Numerical Signatures of Antipatterns: An Approach based on B-Splines

Rocco Oliveto
Software Engineering Lab (SE@SA Lab)
DMI – University of Salerno, Italy
roliveto@unisa.it

Foutse Khomh, Giuliano Antoniol, and Yann-Gaël Guéhéneuc
SOCCER Lab and Ptidej Team
École Polytechnique de Montréal, QC, Canada
foutsekh@iro.umontreal.ca, {giuliano.antoniol, yann-gael.gueheneuc}@polymtl.ca

***Abstract*—Antipatterns are poor object-oriented solutions to recurring design problems. The identification of occurrences of antipatterns in systems has received recently some attention but current approaches have two main limitations: either (1) they classify classes strictly as being or not antipatterns, and thus cannot report accurate information for borderline classes, or (2) they return the probabilities of classes to be antipatterns but they require an expensive tuning by experts to have acceptable accuracy. To mitigate such limitations, we introduce a new identification approach, ABS (Antipattern identification using B-Splines), based on a similarity computed via a numerical analysis technique using B-splines. We illustrate our approach on the Blob and compare it with DECOR, which uses strict thresholds, and with another approach based on Bayesian Beliefs Networks. We show that our approach generally outperforms previous approaches in terms of accuracy.**

I. Introduction

Antipatterns [1] are poor solutions to recurring design problems. They occur in object-oriented systems when developers unwillingly introduce them while designing and implementing the classes of their systems. Antipatterns have a negative impact on the quality of a system [1], [2].

Consequently, their identification has received recently more attention from both researchers and practitioners who have proposed various approaches to detect them. In particular, two approaches were proposed: DECOR, [3] the first systematic method to specify and generate automatically detection algorithms for code smells and antipatterns and Bayesian Beliefs Networks (BBNs), an approach based on BBNs for ranking classes according to their probabilities of participating in antipatterns [4].

However, previous approaches have two main limitations: either (1) they classify classes strictly as being or not antipatterns and thus cannot report accurate information for borderline classes (*e.g.*, DECOR), or (2) they return the probabilities of classes to be antipatterns but require expensive (in time and knowledge) tuning by experts to have acceptable accuracy (*e.g.*, BBNs).

The first limitation lead to the "submarine" effect: several classes may be very close to be identified as antipatterns but remain under the threshold during their evolution. Minor changes can then bring them all above the threshold, falsely leading developers to suspect the latest changes as culprit.

The second limitation cause the detection results to rely too much on the experts' judgement. An incomplete experts' knowledge can cause a high number of false positives, resulting in a waste of time and resources for developers and managers that must skim through the results. Moreover, a model built in a specific context is not easily generalisable to other contexts and should be recalibrated to be effective. The recalibration is a difficult task as historical data and context knowledge are not always available.

In this paper, we propose a new identification approach, called ABS (Antipattern identification using B-Splines), to help overcome these limitations. ABS is based on a learning technique using an interpolation method from the numerical analysis field. The basis of ABS is the building of *signatures* of classes based on quality metrics, as in [5], using B-splines [6], [7] to abstract the metric values. ABS models a class using specific interpolation curves (*i.e.*, B-splines) of plots mapping metrics and their values for the class. The similarity of a given class to an antipattern is computed by calculating the distance between the curve of the class and the curves of classes previously classified as antipatterns (or good classes). Like BBNs, ABS needs a corpus of antipatterns. However, in contrary to BBNs, ABS does not need experts' knowledge to define a learning structure; thus, it reduces the bias introduced in BBNs by the experts' subjectivity when structuring the BBNs of the antipatterns.

We apply ABS to detect the Blob antipattern. The identification accuracy of ABS was compared with DECOR and BBNs. The case study shows that, in general, ABS provides better results than BBNs and DECOR. In few explainable cases, ABS has lower accuracy than previous approaches, *i.e.*, when the training set to build the signature is too small. We also implement ABS in "Sign-o-meter", a tool to assist developers in assessing quickly the probability of classes to become Blobs and the evolution of this probability.

In summary, the main advantages of ABS with respect to previous approaches are:

- When identifying Blobs, it generally outperforms previous approaches in accuracy;

- It is more attractive in practice thanks to the speed and ease to generate and interpret the signatures;
- It is directly portable across systems;
- Its class representation can be also used to monitor the evolution of the quality of a class.

The paper is organised as follows. Section II summarises and discusses previous work. Section III describes and justify our approach for identifying antipatterns, while Section IV provides details on the design of the case study carried out to evaluate the proposed approach. Sections V and VI report on the results achieved and summarises the lessons learned from the case study, respectively. Section VII concludes and suggest future work.

II. Related Work

Webster [8] wrote the first book on "antipatterns" in object-oriented development; his contribution covers conceptual, political, coding, and quality-assurance problems. Riel [9] defined 61 heuristics characterising good object-oriented programming to assess software quality manually and improve design and implementation. Beck [2] defined 22 code smells, suggesting where developers should apply refactorings. Mäntylä [10] and Wake [11] proposed classifications of code smells. Brown *et al.* [1] described 40 antipatterns, including the well-known Blob. These books provide in-depth views on heuristics, code smells, and antipatterns aimed at a wide academic and industrial audience. We build upon this work to propose an approach to characterise antipatterns and identify classes with similar characteristics.

Several approaches to specify and identify code smells and antipatterns have been proposed in the literature. They range from manual approaches, based on inspection techniques [12], to metric-based heuristics [13], [3], [14], where antipatterns are identified according to sets of rules and thresholds defined on various metrics. Manual approaches were defined, for example, by Travassos *et al.* [12], who introduced manual inspections and reading techniques to identify code smells.

