

Appendices

Annexure-A

A.1. Assessing the Appropriateness of paper for PSs

In the above section, we have explained step by step how we have identified the most relevant articles from different digital libraries that can respond to our RQs and according to our inclusion and exclusion criteria. In this section, we describe how a subset of the most relevant studies from 226 selected articles have been identified to study the effect of the evolution of anti-patterns on OOSE, SOA, web services, and Android applications. The assessment and synthesis processes are described below.

A.2. Assessment Mechanism

Our approach to identify the most suitable papers to analyze the effect of anti-patterns on different types of applications is motivated by quality check notions of Kitchenham et al. [21]. Our assessment mechanism is focused on identifying those publications that provide enough information for synthesis across studies to address our research questions. We created and implemented a set of diagnostic parameters to ensure that a lot of evolutionary information (model building data, techniques, datasets, tools, contextual, evaluation measures, etc.) was reported through identified PSs [12]. To be part of this SLR, the paper must report a set of these parameters. It is impossible to properly grasp what was done in a study without this. It is also difficult to contextualize the conclusions given by a study without this. Our assessment procedure for reviewing a study to qualify for our primary study database consists of the following five steps:

Step 1: Confirmation of impact finding study In this review, we only include those approaches that report or analyze the impact or effect of different types of code smells on OOSE, SOA, and Android applications. Few studies appear to be reporting about the impact of smells but in reality, their proposed models did not address the effects of anti-patterns on applications. Few studies reported developer problems concerning code smell impacts. Most of the approaches discussed the relationship between different code metrics and the impacts of smells. These types of approaches focused on smells detection, not their effects. Furthermore, A model is said to be an effect estimate model if it was trained and tested on different datasets [144]. Evolution/impact/effect indicators for the selection of studies in this SLR are given in Table 1.

We only selected those approaches that reported some sort of effects of code smells on one of our domains with models testing on open source or commercial datasets. The assessment record of 226 studies is shown in Table 19.

Step 2: Checking the necessary contextual data

In this step, we ensured that the selected studies must contain contextual information that can be used to perform appropriate interpretation of findings. If an approach does not have complete information about the contextual data then we cannot check the model performance, apply or repeat this study. The incomplete information of the model makes it complex for users to understand and implement. Such types of models cannot be reused for new situations. The contextual indicators and assessment are given in Tables 3 & 4 that we have applied to our 226 identified papers.

Step 3: Checking the necessary model building indicators

Without clear knowledge about modeling techniques, dependent and independent variables, we cannot select an approach that will be answerable to our research questions. The model building information indicators and assessment are given in Tables 22 & 23 which we have applied to our 226 selected papers.

Step 3: Checking the necessary model building indicators

Without clear knowledge about modeling techniques, dependent and independent variables, we cannot select an approach that will be answerable to our research questions. The model building information indicators and assessment are given in Tables 22 & 23 which we have applied to our 226 selected papers.

Step 4: Checking the Dataset Indicators

The datasets play a vital role to make an approach reliable. Tables 24 & 25 present the dataset indicators and assessment that we have applied to verify the quality of the datasets used in various PSs.

Step 5. The Evaluation Measures

The evaluation of the approach must be done by using suitable evaluation techniques according to domains such as accuracy, precision, recall, f-score, AUC, ROC, or comparing the results with available benchmarks. The studies validate their findings by using different evaluation techniques. The communities accept only those studies that validated their claims by using evaluation techniques. Therefore, it is obvious we cannot select any paper for our SLR that did not give any evaluation of their findings. The evaluation assessment of 226 identified relevant studies is given in Table 26.

Primary Studies Selection Algorithm

The primary studies have been selected by applying the selection algorithm given in Figure [?] This selection algorithm assesses the quality of each paper. according to our assessment criteria defined in Tables 18, 20, 21, 23, 25. 269 papers have been selected through this algorithm. The library-wise detail of selected studies is given in Table 27.

