attribute is introduced in a single point (node). Therefore, each node in the lattice represents the objects (classes in the dataset) with the same attributes (tools), *i.e.* the classes detected by a common set of tools given by the attributes.

Hence, a hierarchical partial order is established. Nodes in the lower levels of the lattice, which are connected to higher level nodes, "inherit" the attributes and introduce new attributes. In this way, the node in the lowest position represents the set of classes detected as *God Class* by the five tools, while the node in the highest position represents the set of classes that none of the tools detect as *God Class*. Another fact that can be observed is that all the tools are siblings (in the second level of nodes), hence no tool's result includes another.

As can be seen in Figure 7, only 24 classes are detected by all the tools as a *God Class* (the lowest node in the graph). Other examples of the data observed in the lattice are the classes that are detected by only one tool (on the second level at the top). For example, 856 classes were only detected by JDeodorant, 104 by DÉCOR, 148 by PMD, 274 by iPlasma and 36 by Together. In addition, the next level shows classes detected simultaneously by two tools and none others; for instance, 47 classes were detected by JDeodorant and DÉCOR but no other tool detected them as *God Class*; while Together and iPlasma agree in detecting 8 classes that no other tool detected as *God Class*, etc.

Table 10 shows the results obtained of studying the agreement between the five selected detection tools in order to answer RQ1. As can be seen, the *p-value* shows a significant result and H_0^{GC} can therefore be rejected. According to the *Kappa-Fleiss* interpretation, the degree of agreement between the tools when detecting *God Class* is **weak**. On the other hand, the results of studying the agreement between possible pairs of tools ($C_{5,2}$=10 comparisons) are shown in Table 11, in order to answer RQ2. The same study was conducted using Cohen's Kappa, which is specific for 2 evaluators only and almost identical results have been obtained. Then, we decided to report the analysis based on the *Kappa-Fleiss* coefficient in order to use the same indicator as in the rest of the experiments and studies. Notice that each pair analysis has a significant result according to *p-values*. As can be seen, a **poor or weak agreement** between the different pairs occurs.

TABLE 10: Results for p-value and *Kappa-Fleiss*, when studying the agreement of the 5 selected tools in R1.

Design Smell	p-value	*Kappa-Fleiss*	Interpretation
God Class (GC)	0	0.202	poor agreement (weak)

C. HUMAN EXPERIMENT (E2) RESULTS

Descriptive analysis

In this experiment (based on an online survey as described in Section IV-B), the number of respondents was 93, of which the number of valid responses was 92. Therefore, we have 92 subjects as human evaluators for the five selected classes. The evaluators classified the classes as having a *Design Smell* (*God Class* or *Feature Envy* or both) or not.

As the 92 subjects are dispersed over different areas (20 countries), and the number of subjects per country is not enough to conduct country based analyses, we decided to analyse them by grouping them in geographical areas based on continents. Hence, we group respondents in five different geographical areas: Europe, Asia, America, Africa and Oceania. The distribution of responses over the continents is shown in Table 12. The highest number of answers was obtained from Asia and Europe, while the lowest was obtained from Africa and Oceania.

TABLE 11: Summary of the results of analysing the agreement between possible pairs of tools when detecting *God Class* in R1

Pair agreement	iPlasma	JDeodorant	PMD	DÉCOR
Together	$p:0$ $K:0.156$ poor	$p:8.74e-08$ $K:0.0477$ poor	$p:0$ $K:0.214$ weak	$p:0$ $K:0.244$ weak
iPlasma		$p:0$ $K:0.145$ poor	$p:0$ $K:0.369$ weak	$p:0$ $K:0.195$ poor
JDeodorant			$p:0$ $K:0.259$ weak	$p:0$ $K:0.14$ poor
PMD				$p:0$ $K:0.272$ weak

TABLE 12: Distribution of subjects by geographical area (based on continents).

Asia	Europe	America	Africa	Oceania
60	23	6	2	1

On the other hand, the distribution of subjects by the work activity is shown in Table 13, in which Group 1 (G1) represents the highest frequency and Group 4 (G4) the lowest. When we analysed the responses obtained in Group 5 (G5) "Other", we found the following working activities: "System admin", "Quality Consultant", "Engineer" and some "Msc. Student" or "Student". This motivated us to perform further analyses to classify this variable into two groups: "Industry" and "Academia", in which G1 and Group 2 (G2) are classified into "Academia", Group 3 (G3) and G4 are classified into "Industry" and G5 is divided, the case of students response being classified in "Academia" and the remaining cases in "Industry".

TABLE 13: Distribution of subjects by work activity.

G1	G2	G3	G4	G5
35	13	22	7	15

G1 Professor, Lecturer, Instructor
G2 Researcher
G3 Programmer
G4 Software Architect
G5 Other

Table 14 presents a joint distribution of subjects based on their work activity and experience. The group with the highest frequency was subjects from Academia with an experi-

ence between 5 and 10 years, while the lowest frequency was subjects from Industry with less than 5 years of experience, the same as subjects from Academia with more than 10 years of experience. When looking at the marginal distributions, the majority of the subjects were from Academia rather than Industry. In general, the group of subjects with experience between 5 and 10 years was the highest in frequency (37 subjects) and the other two groups were almost identical with 28 and 27 subjects each.

Table 15 shows three distributions of frequencies regarding the level of expertise in Writing Object-Oriented (OO) code, Reviewing OO code and Experience in *Design Smells* (in the text of the online survey we used the term "Code Smell" as this is more popular among developers). These three factors in the background of the respondents are closely related with *Design Smell* detection and must be taken into account when analysing the results. As can be seen, the highest number of subjects is concentrated in the intermediate level of expertise, regardless of the background factor considered.

Tables 16 and 17 present the results obtained regarding the subjects' effort on *Design Smell* detection. The effort is evaluated from two perspectives: time spent and detection difficulty. Table 16 shows the distribution of the frequencies of time employed by respondents in detecting *Design Smells*. The respondents were informed that they would need 25 minutes on average to perform the task, and approximately 80% of them took less than 25 minutes. Regarding the difficulty of the task, Table 17 shows that a vast majority of the subjects (at least 79%) consider that reviewing third party code, as well as detecting *Design Smells*, range from medium to very difficult when asked to measure the difficulty of the task.

Research questions analysis

In order to answer the research questions regarding the human evaluation experiment (E2), we should remember that the main hypothesis related to this experiment is Hypothesis 1.b: "There is no agreement between human evaluators when they detect *God Class* and *Feature EnvyDesign Smells*". The hypothesis test regarding concordance is based on the *Kappa-Fleiss* statistic with a significance level $p - value \leq 0.05$. Null hypothesis is stated as follows:

H_0 There is no agreement between evaluators ($kappa \leq 0$, given $p - value \leq 0.05$).