Marinescu [13] presented a metric-based approach to identify smells with detection strategies, which capture deviations from good design principles and consist of combining metrics with set operators and comparing their values against absolute and relative thresholds. Similarly to Marinescu, Munro [14] proposed metric-based heuristics to identify code smells; the heuristics are derived from template similar to the one used for design patterns. He also performed an empirical study to justify the choice of metrics and thresholds for detecting code smells.

Moha *et al.* [3] proposed the DECOR method to specify and automatically generate identification algorithms. DECOR includes a domain-specific language based on a literature review of existing work. It also includes algorithms and a platform to automatically convert specifications into identification algorithms and apply these algorithms on any system. DECOR lead to identification algorithm with good precision and perfect recall while allowing quality analysts to easily adapt the specifications to their context. We choose DECOR because it is the current state-of-the-art threshold-based identification approaches.

Khomh *et al.* [4] argued that threshold-based approaches do not handle the uncertainty of the detection results and, therefore, miss borderline classes, *i.e.*, classes with characteristics of antipatterns "surfacing" slightly above or "sinking" slightly below the thresholds because of minor variations in the characteristics of these classes. Consequently, they proposed a BBN for the identification of antipatterns in systems, which output is the probability that a class exhibiting the characteristics of an antipattern is truly an antipattern. Thus, their approach handles the degree of uncertainty that a class is or is not an antipattern. They also showed that BBNs can be calibrated using historical data both from a similar and from a different context. We choose also this approach for comparison because it is the only approach able to qualify continuously the probability of classes to be antipatterns.

Some visualisation techniques, for example [15], were used to find a compromise between fully-automatic identification techniques, which are efficient but lose track of the context, and manual inspections, which are slow and subjective. Other approaches perform fully-automatic identification and use visualisation to present the identification results [16], [17].

Other related approaches include architectural consistency checkers, which have been integrated in style-oriented architectural development environments [18], [19], [20]. For example, active agents acting as critics [20] can check properties of architectural descriptions and identify potential syntactic and semantic errors.

III. Antipattern Identification using B-splines

This section presents our approach for the identification of antipatterns using their signatures, described as B-splines [6], [7] and built from numerical analysis. Recently, B-splines was used to characterise software artefacts (documentation and source code) to compute the textual similarity between them for traceability recovery [21].

ABS first models each class by its particular interpolation curves, *i.e.*, its B-splines, built using a set of metrics and their values for the class. In ABS, we also model antipatterns using B-splines, inferred from a set of classes known as participating in the antipatterns.

Then, we estimate the risk of a class to be an antipattern by computing the similarity of its signature from the signature of known antipattern. To improve its accuracy when computing the risk, we also consider the distance from both antipatterns and a set of good quality classes. We define good classes as any classes that does not participate in an antipattern. The similarity is then computed by calculating the

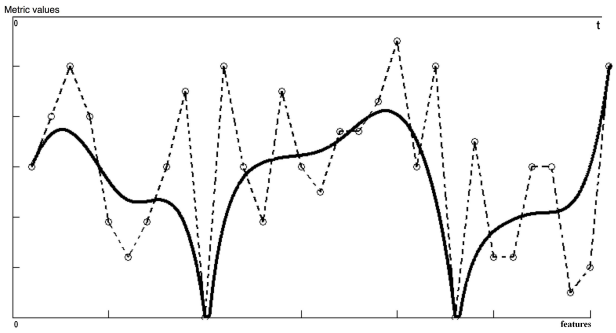


Figure 1. Class/antippaterns representation using B-splines

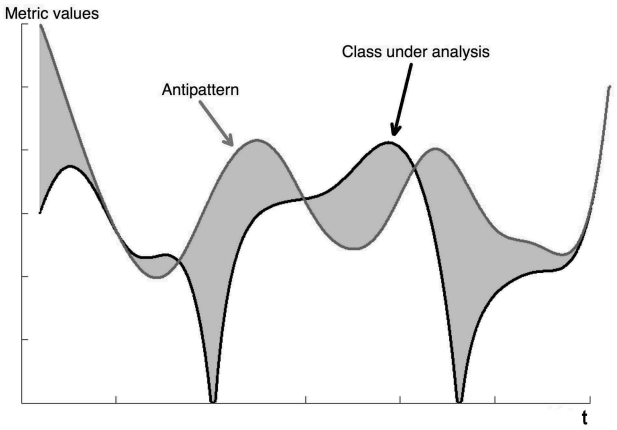


Figure 2. Signature comparison using B-splines

similarity between the corresponding interpolation curves.

We now present how (i) to define a curve representing a class and (ii) to identify candidate antipattern in a system. We illustrate our approach using the Blob. The Blob is also called God class [9]. It is defined as a class that centralises functionality and has too many responsibilities. Brown *et al.* [1] characterise its structure as a large controller class that depends on data stored in several surrounding data classes.

A. Identifying the Signature of Classes

A class can be described by a set of metric values; metrics measuring for example the class cohesion, coupling, and complexity. We can thus represent each class as a set of points in a Cartesian plane with the metrics in X and their values for that class in Y.

Let $M = m_1, m_2, \ldots, m_n$ be a set of metrics computed for each class of a system, then a class c_j is represented by the set of points $\{p_1, p_2, \ldots, p_n\}$ where the coordinates of the generic point p_i are calculated as follows:

$$coord_x(p_i) = i$$

$$coord_y(p_i) = m_{i,j}$$

where $m_{i,j}$ represents the value of the metric m_i for the class c_j.