Table 18: Secondary studies about different aspects of Anti-patterns

Sr.	Ref	Title of Secondary studies	Year
S1	[34]	"A Literature Review on Code Smells and Refactoring, master's thesis,"	2010
S2	[35]	"A review of code smell mining techniques,"	2015
S3	[5]	"A review-based comparative study of bad smell detection tools,"	2016
S4	[7]	"A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems,"	2019
S5	[6]	A systematic literature review: code bad smells in java source code,"	2017
S6	[36]	"A systematic literature review: Refactoring for disclosing code smells in object-oriented software,"	2017
S8	[8]	"A systematic review on the code smell effect,"	2018
S9	[24]	"The impact of code smells on software bugs: A systematic literature review,"	2018
S10	[43]	"A survey on UML model smells detection techniques for software refactoring,"	2019
S11	[44]	"Machine learning techniques for code smell detection: A systematic literature review and meta-analysis,"	2019
S12	[37]	"Software Design Smell Detection: a systematic mapping study,"	2019
S13	[38]	"Software smell detection techniques: A systematic literature review,"	2019
S14	[25]	"Towards a collaborative repository for the documentation of service-based anti-patterns and bad smells,".	2019
S15	[39]	"Oracles of Bad Smells: a Systematic Literature Review,"	2020
S16	[40]	"Code smells: A Synthetic Narrative Review,"	2020
S17	[22]	"Code smells and refactoring: A tertiary systematic review of challenges and observations,"	2020
S18	[45]	"Empirical evaluation of the impact of object-oriented code refactoring on quality attributes: A systematic literature review,"	2018
S19	[41]	"Impact of Code Smells on the Rate of Defects in Software: A Literature Review"	2018
S20	[42]	"A systematic literature mapping on the relationship between design patterns and bad smells,"	2018
S21	[42]	"Causes, impacts, and detection approaches of code smell: a survey,"	2018

Table 19: List of tools Used by All PSs

Tool name	PSs Reference	Domain	No of PSs	Tool name	PSs Reference	Domain	No of PSs
DECOR	[6, 8, 12, 18, 29, 44, 46, 115, 50, 118]	OOSE, android	10	Git	20	SOA	1
Borland Together	[54, 55, 59, 107, 69, 13, 120, 94]	OOSE	8	JCODEODOR	21	OOSE	1
inCode	[55, 56, 59, 107, 69, 13, 94]	OOSE	7	iPerfDetector	22	SOA	1
PMD	[96, 75, 111, 62, 75, 28]	OOSE,SOA	6	JAVA Parser	23	SOA	1
Understand	[67, 26, 120]	OOSE,SOA	3	JDOM	24	OOSE	1
WEKA	[106, 112]	OOSE	2	Sonar	26	SOA	1
Arcan,	[58, 78]	OOSE	2	PerfImpact	33	SOA	1
JDeodorant	[61, 28]	OOSE	2	Test Smell Detector,	34	OOSE	1
Columbus	[62, 67]	OOSE	2	RefactoringMiner	35	OOSE	1
iPlasma	[14, 84]	OOSE	2	Perl	36	OOSE	1
Paprika toolkit	[100, 102]	android	2	Ref-Finder	37	OOSE	1
JCodeOdor,	[111, 112]	OOSE	2	RefDetector	37	OOSE	1
CKJM	[51, 108]	SOA ,android	2	JNOSE test	40	OOSE	1
CVSChangelog	[54]	OOSE	1	inFusion	42	OOSE	1
Pattern-detection	[56]	OOSE	1	JIRA	43	OOSE	1
SonarQube	[58]	OOSE	1	CodeVizard	45	OOSE	1
aDoctor	[10]	SOA	1	LBSDetectors	47	OOSE	1
PETrA	[10]	SOA	1	Designite1	53	OOSE	1
Detex	[60]	OOSE	1	Foodcritic	58	OOSE	1
CCEA	[116]	OOSE	1	Test Smell	60	OOSE	1
SpIRIT	[63]	OOSE	1	MuLATO	65	android	1
VerveineJ7	[63]	OOSE	1	Mimec	69	OOSE	1
inFamix	51	OOSE	[1]	Ptidej	70	OOSE	1
custom built -OOSE	[58, 110]	OOSE	1	NDepends	73	OOSE	1
custom built -android	[29]	android	1	DeMIMA	74	OOSE	1
ESLint21	[117]	SOA	1	NiCad	63	OOSE	1