H_0 is subdivided:

H_0^{GC} There is no agreement between evaluators when detecting *God Class*.

H_0^{FE} There is no agreement between evaluators when detecting *Feature Envy*.

In order to perform this analysis, more than 100 studies were conducted with *Kappa-Fleiss* statistic, with different combinations regarding the evaluators' profile. The research questions relating to this experiment (E2) are: RQ3, RQ6, RQ6a, RQ7.

To obtain more conclusive results of the research questions in the human evaluation experiments (E2), also, we answer the research questions after excluding the 20.7% of respondents that have not experience in Design Smells (None category), as mentioned in Table 15, and on the other hand, after removing the 9.8% of respondents who were too fast, taking less than 5 minutes to answer the survey as shown in Tables 16.

RQ3 What is the degree of agreement between human evaluators in *Design Smell* detection?

Having carried out the inter rater concordance study with the *Kappa-Fleiss* statistic, Table 18 shows the obtained results regarding agreement between subjects on detecting the proposed *Design Smells*. As can be seen, in the *God Class* case, the *p-value* was 0.0565 (greater than 0.05), which indicated not significant results and the null hypothesis (H_0^{GC}) cannot be rejected, and it can be concluded that no agreement is found. Also, in the *Feature Envy* case, the *p-value* is greater than the significance level (0.438), so the result is not significant and the null hypothesis (H_0^{FE}) cannot be rejected; so we can say, there is no agreement in the detection, the concordance level is worse than if the evaluation had been carried out randomly.

After removing the ratio of respondents that have not experience in *Design Smells*, the obtained results showed poor agreement in the *God Class* case, where the *p-value* was 0.0296 (significant), and the *Kappa-Fleiss* value was 0.019. Also, after removing the fast respondents, the obtained results are not significant, in which the *p-values* were 0.306 and 0.245 for Feature envy and *God Class* respectively.

RQ6 How does the degree of experience affect *Design Smell* detection?

RQ6a Is the degree of agreement between human evaluators higher when the group of evaluators has more experience?

The initial explorations in studying agreement by groups indicated that the significance increases if we group the "Inexperienced" (less than 5 years of experience) and the "Experienced" (more than 5 years). To this end, instead of establishing three groups of experience as was asked in the online survey, the data was divided into two groups: less than and more than 5 years of experience.

Table 19 shows results obtained for the *p-value* and the *Kappa-Fleiss* coefficient. The result is only significant when the group of more than 5 years of experience detects *God Class*, but the degree of **agreement is very poor**. In the remaining cases, the *p-values* are not significant ($\not\leq 0.05$), so the null hypothesis cannot be rejected, and we say there is no agreement between evaluators.

After excluding the respondents who have not experience in *Design Smell* and the fast respondents that answered the survey in less than 5 minutes, the result is only significant when the group of more than 5 years of experience detects *God Class*, where the obtained *p-values* were 0.000375 and 0.0362 respectively and the *Kappa-Fleiss* values were 0.0437 and 0.0252 showing a very poor agreement. In the remaining cases, the *p-values* were not significant.

RQ7 How does the background (regarding training, experience and knowledge) and context of evaluators

TABLE 14: Joint distribution of subjects per experience and activity.

Activities/Experience	$years < 5$	$5 \leq years \leq 10$	$years > 10$	Total	%
Industry	11	13	16	40	43.5%
Academia	17	24	11	52	56.5%
Total	28	37	27		

TABLE 15: Frecuency distribution of expertise in writing OO code, reviewing OO code and *Design Smells*.

Expertise/On	Writing OO Code	%	Reviewing OO Code	%	*Design Smells*	%
None	9	9.8%	8	8.7%	19	20.7%
Beginner	14	15.2%	17	18.5%	26	28.3%
Intermediate	44	47.8%	46	50%	35	38%
Expert	25	27.2%	21	22.8%	12	13%

TABLE 16: Distribution of frequencies regarding the time employed in the survey for *Design Smell* detection.

Percentage of evaluators by time intervals in minutes								
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%

TABLE 17: Frequency distributions of the difficulty encountered in reviewing third party code and detecting *Design Smells*.

Percentage of evaluators by task difficulty measurement						
Task/Difficulty	No answer	Very easy	Easy	Medium	Difficult	Very difficult
Code reviewing	4.3%	6.5%	14.1%	51.1%	19.6%	4.3%
Detecting *Design Smells*	4.3%	7.6%	7.6%	50%	22.8%	7.6%

TABLE 18: Results of p-value and *Kappa-Fleiss* when studying agreement between human evaluators in E2.

H_0	p-value	Kappa-Fleiss	Interpretation
H_0^{GC}	0.0565	−0.00536	Not significant.We interpret it as there being no agreement when detecting God Class
H_0^{FE}	**0.438**	0.0132	Not significant. We interpret it as there being no agreement when detecting Feature Envy

TABLE 19: Results of studying agreement in *God Class* and *Feature Envy* detection between evaluators, analysing inexperienced (less than 5 years) vs experienced (more than 5 years) evaluators.

H_0	p-value	Kappa-Fleiss	Interpretation
H_0^{GC}	Inex. **0.215** Exp. 0.00204	Inex. −0.028 Exp. 0.0307	Significant when the more experienced detect God Class but agreement is very poor.
H_0^{FE}	Inex. **0.524** Exp. **0.267**	Inex. −0.0146 Exp. −0.0111	Not significant. We interpret it as there being no agreement when detecting Feature Envy in any case.

affect *Design Smell* detection?

RQ7a Does the work context, geographical area where the developers are from or whether the context is industrial or academic have any effect?

The studies of agreement in *God Class* and *Feature Envy-Design Smell* detection, taking into account the geographical area of the human evaluators, indicated that there was no difference from one area to another, except for Europe as can be seen in Table 20. In fact, all the results (except Europe) are not significant; in all cases (All respondents, After removing inexperience respondents, After removing fast respondents), the obtained *p-values* were quite large (for each area, for each *Design Smell*). The only case in which significant results were obtained was for respondents from Europe when detecting *God Class*. With all respondents case, the *p-value* was close to zero, and the *Kappa-Fleiss* value was 0.18, which is very close to being interpreted as **weak (or fair) agreement**.

The same analysis was repeated excluding the respondents who have not experience in *Design Smells*, in the first place, and, in the second case, removing the fast respondents. Similar results to the analysis conducted with the entire set of respondents were obtained with *p-values* were close to zero, and the *Kappa-Fleiss* values were 0.24 and 0.16 respectively. Only one respondent from Oceania area, so, we cannot compute the degree of agreement between evaluators.

The test carried out to study the impact of the workplace (Academia vs. Industry) on the degree of agreement, when detecting *God Class* and *Feature EnvyDesign Smells*, were

TABLE 20: Results of studying agreement in *God Class* and *Feature Envy* detection taking into account the geographical area of human evaluators.