This representation of a class allows us to use a numerical analysis technique to define the signature of the class by interpolating the points associated to the class. We consider the points representing a class as control points (also called De Boor's points) of a uniform B-spline curve [6], [7], which is a generalisation of a Bézier curve, *i.e.*, a piecewise Bézier curves [6], [7]. Thus, given the set of points $(P_{j_i})_{i\in\{1,\ldots,n\}}$ where $P_{j_i} = (i, m_{i,j})$ represent the class c_j, the parametric B-spline curve on the domain $[0, 1]$, with degree *k* and equidistant *knots* (uniform B-spline) $\{t_l : l = 0, \ldots, n + k - 1\}$, is defined as follows [6], [7]:

$$Bspline_{c_j}(t) = \sum_{i=0}^{n-1} (P_{j_{i+1}}) \cdot B_{i,k}(t).$$

where $t_0 \leq t_1 \leq \cdots \leq t_{n+k-1}$, $t_l \in [0, 1]$, and the polynomials $B_{i,k}(t)$ are calculated using the numerically stable De Boor's recursive formula [6], [7]:

$$B_{l,0}(t) = \begin{cases} 1 & \text{if } t_l < t < t_{l+1} \\ 0 & \text{otherwise,} \end{cases}$$

$$B_{l,k}(t) = \frac{(t - t_l) \cdot B_{l,k-1}(t)}{t_{k+l-1} - t_l} + \frac{(t_{k+l} - t) \cdot B_{l+1,k-1}(t)}{t_{k+l} - t_{l+1}}$$

The domain and co-domain of our B-spline function are thus respectively $[0, 1]$ and R. Figure 1 shows the graphical representation of a class. The dashed and the bold lines denote the control polynomial and the B-spline curve representing the class, respectively. The B-spline is a curve approximation technique where the control points influence the shape of the curve but the curve does not interpolate the control points, except for the first and last points [6], [7]. However, it is possible to force the B-spline to interpolate a set of given points [6], [7] and we force the B-spline to interpolate the points that represent metrics with values equal to 0, as shown in Figure 1: we do not give to these points the average value of the preceding and following points.

We choose a B-spline to derive a curve representing a class because (1) it is fast and easy to compute, (2) it provides local control on the curve, and (3) the degree of the B-spline is independent of the number of control points. In particular, we observed in our experiments that computing the B-spline never took more than 1 second and that $k = 30$ is a good compromise between computational complexity and identification accuracy. The latter observation is consistent with that in previous work [21].

B. Comparing the Signature of Classes

Once obtained the signatures of each class of a system, it is possible to compare these with those of classes previously classified as antipatterns. The conjecture is that if a class

has a signature similar to that of a previously classified antipattern, then the class is also probably this antipattern.

Let $B = \{b_1, b_2, \ldots, b_p\}$ be the set of classes previously classified as being an antipattern, for example Blobs[1]. The class signatures of class c_i and of Blob b_j can be compared using the distance between the corresponding B-spline curves, *i.e.*, $Bspline_{c_i}(t)$ and $Bspline_{b_j}(t)$. We define the distance between two B-spline curves using the normalised 1-norm, as shown in Figure 2:

$$D(c_i, b_j) = \int_0^1 \left| Bspline_{c_i}(t) - Bspline_{b_j}(t) \right| dt$$

The order of the metrics on the X-axis influences the representation of a class and the distance between two classes (or a class and an antipattern). We studied different ways of ordering the metrics and did not find any substantial difference in the identification results.

We define two approaches to compute the similarity of a class to an antipattern. In the first approach, called *one-tailed*, we compute the similarity between the class signature and that of the *ideal* signature of the antipattern, defined as the average signature of some previously classified antipatterns. Thus, if B is the set of classes with the antipattern, the similarity between a class c_i and that antipattern, *e.g.*, Blob (also known as God Class), is:

$$godliness(c_i, B) = 1 - \frac{D(c_i, b_{ideal})}{\max_{j,k} D(c_j, b_{ideal})}$$

where the higher is the "godliness" [22] value, the higher is the similarity of the class to an ideal Blob.

In the second approach, called *two-tailed*, we compute the similarity of a class to an antipattern including also its similarity to classes previously classified as good quality classes. Thus, other than building the set B, we also build $G = \{g_1, g_2, \ldots, g_q\}$, containing the signatures of classes previously classified as good classes. We then compute the similarities of c_i with every elements in B and G and rank each pair according to these similarities. Finally, for the Blob, we compute the godliness of the class as:

$$godliness(c_i, B) =$$

$$\left| \sum_{j=1}^{p} w_{B_{i,j}} \cdot \frac{D(c_i, b_j)}{\max_{l,s} D(c_l, b_s)} - \sum_{k=1}^{q} w_{G_{i,k}} \cdot \frac{D(c_i, g_k)}{\max_{l,s} D(c_l, g_s)} \right|$$

where $w_{B_{i,j}} = \frac{1}{pos(c_i, b_j)}$ and $w_{G_{i,k}} = \frac{1}{pos(c_i, g_k)}$ represent weights based on the positions of the pairs (c_i, b_j) and (c_i, g_k), respectively, in the ranked list.

[1]These classes can be class of a same system or classes taken from different systems.

C. Visualising and Following the Evolution of Signatures

The graphical representation of the B-splines characterising classes and antipatterns is also useful to monitor the evolution of classes in time. We implemented a tool, called Sign-o-Meter, that displays to related and complementary views of the signatures of a class:

1) One dial showing the similarity between the class under development and an antipattern, *e.g.*, Blob using the godliness metric, computed using the *two-tailed* approach.
2) A plot of three B-spline curves representing, respectively, the current signature of the class, the signature of the previous version of the class, and the signature of an antipattern. This plot uses the *one-tailed* approach only.

Figure 3 illustrates Sign-o-Meter on the Blob for class `org.apache.xerces.xs.XSElementDecl` in Xerces v2.7.0. On the left hand side, it shows the dial displaying the godliness measure of the class. Higher is the value, higher is the probability that the class is a Blob.

On the right hand side, it shows a dash B-Spline curve for the previous version of the class, a thin continuous line for the B-spline curve of the current version of the class, and a thick line for the curve characterising the ideal Blob (from the previous classified Blob).