Respondents	DS	Europe		Africa		Asia		North America	
		P	*K*	*P*	*K*	*P*	*K*	*P*	*K*
All respondents	GC	1.45e-10	0.18	**0.804**	-0.111	**0.192**	-0.014	**0.341**	0.11
	FE	**0.83**	0.006	**0.804**	-0.111	**0.598**	-0.006	**0.739**	-0.039
		P	*K*	*P*	*K*	*P*	*K*	*P*	*K*
Removing Inexperts	GC	7.11e-15	0.24	**0.804**	-0.111	**0.302**	-0.015	**0.341**	0.11
	FE	**0.098**	0.051	**0.804**	-0.111	**0.972**	0.001	**0.768**	-0.042
		P	*K*	*P*	*K*	*P*	*K*	*P*	*K*
Removing Fast Respondents	GC	4.00E-07	0.164	**0.804**	-0.111	**0.187**	-0.018	**0.556**	0.083
	FE	**0.917**	-0.003	**0.804**	-0.111	**0.655**	-0.006	**0.191**	0.185

God Class = GC, Feature Envy = FE, Design Smell = DS, P-value = P, Kappa-Fleiss = K.

all not significant, so no agreement was found. Therefore, it seems that the workplace does not influence the degree of agreement among subjects when detecting *God Class* and *Feature EnvyDesign Smells*.

RQ7 ...

RQ7b Does the evaluator's background (regarding expertise in object-oriented programming, in code reviewing or his/her knowledge level on *Design Smells*) affect the degree of agreement?

In Table 21, the cases of results in grey cells obtained *p-values* greater than 0.05, indicating that we cannot reject the null hypothesis. Therefore, we assume there is no agreement between the evaluators. However, in the other cases, some agreement can be admitted, which can be interpreted by looking on the *Kappa-Fleiss* value interpretation as very poor. As can be seen, in the all respondents case, the agreement occurs in *God Class* detection in the group proficient in the tasks, while in the *Feature Envy* case, agreement occurs in the group of inexperts. It is curious that the same pattern is repeated for the three activities.

Similar results are obtained regarding the *God Class* detection after removing the inexperience respondents (None category 19 of 92) and the fast respondents from the analysis, where poor agreement is found except for reviewing OO code task. While, in the *Feature Envy* detection case, the agreement only occurred in the group of inexperts, in particular, with writing and reviewing OO code.

D. STUDIES S1 AND S2 RESULTS

This section presents the results of the joint studies S1 and S2 that do not introduce new data, but which make analyses with data jointly from E1+E2 and R1+E2. These studies allow the research questions which affect the comparison between human evaluators and detection tools (RQ4, RQ5, RQ6b) to be answered.

RQ4 What is the degree of agreement between human evaluators and tools in *Design Smell* detection?

This question concerning the tool-human joint studies S1 and S2 as are shown in Figure 1 in general and in detail in Figures 4 and 6. The S1 study is related to the data obtained from the E1 and E2 experiments, while the S2 is related to the data from the R1 and E2 experiments. In S1, we compared the results of subjects with the same six tools in E1, which include: Together, JDeodorant, iPlasma, inFusion, inCode and JSensorSmell when detecting *God Class* and *Feature Envy* in the classes of the Apache Lucene project; in particular, the results for the five selected classes proposed to respondents in the online survey. The performed tests indicate that there is **no agreement** between human evaluators and tools in either case, *God Class* or *Feature Envy*. The obtained *p-values* were 0.111 for *God Class* and 0.51 for *Feature Envy*, respectively.

In the second study (S2), we compared the results of subjects with the five selected tools in the replication study R1, these were Together, JDeodorant, iPlasma, DÉCOR and PMD, when detecting *God Class* in the classes of 24 projects, in particular, their results in the five selected classes proposed to respondents in the online survey. The obtained results indicate that there is a **poor agreement** between human evaluators and tools in the *God Class* case. The obtained *p-value* was 0.0359 (less than 0.05), and the *Kappa-Fleiss* value was 0.0137. In the *Feature Envy* case, only three of the tools used in the comparison could detect *Feature Envy* (Together, JDeodorant, iPlasma). The obtained *p-value* was 0.425 (greater than 0.05), which indicated non-significant results, and it can be concluded that no agreement is found between human evaluators and tools.

After removing the inexperience respondents from the question analysis, the obtained results indicate that there is only a poor agreement between human evaluators and tools in the *God Class* case in the second study (S2), where the obtained *p-value* was 0.0213 and the *Kappa-Fleiss* was 0.0188, while in the other case after removing the fast respondents, similar results are obtained, in which no agreements are found.

RQ5 Which tools coincide more with human evaluators in *Design Smell* detection?

When we study the degree of agreement between subjects and each tool in the tool experiment (E1) separately, for detecting *God Class* and *Feature Envy*; the obtained results were not significant in all cases (*p-values* were greater than 0.05), which we interpret as there being **no agreement**. Also, when we study the agreement after removing the inexperience (None Category) and the fast subjects (less than 5 minutes answering) and each tool from the question analysis, the obtained results were not significant in all cases, except for significant results in the *God Class* case with all tools ($p - values = 0.035$, $Kappa - Fleiss = 0.0182$) and we interpret this as a poor agreement.

TABLE 21: Results of studying agreement by groups of subjects classified by expertise regarding different activities closely related to *Design Smell* detection (Writing OO code, Reviewing OO code, *Design Smells*).

All Respondents	Inexperts (None+Beginner)			Proficient (Intermediate+Expert)		
Writing OO code	DS	p	k	DS	p	k
	GC	**0.579**	**0.015**	GC	0	0.117
	FE	0.008	0.078	FE	**0.605**	**0.005**
Reviewing OO code	DS	p	k	DS	p	k
	GC	**0.159**	**0.036**	GC	0.011	0.024
	FE	0.005	0.073	FE	**0.399**	**0.008**
Design Smells	DS	p	k	DS	p	k
	GC	**0.960**	**0.001**	GC	0.006	0.001
	FE	0.527	0.009	FE	**0.101**	**0.022**
After Removing Inexperts (None)	Inexperts (Beginner)			Proficient (Intermediate+Expert)		
Writing OO code	DS	p	k	DS	p	k
	GC	**0.19**	**0.098**	GC	0.004	0.029
	FE	**0.402**	**0.063**	FE	**0.278**	**0.011**
Reviewing OO code	DS	p	k	DS	p	k
	GC	**0.13**	**0.083**	GC	**0.095**	**0.027**
	FE	**0.099**	**0.091**	FE	**0.25**	**0.012**
Design Smells	DS	p	k	DS	p	k
	GC	**0.9**	**-0.003**	GC	0.006	0.037
	FE	**0.113**	**0.039**	FE	**0.101**	**0.022**
After Removing Fast Respondents	Inexperts (None+Beginner)			Proficient (Intermediate+Expert)		
Writing OO code	DS	p	k	DS	p	k
	GC	**0.256**	**0.05**	GC	0.007	0.030
	FE	0.003	0.119	FE	**0.807**	**0.003**
Reviewing OO code	DS	p	k	DS	p	k
	GC	0.047	0.072	GC	0.028	0.026
	FE	0.003	0.108	FE	**0.387**	**0.010**
Design Smells	DS	p	k	DS	p	k
	GC	**0.76**	**-0.006**	GC	0.013	0.043
	FE	**0.941**	**0.001**	FE	**0.385**	**0.015**

In addition, when we study the degree of agreement between subjects and each tool in the replication study (R1) separately, for detecting the same *Design Smells*, we obtained the following results: in the *Feature Envy* case, the obtained *p-values* were 0.347 (greater than 0.05) in all cases, so the results were not significant and we interpret this as **no agreement** is found. In the *God Class* case, the obtained *p-values* are not significant in all cases ($p - value = 0.0617$), which indicates a **no agreement**, except for significant results in JDeodorant case ($Kappa - Fleiss = 0.0255$), a **poor agreement**. Also, when we study the agreement after removing the inexperience and fast respondents from the question analysis, similar results were obtained, in which only the JDeodorant case was significant ($p-value = 0.0237$, $Kappa - Fleiss = 0.0192$) and interpreted as a poor agreement. In general, we concluded that none of the selected detection tools is close to the human evaluators, but all were closer in the *God Class* case versus the *Feature Envy* case.

RQ6 How does the degree of experience affect *Design Smell* detection?

RQ6b Is the degree of agreement with detection tools higher when the group of evaluators has more experience?

We performed a cross study for each tool with different groups of human evaluators, based on the geographical area, the work context (in Academia or Industry), the years of experience in their professional activities, and the degree of expertise in the activities relevant to *Design Smell* detection. The results show that a similar pattern is obtained for each tool. This pattern can be described as follows:

Significant results were obtained that allow the null hypothesis to be rejected. The *Kappa-Fleiss* coefficients were very small and, therefore, agreement is interpreted as poor when the evaluator profile matches:

- Proficient evaluators in the activity of writing OO code and/or in reviewing OO code when detecting *God Class*.
- Inexpert evaluators in the activity of writing OO code and/or in reviewing OO code when detecting *Feature Envy*.
- Proficient evaluators in *Design Smells* when detecting *God Class*.
- Evaluators from Europe when detecting *God Class*.

The remaining cases of evaluators' profiles under study gave non-significant results, which are interpreted as no agreement being found.

VI. DISCUSSION

In this section, we highlight the significant results that answer the research questions of this study from two different contexts: tool and the human evaluation. Then, we address the main threats to the validity of the conducted experiments.

A. FINDINGS

Tool Evaluation Context

The *Design Smell* detection results were varied from one tool to another, as confirmed in the first experiment E1 and its replication R1. The FCA graph in Figure 7 shows the relationships between the detection accomplished by the tools. As can be seen, JDeodorant detected the highest number of *Design Smells* that the remained tools did not detect, while Together has detected the lowest number. These numbers

indicate that the detection technique used in JDeodorant is distinguished from other methods applied in the other tools. The number of detected *Design Smells* by each tool were varied; for example, 856 classes were only detected by JDeodorant, 104 by DÉCOR, 148 by PMD, 274 by iPlasma, and 36 by Together. Only a set of 24 *Design Smells* from the whole group were detected by all tools. The detection results do not mean that all detected smells are confirmed positive, but there exist some false positives *Design Smells*. The differences are indicators of the low degree of agreement between tools. The obtained results prompt the community interested in *Design Smells* detection to involve other factors that can assist and contribute in detecting the true positive. Therefore, improving the degree of agreement between detection tools. The results have been interpreted as having a poor degree of agreement or no agreement at all. In the light of the *Kappa* results, the lack of agreement between detection tools was not surprising. Different approaches and techniques have been proposed to detect *Design Smells*, which later developed into detection tools. Also, the inconsistency in understanding with respect to the precise definition (specification) of a *Design Smell*, and specifically, in our case, *God Class* and *Feature Envy*, leads to inconsistencies in the detection results.

The degree of agreement between possible pairs of tools was also studied. The results show a very good degree of agreement between the group of inCode, inFusion and iPlasma when they detect *God Class* and *Feature EnvyDesign Smells*. The value of the *Kappa-Fleiss* coefficient is (Kappa = 1) between each pair of these three tools. The pair of inCode and inFusion formed a special case because their results were identical as regards the number of detected *Design Smells* and the names of smelly classes in the first experiment E1. By looking at the origin of this pair of tools (inCode, inFusion), we find they were developed by the same institution, *i.e.*, they have a common origin. Also, this pair, with iPlasma, have a very good level of agreement in all cases of *Design Smell* detection, despite the difference in the number of classes detected. The reason we find is that the iPlasma tool is the origin of inCode and inFusion. iPlasma was developed by the LOOSE research group, led by Dr. Radu Marinescu at the Politécnica University of Timișoara in Romania. On the other hand, inCode and inFusion were tools developed by the intooitus company[4], which was a startup that originally arose from the same group (LOOSE). Therefore, these tools and their techniques have iPlasma as their base.

Human Evaluation Context

The number of respondents who participated in the survey was 93; we consider 92 valid responses was a good result, compared with those of similar studies described in the related work (see Section VII). Most of the respondents required less than the specified time to detect the smells. Given that the respondents should detect *Design Smells* in five classes, there is an average of 6 minutes per class to analyse the presence of a *Design Smell*. This confirms the fact that detecting *Design Smells* in a large project without the assistance of tools is not feasible. In our opinion, the reliability of the responses was very good because the survey was distributed randomly (except for the case of experts in *Design Smells*), and the subjects were diverse in background and experience.

The obtained *p-values* and *Kappa-Fleiss* results show a lack of agreement between subjects in general, and were found to be very poor in the cases showing some degree of concordance. Several factors may affect the degree of agreement, depending on how the respondents look at *Design Smell* problems, understand the definition of smells, or what their background is. In our case, in general, the degree of agreement was varied from one smell to another. In the *Feature Envy* case, no agreement was found, while in the *God Class* case agreement was found to be very close to poor. One of the possible explanations may be that *God ClassDesign Smell* concerns the size and cyclomatic complexity concepts, and the evaluators tend to understand these concepts better than others. On the other hand, the *Feature EnvyDesign Smell* concerns the cohesion and coupling concepts in object-oriented code, as well as with the violation of GRASP patterns, which are potentially confusing to understand compared with *God Class* related concepts.