Such a plot provides information on the evolution of the class by allowing developers to compare the signatures of the current version of a class to the one of its previous version and to the ideal antipattern. Consequently, it allows developers to realise that their class is moving towards or away from the ideal antipattern and, thus, to take the appropriate actions.

IV. Design of the Case Study

We study our novel approach to identify classes participating in antipatterns and demonstrate the applicability of the numerical analysis technique. The description follows the Goal–Question–Metric template [23].

A. Definition and Context

The *goal* of our case study is to analyse whether ABS has a better identification accuracy for the Blob than DECOR [3] and than an approach based on BBNs [4]. The *purpose* of our case study is to show that ABS helps in improving the quality of systems by supporting the detection of antipatterns. The *quality focus* is to improve the accuracy of the identification of antipattern, and in particular that of the Blob. The *perspective* is both of researchers, who want to evaluate (1) the use of numerical analysis for representing classes and (2) the improvement of the identification accuracy of Blob; and of quality analysts, who perform evaluation activities and are interested in locating parts of a system that need improvements with the least possible efforts.

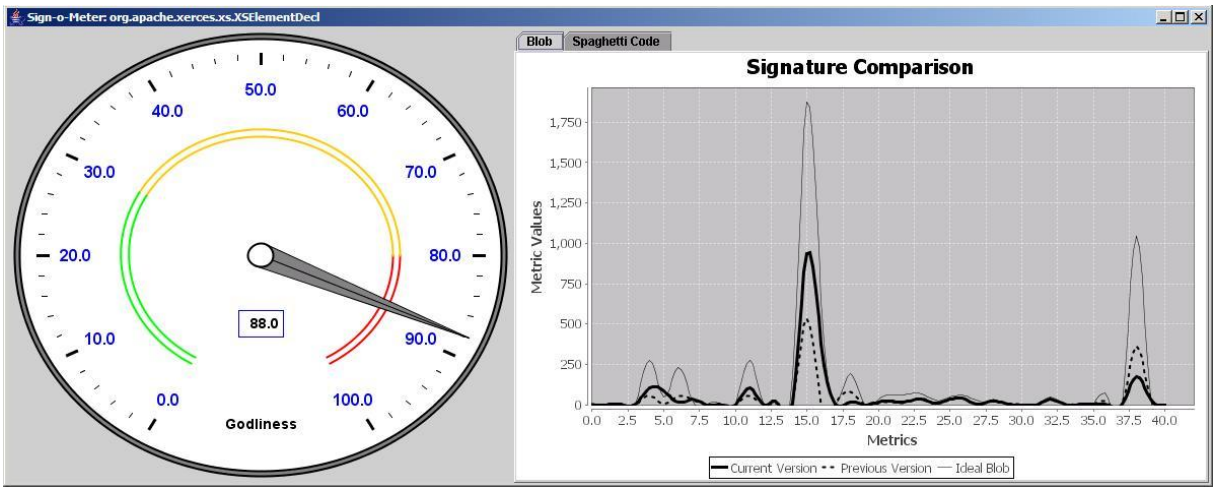


Figure 3. Screen-shot of Sign-o-Meter on class `org.apache.xerces.xs.XSElementDecl`

Table I
PROGRAM CHARACTERISTICS

Systems	# of Classes	KLOCs	# of Blobs
Gantt Project v1.10.2	188	31	4
Xerces v2.7.0	589	240	15
Total	777	271	19

We conducted our case study using two open-source Java systems: GanttProject v1.10.2 and Xerces v2.7.0, which characteristics are summarised in Table I. GanttProject[2] is a system for creating project schedules by means of Gantt charts and resource-load charts. It enables breaking down projects into tasks and establishing dependencies between these tasks. Xerces[3] is a family of packages for parsing and manipulating XML files. It implements a number of standard API for XML parsing, including DOM, SAX, and SAX2. We chose these systems because they are medium-size open-source systems, yet small enough to manually identify occurrences of the Blobs.

To evaluate the accuracy of ABS, we asked two undergraduate students and two graduate students to identify occurrences of the Blob in the two systems manually. The pair of undergraduate students performed the task together to follow previous results [24] hinting that, on maintenance activities, the performance of a pair of undergraduate students is about the same as that of one graduate student.

Prior to their manual identification of Blobs, the students were presented with several anti-patterns and the Blob in particular. Then, each student/pair analysed every class of the two systems systematically and classified it as a Blob or a good class. We subsequently independently combined their analyses and whenever at least two of the three students/pair considered a class as a Blob, we tagged the class as a true occurrence. The number of Blobs is reported in Table I.

All metrics and properties required to identify antipatterns are extracted using the POM framework [5]. The metrics and the list of identified Blobs are available for replication [4].

B. Planning and Research Questions

We study the accuracy of our approach in the two following scenarios:

- *intra-system identification*: we assume that historical data (*i.e.*, correctly identified Blobs) are available for a given system (*i.e.*, Xerces). We use this data to identify other Blobs in the same system. We divide the classes of Xerces in three subsets with 5 instances of Blobs in each subset. Then, we use two of the subsets as training set and the third as test set in a 3-fold cross-validation. (We did not use GanttProject because it contains too few instance of Blob, *i.e.*, 4).
- *extra-system identification*: we study the accuracy of ABS using heterogeneous data. We assume that a quality analyst has access to the historical data from one system *i.e.*, GanttProject. This data is then used to identify Blobs in the other system, *i.e.*, Xerces. We also perform the same study in the other direction, *i.e.*, using Blobs in Xerces to identify occurrences in GanttProject.

We formulated the following research questions:

- **RQ**$_1$: To what extent a model built with a B-spline is able to detect Blobs in a system?
- **RQ**$_2$: Is a model built with a B-spline better than state of-the-art approaches, such as DECOR and BBNs, both intra system and extra system?