We studied separately the impact on agreement in detection of evaluators' background regarding the geographical area, the work context (in Academia or Industry), the years of experience in their professional activities, and the degree of expertise in the activities we consider relevant to *Design Smell* detection, such as writing and reviewing Object-Oriented code and knowledge in *Design Smell* topic. The results show that the evaluators belonging to the academic or industrial environment does not influence degree of agreement, despite the fact that the people working in the software industry are closer to source code problems. Regarding the geographical area, the only case where we found a fair (weak) degree of agreement between the evaluators was in participants from Europe. The justification for this case may be due to the technique used to publish the survey, as we mentioned before, such as contacting some companies in European countries, or submitting the survey to the list of collected emails of the authors with articles related to *Design Smells* because most of them were from Europe. This submission was necessary in order to guarantee the presence of subjects with a knowledge level regarding *Design Smells* higher than "Intermediate".

When expertise is taken into account, we found a poor degree of agreement between the most experienced and proficient evaluators (more than 5 years, Intermediate + Expert) when detecting *God Class*; while when detecting *Feature Envy* the same degree of agreement was found in the group of inexperienced and inexpert evaluators (less than 5 years, None + Beginner). The same pattern is repeated for the three types of activities (reviewing OO, writing OO, knowledge in *Design Smells*). We assume that the most experienced group has developed the aspects related to size and complexity man-

[4]https://www.intooitus.com/

agement better in comparison to the inexperienced group. Subjects with beginner experience tend to produce longer and more complex methods and classes. Their inexperience let them to think that they could manage the sizes that the experts considered too large. Additionally, we assume that the youngest respondents were formed better in cohesion and coupling concepts, as well as in GRASP patterns, Design Principles, etc., in their Software Engineering and Programming courses. They were aware of these problems, hence they have some agreement (poor) in *Feature Envy* detection.

VII. RELATED WORK

Several studies have focused on automatically detecting *Design Smells*. However, a limited number of studies have attempted to tackle this problem with empirical studies to analyse subjective evaluation by persons, and compared it with detection tools. In this section, we address the studies in the literature which have focused on analysing *Design Smell* detection based on two aspects of evaluation: tool-based and human-based evaluation.

A. TOOL-BASED EVALUATION

[21] described the experience of using six *Design Smell* detection tools: JDeodorant, PMD, iPlasma, inFusion, Stench Blossom, and DÉCOR. They conducted a comparative study of tools based on the programming language the tool can analyse, the *Design Smells* it can detect, whether the tool can additionally perform refactoring, as well as usability issues. These usability issues include some concerns regarding tool integration into a development environment or acting as a separate tool, and whether the tool provides links to the location of the *Design Smells* in code. Several versions of the same open source project (GanttProject) were used to detect *Design Smells*.

[17] also conducted a comparative study between four different *Design Smell* detection tools: JDeodorant, inFusion, PMD and Checkstyle. Using these tools they analysed a single open source project written in Java (GanttProject) to detect six *Design Smells* including *Feature Envy* and *God Class*. In this case they checked the agreement between tools with a *Kappa-Fleiss* statistical test. The results show there is no agreement between these tools. The authors' analysis was that it can be explained by the different techniques used in each tool to detect *Design Smells*, and by the differences in the metrics threshold in those cases where the tool used a metrics based technique.

[22], in 2013, performed a comparative study between JDeodorant and inCode tools in *God Class* and *Feature EnvyDesign Smell* detection. This study used the same *Design Smells* that we have used in our study but with two detection tools. The authors analysed two versions of a multimedia application named Xtreme Media Player. The comparison was based only on the number of *Design Smells* detected. The degree of agreement between evaluators was not studied.

[35] published a comparative study of *Code Smell* detection tools, but it did not include an analysis of the degree of agreement between evaluators. In this case, and in the rest of the works presented in this subsection, the evaluators were always tools.

[23] presented a review of Bad Smells' detection tools as a systematic literature review. Authors found 84 detection tools and report on 29 of these tools based on availability to download and install. A comparison of inFusion, JDeodorant, PMD, and JSpIRIT with respect to two Bad Smells: Large Class and Long Method. To accomplish the comparison two software projects were used.

[15] conducted an evaluation of inFusion, JDeodorant, PMD, and JSpIRIT and compared them. The study was focused on God Class, God Method, and Feature Envy. Agreement was calculated among tools and between pairs of tools using two software projects along different versions of them.

B. HUMAN-BASED EVALUATION

[25] described two experiments conducted over 2003 and 2004. In the first experiment, they carried out a survey asking a group of evaluators about the presence of three method level *Design Smells* (*Long Method*, *Long Parameter List* and *Feature Envy*) in the source code. Furthermore, the evaluators were asked whether they thought the methods should be refactored to remove the *Design Smells* or not. In the second experiment, the evaluators were not asked to detect the smells, but only whether some methods should be refactored or not. The first experiment was conducted with 46 evaluators and the second with 36. All evaluators were master degree students. 50% of them had some experience as developers in the software industry. The results of the first experiment show a high degree of agreement between evaluators detecting *Long Method* and *Long Parameter List* in contrast to the poor agreement detecting *Feature Envy*, as well as in deciding whether the method should be refactored. The single question of the second experiment regarding method refactoring also yielded poor/weak agreement.

[24] carried out a questionnaire based experiment in a small software development company in Finland in 2004. The company had 18 developers and 12 of them answered the survey, which included questions about the presence of 23 *Design Smells* in different modules developed in the company itself. Each developer evaluated three modules on average, obtaining four evaluations for each module. Despite the few data available, the researchers concluded that leader developers (have most experience and knowledge) detect more structural *Design Smells*, while regular developers detect more smells such as *Duplicate Code* or *Dead Code*. They found much subjectivity in the evaluation. Three *Design Smells* (*Large Class*, *Long Parameter List* and *Duplicate Code*) were selected for a study conducted using source code metrics and they found that the subjective developers' evaluation did not correlate with the metrics.

[27] conducted a study that took into account the opinions of 2 experts, who were external to an organisation, and 6 developers chosen from a selection process, in order to assign

them the maintenance of two software systems. These 8 evaluators were asked about the maintainability of the systems before the beginning of the maintenance process and during the process as well. The authors describe the identification of 13 important factors that influence maintainability. Both the external experts and the developers in charge of maintenance agreed on 9 of these factors. The researchers attempted to correlate this information with the *Design Smell* detection used as a maintainability indicator. They obtained partial correlations in the case of some *Design Smells*. Then, they analysed which of these *Design Smells* (such as *God Class*, *God Method*, *Lazy Class*, *Message Chain*, *Long Parameter List*, *Duplicate Code*, *Switch statements*, *Feature Envy*, *Shotgun Surgery*, *ISP Violation*) can be automatically detected by existing tools in order to give a quantitative view of maintainability, which is essentially a qualitative issue.

Again, [28] designed a survey to explore whether *Design Smells* were important to developers or not. They also wanted to know, in the cases where *Design Smells* were not considered important, whether this was due to the irrelevance of the concept, developers' lack of knowledge, or lack of appropriate tools to detect and remove the *Design Smells*. The researchers surveyed 85 professional developers who were recruited through a web portal for job offers to freelances. The results showed that 32% of the respondents had not heard of *Design Smells* or similar terms before. 22% had heard about them from some blogs or discussion forums, but they were not sure what they really were. This means that there was a lack of knowledge about *Design Smells* in more than 50% of the participants. On the other hand, 21% knew the concept of *Design Smells*, but had never applied it in practice. The remaining 18% had a good or strong knowledge of *Design Smells* and applied it in their daily activities. Just 2 out of the 85 respondents declared that they used any *Design Smell* detection tool. As part of the same study, the authors elaborated a ranking of the best known *Design Smells*. They concluded, on the one hand, that training and dissemination of *Design Smell* concepts and related activities are required; and, on the other hand, that *Design Smell* detection tool providers should improve some usability issues, including being ready-to-use but configurable at the same time.

[26] published a study with 34 participants who were 15 master degree students, 10 developers working on open source projects, and 9 developers working in the software industry. The main goal of the study was to ascertain to what extent developers understand *Design Smells* as problems that should be solved. Moreover, they wanted to check which *Design Smells* are considered the most harmful. The results showed that several *Design Smells* were not acknowledged as problems to solve. Nevertheless, *Design Smells* related to size and complexity such as *God Class*, were always seen as problems. *God Class* was, in fact, top in the most harmful ranking. Also, the case of *Feature Envy* was interesting because it was one of those with the greatest variability in respondents' answers. The authors concluded in this case that it was probably due to a misunderstanding of or noncompliance with Object Oriented principles.

To overcome the shortcomings of previous works and their findings, in this paper, we conducted not just tool comparisons but also analysed the degree of agreement between them. We not only carried out a human evaluation, but we also did a joint study including humans and tools. For this purpose, the proposed work differs from the previous works in the following aspects:

- A large set of 24 open-source software formed by 12,587 classes has been used to conduct the experiments.
- Two different sets of design smell detection tools have been used to automatically detect the God Class and Feature Envy.
- A large set of human evaluators (92 evaluators) have participated in the human experiment.
- The results have been compared by replicating the experiments on another dataset.

By comparing the results with previous studies, this study confirms the literature conclusions concerning the degree of agreement between the different types of evaluators (tools, human), which denotes no agreement exists between evaluators, and if it exists, it's very poor. Moreover, the results showed that the degree of agreement between the various types of evaluators could be improved if other information related to software context and human evaluators should be taken into account during detection tools development.

VIII. THREATS TO VALIDITY

It is necessary to consider the potential threats which may affect the validity of the studies results, that are: construct validity, internal validity, external validity and conclusion validity according to [45]. In the following, those issues that may have threatened the validity of the studies presented in this paper are described.

Construct validity is concerned with the relationship between theory and observation. The selected set of detection tools that we used to detect *God Class* and *Feature Envy-Design Smells* are considered a threat to construct validity in this study. Despite the fact that these tools have been used widely in the state of the art, other detection tools could be used to confirm the results. We managed this threat by selecting the tools based on restricted criteria, such as common in detecting the selected smells and in analysing the same version of Java source code, and others explained in Sections IV-A. Another threat to construct validity is related to the chosen *Design Smells*. To overcome this threat, we selected the most cited *Design Smells* in the literature. These smells were common to the group of selected tools. Furthermore, the smells belong to different scopes (class and method levels) and categories in order to cover a wide spectrum despite dealing with a reduced set of *Design Smells*.

External validity is the degree to which the results can be generalised and transferred to other situations. The most significant threat arose from the nature of the analysed project. All the projects were open source and written in Java so that

the selected detection tools could be used on all the projects. Thus, the results only can be generalised to open source projects implemented in the same language. In addition, the generalisation of the results is limited due to the nature of the chosen *Design Smells* and the tools that are capable of detecting those *Design Smells*. Regarding the human evaluation, the generalisation could be limited to professionals with similar profiles. To overcome this limitation we try to reach a variety of them.

Internal validity is related to any negative effects on the experiment design. The main threat to internal validity relates to the respondents' reliability because the survey is published on the web (online survey). So, we have no procedure to control the survey, and we do not know if the respondents take the information concerning the *God Class* and *Feature Envy* definitions seriously before answering the survey. For example, the expected average time to complete the survey was 25 minutes, yet we found 9.8% of respondents took less than 4 minutes. This ratio could indicate how some of the respondents tackled the questions, but it is not conclusive as it cannot be precisely assured how much time a person spends in detecting the presence of a given smell in a given class.

The threat to internal validity can also be explained in terms of the extent to which the dependent variables variation can be explained by rare independent variables, which means that some odd variables which can influence dependent variables are kept under control when designing and performing the experiments. In E2, the human evaluators study, the subjects answered an online survey so threats to validity due to fatigue or tedium or carry-over were not controlled. The survey was designed in order to reduce as much as possible the average time needed to complete it while avoiding fatigue or carry over. In addition, one of the participants was eliminated from the study due to anomalies in the answers, in order to avoid distortion in the results of the experiments. Further studies can accomplish more controlled replications of these experiments.

Conclusion validity concerns those aspects that might affect the ability to draw a correct conclusion, such as the data collection, the reliability of the measurement, and the validity of the statistical tests. We have explicitly mentioned and explained all these aspects along the presentation of the design of the experiments. Further replications grouping tools based on the same techniques, more homogeneous profile of the human evaluators, larger datasets, etc., could confirm the results of these experiments.

IX. CONCLUSIONS AND FUTURE WORK

In this work, we performed a set of experiments related to identifying the degree of agreement between different types of evaluators (tools and humans); on detecting two types of attractive *Design Smells*: *God Class* and *Feature Envy*.

By looking at the obtained results in the tool evaluation experiments, we can conclude that our suspicions are confirmed about the poor or weak, or even nonexistent, degree of agreement between different detection tools.

In the light of the obtained results, we have detected that it is necessary, on the one hand, to have tools capable of detecting a broad spectrum of *Design Smells*; while on the other hand, it is necessary for such tools to be easily compared, according to a "standard pattern". All these facilities will encourage the software industry to adopt some of these tools, in similar way to the adoption of the refactoring tools. We should work on the integration of these tools with the current tendencies of project automation, which permits detection reports to be obtained automatically without human intervention. It is also possible that each tool could be more sensitive to certain characteristics of the projects. With this hypothesis, we have elaborated a study to analyse some project characteristics that can influence how the tools detect *Design Smells*.