C. Data Collection and Analysis

To answer the two previous research questions for each identification scenario, we collect the number of correct

[2]http://ganttproject.biz/index.php
[3]http://xerces.apache.org/
[4]http://www.ptidej.net/downloads/experiments/csmr10a/

Blobs and false positives identified by each identification approach. We compared the identified Blobs with the manually-built Oracle via the Information Retrieval (IR) metrics [25]:

$$recall = \frac{|correct \cap suggested|}{|correct|}$$

$$precision = \frac{|correct \cap suggested|}{|suggested|}$$

where $correct$ and $suggested$ represent the sets of known Blobs and of candidate Blobs suggested by an approach, respectively.

V. Analysis and Interpretation of the Results

We now discuss the results of our case study.

A. Intra-system Identification

First, to answer $\mathbf{RQ}_1$, we apply ABS to identify the Blobs in Xerces using previously-identified Blobs in the same system. We use both the *one tailed* and the *two tailed* approaches. We use different sizes of set G, *i.e.*, the set of classes classified as good quality classes. We achieve the best accuracy when the size of G is twice as much as that of B (*i.e.*, 20 good quality classes for 10 Blobs).

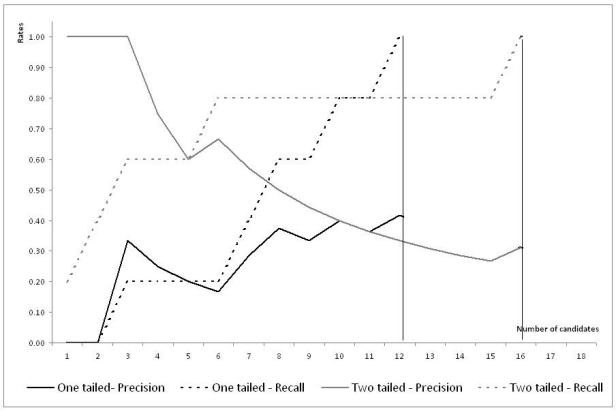
Figure 4 shows the precision and recall curves achieved when performing the 3-fold cross-validation. (In this figure and the following, the fine vertical lines represent the limits where 100% recall is reached for each approach.) The training and test sets influence the results. However, in all the three cases, all the Blobs were identified by analysing almost the same number of classes (*i.e.*, 18, 21, and 20 when applying the *one-tailed* approach and 16, 22, and 19 when applying the *two-tailed* approach).

Both the *one tailed* and *two tailed* approaches are quite stable and the same cutting-threshold was used in all the three cases for identifying all the Blobs, *i.e.*, 0.9 for the *one-tailed* approach and 0 for the *two-tailed* approach. (In the two-tailed approach, the godliness of a class is computed as the *difference* between the similarities of the class to Blobs and the similarities to good classes. Thus, the threshold is naturally 0: the balance between Blobs and good classes.)

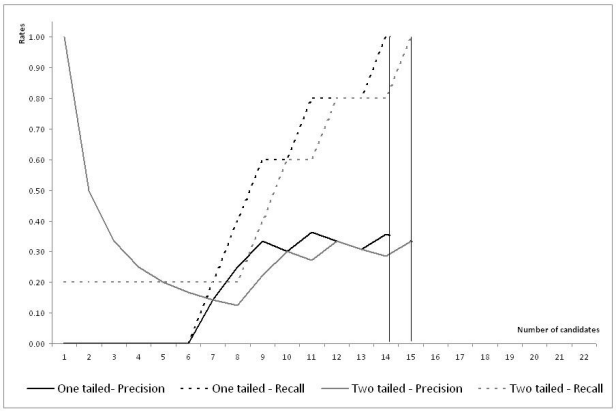
Even if the two approaches achieved comparable level of precision when recall is equal to 1, in general the *two-tailed* approach sensibly outperforms the *one-tailed* approach. In particular, the former provides generally better accuracy in two cases (as shown in Figures 4a and 4c). In the third case, the accuracies of the two approaches are comparable (see Figure 4b).

With respect to $\mathbf{RQ}_2$, Table II shows the number of candidate classes that have to be inspected before detecting the five known Blobs. The order of inspection depends on the godliness level of a class.

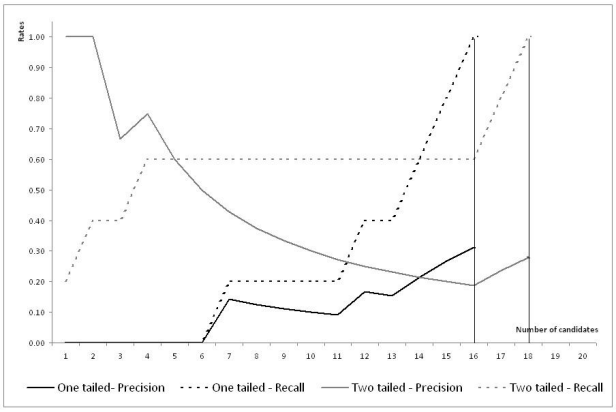
The accuracy of DECOR is similar to that of ABS, because a quality analyst must potentially inspect 13, 13, and 17 occurrences to identify the five known Blobs. Such a result is due to the small size of B, *i.e.*, too few occurrences of the Blobs because Xerces contains only 15 Blobs and 10 of these are used to train ABS for identifying the remaining 5 Blobs. Thus, better results would be achieved by enriching the set of previously classified Blobs. (The sum of the occurrences returned by DECOR does not equal the total



(a)

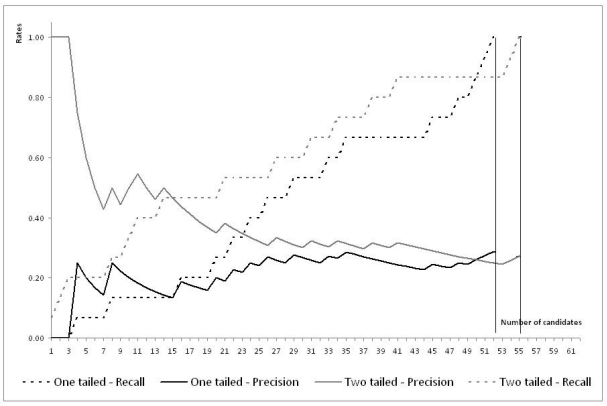


(b)