Regarding the results obtained for the human evaluation experiment, in which subjects with different profiles and background, identify *God Class* and *Feature EnvyDesign Smells* in the five classes, we have studied the degree of agreement between them in smells detection. In addition, we have studied the degree of agreement between human subjects and the two sets of selected tools in the tool experiment (E1) and the replication study (R1), as mentioned before. The main conclusion we have been able to reach is that we have confirmed our suspicion that there is no agreement between human evaluators in general and between human evaluators and detection tools. In the reduced cases where we found some agreement, the degree is poor or weak, very poor in almost in all cases.

We detected an evaluator profile that yields better results in the degree of agreement, in some cases, between the evaluators and detection tools. This profile indicates that the experienced developers have a better coincidence in the questions concerning the size and complexity, while the developers with little experience have more recent knowledge about the principles and patterns of object-oriented design and then have a better coincidence in the questions concerning violations of those principles and patterns.

The study clearly shows that more training on *Design Smells* is required. In our study, 49% of respondents considered themselves without experience or beginners in *Design Smells*. Only 13% indicated that they considered themselves an expert on the subject and this taking into consideration the fact that we made an effort to reach experts in the subject via a compilation of a list of emails of the authors of articles relating to *Design Smell* detection requesting their participation.

The observed results allow us to elaborate some recommendations, which may contribute to the adoption of *Design Smell* detection techniques in the software industry:

- Training on *Design Smells* should be incorporated into the studies where the future software developers and researchers in software engineering are prepared.
- Consensus benchmarks should be established to assure that the detection tools meet the minimum to guarantee their usefulness in detecting *Design Smells*.

- It is necessary to have the opinion of professional experts in those activities that relate to *Design Smell* detection, such as writing code, reviewing or reading code developed by others, and in *Design Smell* knowledge at the moment of validating the results of the detection tools.

As future work, we believe it would be interesting, as mentioned above, to conduct replications of this set of experiments with different groups of tools, *Design Smells*, subjects of more homogenous profile and high experience in the aspects related to *Design Smell* detection.

REFERENCES

[1] Arthur J. Riel. Object-Oriented Design Heuristics. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 4 1996.

[2] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. Refactoring: Improving the Design of Existing Code. Object Technology Series. Addison-Wesley, 6 1999.

[3] William J. Brown, Raphael C. Malveau, Hays W. McCormick III, and Thomas J. Mowbray. AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. John Wiley and Sons, 3 1998.

[4] Thierry Miceli, Houari A. Sahraoui, and Robert Godin. A metric based technique for design flaws detection and correction. In 14^{th} IEEE international conference on Automated software engineering, pages 307–310, Washington, DC, USA, 1999. IEEE Computer Society.

[5] Radu Marinescu. Detecting design flaws via metrics in Object-Oriented systems. In 39^{th} International Conference and Exhibition on Technology of Object-Oriented Languages and Systems, page 173, Washington, DC, USA, 2001. IEEE Computer Society.

[6] Adrian Trifu and Iulian Dragos. Strategy-based elimination of design flaws in object- oriented systems. In 4^{th} Workshop on Object-Oriented Reengineering, pages 55–61, 2003.

[7] Michele Lanza and Radu Marinescu. Object-Oriented Metrics in Practice - Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer, 2006.

[8] Javier Pérez, Carlos López, Naouel Moha, and Tom Mens. A classification framework and survey for design smell management. Technical Report 2011/01, Grupo GIRO, Departamento de Informática, Universidad de Valladolid, 3 2011.

[9] K. Alkharabsheh, Y. Crespo, E. Manso, and J.A. Taboada. Software design smell detection: a systematic mapping study. Software Quality Journal, 2018.

[10] Francisco J. Pérez-García. Refactoring planning for design smell correction in object oriented software. PhD thesis, Universidad de Valladolid, 2011.

[11] David Budgen, John Bailey, Mark Turner, Barbara Kitchenham, Pearl Brereton, and Stuart Charters. Lessons from a cross-domain investigation of empirical practices. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, EASE'08, pages 88–99, Italy, 2008. British Computer Society.

[12] Eva van Emden and Leon Moonen. Java quality assurance by detecting code smells. In 9^{th} Working Conference on Reverse Engineering, pages 97–107. IEEE Computer Society, 10 2002.

[13] Khalid Alkharabsheh, Yania Crespo, Manuel Fernández-Delgado, José R. Viqueira, and José A. Taboada. Exploratory study of the impact of project domain and size category on the detection of the god class design smell. Software Quality Journal, 29(2):197 – 237, 2021.

[14] Tushar Sharma, Vasiliki Efstathiou, Panos Louridas, and Diomidis Spinellis. Code smell detection by deep direct-learning and transfer-learning. Journal of Systems and Software, 176:110936, 2021.

[15] Thanis Paiva, Amanda Damasceno, Eduardo Figueiredo, and Cláudio Sant'Anna. On the evaluation of code smells and detection tools. Journal of Software Engineering Research and Development, 5(1):7, Oct 2017.

[16] Douglas Kirk, Marc Roper, and Murray Wood. A heuristic-based approach to code-smell detection. In 1^{st} Workshop on Refactoring Tools, number 2007-8, pages 55–56, 7 2007.

[17] Francesca Arcelli Fontana, Pietro Braione, and Marco Zanoni. Automatic detection of bad smells in code: An experimental assessment. Journal of Object Technology, 11(2):5: 1–38, 2012.

[18] Can Zhu Junpeng Jiang and Xiaofang Zhang. An empirical study on the impact of code contributor on code smell. International Journal of Performability Engineering, 16(7):1067, 2020.

[19] Xiaofeng Han, Amjed Tahir, Peng Liang, Steve Counsell, and Yajing Luo. Understanding code smell detection via code review: A study of the openstack community. CoRR, abs/2103.11446, 2021.

[20] Ananta Kumar Das, Shikhar Yadav, and Subhasish Dhal. Detecting code smells using deep learning. In TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), pages 2081–2086, 2019.

[21] Francesca Arcelli Fontana, Elia Mariani, Andrea Morniroli, Raul Sormani, and Alberto Tonello. An experience report on using code smells detection tools. In Fourth International IEEE Conference on Software Testing, Verification and Validation, ICST 2012, Berlin, Germany, 21-25 March, 2011, Workshop Proceedings, pages 450–457, 2011.

[22] A. Hamid, M. Ilyas, M. Hummayun, and A. Nawaz. A Comparative Study on Code Smell Detection Tools. International Journal of Advanced Science and Technology, 60:25–32, 2013.

[23] Eduardo Fernandes, Johnatan Oliveira, Gustavo Vale, Thanis Paiva, and Eduardo Figueiredo. A review-based

comparative study of bad smell detection tools. In Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16, pages 18:1–18:12, New York, NY, USA, 2016. ACM.

[24] Mika Mäntylä and Casper Lassenius. Subjective evaluation of software evolvability using code smells: An empirical study. Empirical Software Engineering, 11(3):395–431, 2006.

[25] Mika Mäntylä, Jari Vanhanen, and Casper Lassenius. Bad smells - humans as code critics. In 20th International Conference on Software Maintenance (ICSM 2004), 11-17 September 2004, Chicago, IL, USA, pages 399–408, 2004.

[26] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, and Andrea De Lucia. 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, pages 101–110, 2014.

[27] Aiko Fallas Yamashita and Leon Moonen. Do code smells reflect important maintainability aspects? In 28th IEEE International Conference on Software Maintenance, ICSM 2012, Trento, Italy, September 23-28, 2012, pages 306–315, 2012.

[28] Aiko Fallas Yamashita and Leon Moonen. Do developers care about code smells? an exploratory survey. In 20th Working Conference on Reverse Engineering, WCRE 2013, Koblenz, Germany, October 14-17, 2013, pages 242–251, 2013.

[29] Cohen Patricia. West Stephen G. Aiken Leona S. Cohen, Jacob. Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences. Taylor Francis Group, Mahwah, 3rd edition edition, 2002.

[30] Kilem Gwet. Handbook of inter-rater reliability: The definitive guide to measuring the extent of agreement among raters. 4rd edition edition, 01 2012.

[31] Bernhard Ganter and Rudolf Wille. Formal Concept Analysis: Mathematical Foundations. Springer, Berlin/Heidelberg, 1999.

[32] William C. Wake. Refactoring Workbook. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[33] Alban Tiberghien, Naouel Moha, Tom Mens, and Kim Mens. Répertoire des défauts de conception. Technical Report 1303, University of Montreal, 2007.

[34] Alberto Mendo Alonso, Javier Sobrino Fernández, Javier Pérez, and Yania Crespo. Evaluación de herramientas de detección y corrección de defectos de diseño. Technical Report 2011/02, Grupo GIRO, Departamento de Informática, Universidad de Valladolid, 3 2011.

[35] Ghulam Rasool and Zeeshan Arshad. A review of code smell mining techniques. J. Softw. Evol. Process, 27(11):867–895, 11 2015.

[36] Matthew James Munro. Product metrics for automatic identification of" bad smell" design problems in java source-code. In Software Metrics, 2005. 11th IEEE International Symposium, pages 15–15. IEEE, 2005.

[37] Eduardo Kessler Piveta, Marcelo Hecht, Marcelo Soares Pimenta, and Roberto Tom Price. Detecting bad smells in aspectj. J. UCS, 12(7):811–827, 2006.

[38] Bartosz Walter and Blazej Pietrzak. Multi-criteria detection of bad smells in code with UTA method. In 6^{th} International Conference on Extreme Programming and Agile Processes in Software Engineering, volume 3556 of Lecture Notes in Computer Science, pages 154–161. Springer, 6 2005.

[39] Marie Davidian, Brian Everitt, Ron S. Kenett, Geert Molenberghs, Walter Piegorsch, and Fabrizio Ruggeri, editors. Wiley StatsRef: Statistics Reference Online. John Wiley & Sons, Ltd, 2014.

[40] Carlos López. JSmellSensor, 2012.

[41] Carlos López. Detección de defectos de diseño mediante métricas de código. PhD thesis, Universidad de Valladolid, 12 2012.

[42] Khalid Alkharabsheh, Shahed Almobydeen, Jose A Taboada, and Yania Crespo. Influence of nominal project knowledge in the detection of design smells: An exploratory study with god class. International Journal of Advanced Studies in Computers, Science and Engineering, 5(11):120, 2016.

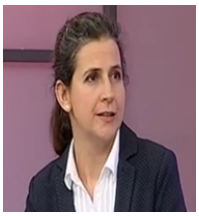
[43] Khalid Alkharabsheh, Yania Crespo, Jose Taboada, and Esperanza Manso. Comparación de herramientas de detección de design smells. In XXI Jornadas de Ingeniería de Software y Bases de Datos, 2016.

[44] Simon Andrews and Constantinos Orphanides. Fcabedrock, a formal context creator. In Conceptual Structures: From Information to Intelligence, 18th International Conference on Conceptual Structures, ICCS 2010, Kuching, Sarawak, Malaysia, July 26-30, 2010. Proceedings, pages 181–184, 2010.

[45] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén. Experimentation in Software Engineering. International Series in Software Engineering. Springer-Verlag, 2012.

KHALID ALKHARABSHEH is Assistant professor at the Department of Software Engineering of the Al Balqa Applied University (BAU). He is a member of COGRADE research group and trainee of the Centro Singu lar de Investigación en Tecnoloxías Intelixentes (CITIUS). He received his Master degree (MS) in computer science from Al-Balqa Applied University, Jordan in 2005, and the Ph.D. from Santiago de Compostela University, Spain in 2019. His research interests focused on machine learning, software quality, empirical software engineering, software validation, and verifcation, Design Smell Detection, Object-Oriented language, and Java.

SADI ALAWADI holds a Ph.D. in Computer science/AI with honor from the Research Center of Intelligent Technologies (CiTIUS) of the University of Santiago de Compostela, Spain, in 2018 and a Master degree in Softcomputing and intelligence system, 2012, from Granada University - Spain. During the last two years he worked as a postdoc for the IOTAP Research Center at Malmö University, Sweden. Currently, he is working at the Department of Information Technology, Division of Scientific Computing, Uppsala University. His main research lines include IOT systems, IoT middleware, End-User Development in the IOT, Machine learning, Deep learning, federated learning, transfer learning, Smart cities, and their related systems, Bigdata, Real-time analysis, Dimensionality reduction, and data visualization Context Awareness, and Blockchain.



YANIA CRESPO is an Associate Professor at the Computer Science Department of the Universidad de Valladolid (UVA). She is a founding member of the GIRO research group of UVA. She obtained a master's degree (1995) and a PhD (2000) in computer science at the University of Havana and University of Valladolid, respectively. Her current research interests are related to software quality, smells detection and refactoring, software product lines, software maintenance and evolution.

M. ESPERANZA MANSO is a Professor at the Computer Science Department of the Universidad de Valladolid. She is a founding member of the GIRO (Grupo de Investigación en Reutilización y Orientación a Objeto) research group of the UVA. Her current research interests are related to software engineering, programming languages, and information science. Their current project is Threshold Prediction

JOSÉ A. TABOADA GONZÁLEZ is Associate Professor in the Department of Electronics Computing at the University of Santiago de Compostela (USC), Spain, and currently member of the Centro Singular de Investigación en Tecnoloxías da Información (CITIUS) at the same University. He is graduated from the electronics program at the Faculty of Physics in 1990 and he received his PhD in Applied Physics in 1996 from the USC. He has been part of several national and European projects and contracts in the fields of Intelligent Systems. Other fields of interest include Big Data.