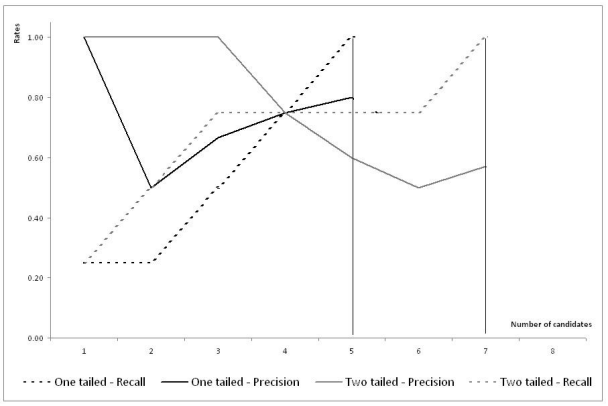


(c)

Figure 4. Intra project identification on Xerces: Precision and recall curves achieved performing a 3-fold cross-validation.



(a) Results achieved on Xerces



(b) Results achieved on GanttProject

Figure 5. Extra-system identification: Results achieved with ABS

Table II
INTRA-SYSTEM IDENTIFICATION ON XERCES: COMPARISON OF DECOR, BBNS, AND ABS.

Folds	# of Classes to Inspect				# of Blobs
	DECOR	BBN	ABS		
			One Tailed	Two Tailed	
1	13	6	12	16	5
2	13	7	14	15	5
3	17	9	16	18	5

Table III
EXTRA-SYSTEM IDENTIFICATION: COMPARISON OF DECOR AND ABS (TWO TAILED).

Systems	# of Classes to Inspect		# of Blobs
	DECOR	ABS	
GanttProject	16	7	4
Xerces	45	55	15

number of Blobs, 45, when analysing Xerces as a whole because two occurrences must include classes in *another fold* to be considered Blobs.) accuracy of the BBN approach is better than that of ABS, for the same reasons as before. However, for ABS, the precision of 0.3 for a recall of 1 is still acceptable to help improving the quality of a system.

B. Extra-system Identification

To answer $\mathbf{RQ}_1$, we apply ABS to identify Blobs in GanttProject and Xerces while assuming data on the Blobs in the other system. We also compare the two approaches *one tailed* and *two tailed*. Again, best accuracy was achieved for the *two-tailed* approach when the number of classes previously classified as good classes is twice the number of the classes previously classified as Blobs.

Figure 5 shows the precision and recall curves in the two cases. The lozenges for DECOR show that a quality analyst must study all the detected occurrences to identified the Blobs. The results achieved on Xerces show that the both two *one-tailed* and *two-tailed* approaches are able to identify all the correct occurrences of the Blob with the same level of precision (more than 20%). Yet, the *two-tailed* approach provides generally better identification accuracy when analysing the trend of precision and recall. Similar results are achieved on GanttProject, but the *one tailed* approach has a better trend of recall and precision.

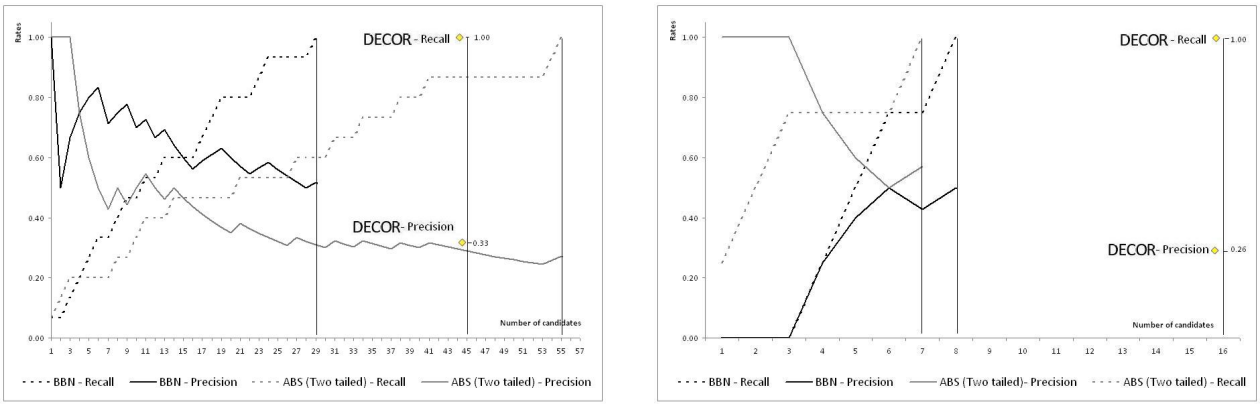
The results achieved in the extra-system scenario confirm those in the intra-system scenario. They also confirm the stability of ABS. In particular, the godliness thresholds used for identifying all the Blobs were the same as the ones used in the intra-system scenario, *i.e.*, 0.9 and 0.

With respect to $\mathbf{RQ}_2$, we compare the accuracy of ABS with DECOR and BBNs. The best results in the previous experiments were achieved for ABS using the the *two-tailed* approach, therefore, we use this approach for comparing ABS with DECOR and BBNs.

Table III shows the number of candidate classes that must be inspected before identifying the four known Blobs in GanttProject using DECOR and ABS (*two-tailed* approach). The order of inspection depends on the godliness level of the class, yet the accuracy of ABS is better than that of DECOR on GanttProject, while almost comparable on Xerces (due to the little number of Blob in GanttProject). These results confirm those obtained previously and strongly suggests that providing ABS with a large number of Blobs increases the identification accuracy of our approach.

The results achieved when comparing ABS with DECOR are confirmed when comparing ABS with BBNs. Figure 6 shows the precision and recall curves achieved by the two approaches on the two systems. ABS again outperforms BBNs on GanttProject, while BBNs provide better results on Xerces. ABS requires the inspection of 7 and 57 classes to identify the Blobs in GanttProject and Xerces, respectively. While BBNs required the analysis of 10 and 36 classes to identify Blobs in GanttProject and Xerces, respectively.

The results achieved are promising because they suggest that in the absence of historical data or in the presence of



(a) Results achieved on Xerces

(b) Results achieved on GanttProject

Figure 6. Extra project identification: Comparison of DECOR, BBNs, and ABS (two tailed)

little historical date on a specific system, quality analysts could use ABS to exploit the knowledge from different systems and obtain acceptable precision and recall in the identification of antipatterns. These results also show that quality analysts could use our novel approach to exploit data external to her company and then adapt and apply ABS to her context successfully.

C. Threats to Validity

The main threat to the *external validity* that could affect the generalisation of the results presented previously relates to the analysed systems. To mitigate this threat, we used two medium-size systems, which support different activities and are open source and thus are available for replication purposes. Nevertheless, we plan to replicate our study in the future on larger systems to further confirm the generalisability of our results.

A threat to the *internal validity* is represented by the oracle used to analyse the accuracy of the identification approaches. Our oracle was manually defined by analysing the two systems used in the case study, which include an amount of subjectivity. To mitigate this threat and reduce the risk of classification errors, we asked four students (two undergraduate students and two graduate students) to independently identify occurrences of the Blob in the two systems. Prior to their manual detection of Blobs, students were presented with several anti-patterns and the Blob in particular, yet we did not interfere with their chosen process of identification to avoid biaising their process towards occurrences that could be more easily identified by ABS. Moreover, to mitigate the differences of background between students, undergraduate students performed the task together because, on maintenance activities, the performance of a pair of undergraduate students is about the same as that of one graduate student. Finally, an candidate occurrence was classified as real Blob only when two or more students classified it as such. Such a process makes us quite confident about the accuracy of the Oracle.

Regarding the *construct validity*, *i.e.*, the relation between theory and observation, recall and precision are widely used metrics for assessing the accuracy of a classification method. Such metrics are useful to analyse the accuracy of an approach, as well as to compare it with different approaches.

VI. Lessons Learned

The results of the case study taught us a number of lessons and highlighted some open issues:

- **B-splines are a valuable approach for representing and characterising classes.** The results of the case study show that the accuracy of ABS is comparable or superior to that of previous approaches. In addition, ABS provides a graphical, simple, and fast means to compare classes, as shown in Figure 1.
- **The signatures can be used to monitor the evolution of classes.** The graphical representation and the godliness are also useful to developers to monitor the evolution of classes on a day-to-day basis and provide feedback to the developers on the "quality" of their classes as they evolve, as illustrated by the Sign-o-Meter in Section III-C.
- **ABS is simple.** No background knowledge is required to built either the rule cards or the structure of the BBNs. Thus, ABS does not embed the experts' subjective understanding of the definitions of the antipatterns. Yet, it requires an Oracle that provides occurrences of some antipatterns. However, this Oracle can be built and improved collectively and–or borrowed from others.
- **ABS is directly portable across systems.** ABS keep a consistent accuracy when applied to different systems while, to maintain their accuracies, both BBNs and DECOR require recalibration of the conditional probabilities [4] or changes to the thresholds [3]. In addition, with ABS, the same thresholds, *e.g.*, godliness threshold, can be used across various systems.
- **The number of occurrences of an antipattern in**

the training sets influences the accuracy of ABS. In comparison to BBNs, ABS requires a higher number of already-classified occurrences to provide the same accuracy. BBNs compensate the lack of occurrences with experts' knowledge when available. However, the portability of ABS allows to improve its accuracy by "borrowing" occurrences from different systems.

VII. Conclusion and Future Work

In this paper, we presented a novel approach to identify occurrences of antipatterns using the signature of the classes and of the antipatterns. Signature are built using numerical analysis of metric values. We use a B-splines to approximate the signature of a class or an antipattern and the distance between signatures to identify classes participating to antipatterns. The signature of an antipattern is computed as the average of the signatures of a set of known classes participating to the antipattern.

We introduce two approaches to use B-splines to compare the signature of a class with that of an antipattern: a one-tailed approach, where the similarity between the class signature and that of the antipattern is computed, and a two-tailed approach, where this similarity is computed using both the signature of the antipattern and the average signature of a set of good quality classes.

We illustrated our approach on the Blob antipattern and compared it with DECOR [3], which uses strict thresholds, and an approach based on BBNs. We showed that our approach outperforms in general previous approaches in precision and recall while being more attractive in practice thanks to the speed and ease to generate and interpret the signatures. In few explainable cases, our approach has lower precision or recall than previous approaches, when the training set to build the signature is too small.

Future work include replicating our cast study on larger systems to assess the generalisability of our novel approach. It also include computing the signatures of other antipatterns than the Blob and again replicate our study to assess the impact of the antipattern on accuracy. Finally, we are currently working on an empirical study to assess the usefulness to developers of the Blob-o-Meter when assessing and evolving the quality of a system.

Acknowledgment

This work has been partly funded by the Canada Research Chair on Software Patterns and Patterns of Software.

References

[1] W. J. Brown, R. C. Malveau, W. H. Brown, H. W. McCormick III, and T. J. Mowbray, *Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis*, 1^{st} ed. John Wiley and Sons, March 1998. [Online]. Available: www.amazon.com/exec/obidos/tg/detail/-/0471197130/ref=ase_theantipatterngr/103-4749445-6141457

[2] M. Fowler, *Refactoring – Improving the Design of Existing Code*, 1^{st} ed. Addison-Wesley, June 1999.

[3] Naouel Moha, Y.-G. Guéhéneuc, L. Duchien, and A.-F. L. Meur, "DECOR: A method for the specification and detection of code and design smells," *Transactions on Software Engineering (TSE)*, 2009, 16 pages. [Online]. Available: http://www-etud.iro.umontreal.ca/~ptidej/Publications/Documents/TSE09.doc.pdf

[4] Foutse Khomh, Stéphane Vaucher, Y.-G. Guéhéneuc, and H. Sahraoui, "A bayesian approach for the detection of code and design smells," in *Proceedings of the* 9^{th} *International Conference on Quality Software (QSIC)*, C. Byoung-ju, Ed. IEEE Computer Society Press, August 2009, 10 pages. [Online]. Available: http://www-etud.iro.umontreal.ca/~ptidej/Publications/Documents/QSIC09.doc.pdf

[5] Y.-G. Guéhéneuc, H. Sahraoui, and Farouk Zaidi, "Fingerprinting design patterns," in *Proceedings of the* 11^{th} *Working Conference on Reverse Engineering (WCRE)*, E. Stroulia and A. de Lucia, Eds. IEEE Computer Society Press, November 2004, pp. 172–181, 10 pages. [Online]. Available: http://www-etud.iro.umontreal.ca/~ptidej/Publications/Documents/WCRE04.doc.pdf

[6] C. de Boor, "On calculating with b-splines," *Journal of Approximation Theory*, vol. 6, pp. 50–62, 1972.

[7] M. G. Cox, "The numerical evaluation of b-splines," *Journal of the Institute of Mathematics and its Applications*, vol. 10, pp. 134–149, 1972.

[8] B. F. Webster, *Pitfalls of Object Oriented Development*, 1^{st} ed. M & T Books, February 1995. [Online]. Available: www.amazon.com/exec/obidos/ASIN/1558513973

[9] A. J. Riel, *Object-Oriented Design Heuristics*. Addison-Wesley, 1996.

[10] M. Mantyla, "Bad smells in software - a taxonomy and an empirical study." Ph.D. dissertation, Helsinki University of Technology, 2003.

[11] W. C. Wake, *Refactoring Workbook*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[12] G. Travassos, F. Shull, M. Fredericks, and V. R. Basili, "Detecting defects in object-oriented designs: using reading techniques to increase software quality," in *Proceedings of the* 14^{th} *Conference on Object-Oriented Programming, Systems, Languages, and Applications*. ACM Press, 1999, pp. 47–56.

[13] R. Marinescu, "Detection strategies: Metrics-based rules for detecting design flaws," in *Proceedings of the* 20^{th} *International Conference on Software Maintenance*. IEEE Computer Society Press, 2004, pp. 350–359.

[14] M. J. Munro, "Product metrics for automatic identification of "bad smell" design problems in java source-code," in *Proceedings of the* 11^{th} *International Software Metrics Symposium*, F. Lanubile and C. Seaman, Eds. IEEE Computer Society Press, September 2005. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/METRICS.2005.38

[15] F. Simon, F. Steinbrückner, and C. Lewerentz, "Metrics based refactoring," in *Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR'01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 30.

[16] M. Lanza and R. Marinescu, *Object-Oriented Metrics in Practice*. Springer-Verlag, 2006. [Online]. Available: http://www.springer.com/alert/urltracking.do?id=5907042

[17] E. van Emden and L. Moonen, "Java quality assurance by detecting code smells," in *Proceedings of the 9th Working Conference on Reverse Engineering (WCRE'02)*. IEEE Computer Society Press, Oct. 2002. [Online]. Available: citeseer.ist.psu.edu/vanemden02java.html

[18] D. Garlan, R. Allen, and J. Ockerbloom, "Architectural mismatch: Why reuse is so hard," *IEEE Software*, vol. 12, no. 6, pp. 17–26, 1995.

[19] R. Allen and D. Garlan, "A formal basis for architectural connection," *ACM Transactions on Software Engineering and Methodology*, vol. 6, no. 3, pp. 213–249, 1997. [Online]. Available: citeseer.ist.psu.edu/allen97formal.html

[20] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, "A comprehensive approach for the development of modular software architecture description languages," *ACM Transactions on Software Engineering and Methodology*, vol. 14, no. 2, pp. 199–245, 2005.

[21] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella, "Traceability recovery using numerical analysis," in *Proceedings of 16th Working Conference on Reverse Engineering*. Lille, France: IEEE CS Press, 2009.

[22] Stéphane Vaucher, Foutse Khomh, Naouel Moha, and Y.-G. Guéhéneuc, "Prevention and cure of software defects: Lessons from the study of god classes," in *Proceedings of the* 16^{th} *Working Conference on Reverse Engineering (WCRE)*, G. Antoniol and A. Zaidman, Eds. IEEE Computer Society Press, October 2009, 10 pages. [Online]. Available: http://www-etud.iro.umontreal.ca/~ptidej/Publications/Documents/WCRE09b.doc.pdf

[23] R. Basili and D. M. Weiss, "A methodology for collecting valid software engineering data," *IEEE Transactions on Software Engineering*, vol. 10, no. 6, pp. 728–738, November 1984. [Online]. Available: www.research.avayalabs.com/user/weiss/Publications.html

[24] F. Ricca, M. D. Penta, M. Torchiano, P. Tonella, M. Ceccato, and C. A. Visaggio, "Are fit tables really talking?: a series of experiments to understand whether fit tables are useful during evolution tasks," in *Proceedings of the* 30^{th} *international conference on Software engineering*, M. D. Wilhelm Schäfer and V. Gruhn, Eds. IEEE Computer Society Press, 2008, pp. 361–370.

[25] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*. Addison-Wesley, 1999.