Table 27: List of Primary Studies

Sr.	Ref	Title of Primary Study	Library
1	[50]	“Improving the survivability of RESTful Web applications via declarative fault tolerance”	Wiley
2	[97]	“An bad smells and class error probability in the post-release object-oriented system evolution”	S. Direct
3	[82]	“Code smells as system-level indicators of maintainability: An empirical study”	S. Direct
4	[109]	“The relationship between design patterns and code smells: An exploratory study”	S. Direct
5	[61]	“Code Smell Severity Classification using Machine Learning Techniques”	S. Direct
6	[105]	“How developers perceive smells in source code: A replicated study”	S. Direct
7	[71]	“Towards a Severity and Activity based Assessment of Code Smells”	S. Direct
8	[27]	“A Large-Scale Empirical Study on the Lifecycle of Code Smell Co-occurrences”	S. Direct
9	[108]	“Are architectural smells independent from code smells? An empirical study”	S. Direct
10	[10]	“On the Impact of Code Smells on the Energy Consumption of Mobile Applications”	S. Direct
11	[88]	“To what extent can maintenance problems be predicted by code smell detection?–An empirical study”	S. Direct
12	[98]	“An exploratory study of the impact of antipatterns on class change- and fault-proneness”	Springer
13	[72]	“Competitive Evolutionary Code-Smells Detection”	Springer
14	[110]	“Investigating the evolution of code smells in object-oriented systems”	Springer
15	[62]	“Assessing the capability of code smells to explain maintenance problems: an empirical study combining quantitative and qualitative data”	Springer
16	[89]	“Recognizing antipatterns and analyzing their effects on software maintainability”	Springer
17	[51]	“An code smells for refactoring”	Springer
18	[99]	“Investigating the relation between lexical smells and change-and fault-proneness: an empirical study”	Springer
19	[11]	“On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation”	Springer
20	[73]	“A large-scale empirical study of code smells in JavaScript projects”	Springer
21	[118]	“Improving change prediction models with code smell-related information”	Springer
22	[111]	“iPerfDetector: Characterizing and detecting performance anti-patterns in iOS applications”	Springer
23	[75]	“Studying the evolution of exception handling anti-patterns in a long-lived large-scale project”	Springer
24	[91]	“Common Refactorings, a Dependency Graph and some Code Smells: An Empirical Study of Java OSS”	ACM
25	[92]	“Drivers for Software Refactoring Decisions”	ACM
26	[100]	“Effectiveness of Encapsulation and Object-oriented Metrics to Refactor Code and Identify Error Prone Classes using Bad Smells”	ACM
27	[14]	Are Automatically-Detected Code Anomalies Relevant to Architectural Modularity? An Exploratory Analysis of Evolving Systems	ACM
28	[95]	Investigating the Impact of Code Smells Debt on Quality Code Evaluation	ACM
29	[112]	Detecting Bad Smells in Source Code using Change History Information	ACM
30	[86]	Exploring the Impact of Inter-smell Relations on Software Maintainability: An Empirical Study	ACM
31	[63]	Some Code Smells Have a Significant but Small Effect on Faults	ACM
32	[77]	An Empirical Study of the Performance Impacts of Android Code Smells	ACM
33	[74]	Mining performance regression inducing code changes in evolving software	ACM
34	[116]	On the Diffusion of Test Smells in Automatically Generated Test Code: An Empirical Study	ACM
35	[140]	Why We Refactor? Confessions of GitHub Contributors	ACM
36	[119]	Empirical Analysis on Effectiveness of Source Code Metrics for Predicting Change-Proneness	ACM
37	[79]	Investigating the relationship between bad smells and bugs in software systems	ACM

38	[29]	A Large-Scale Empirical Study on the Effects of Code Obfuscations on Android Apps and Anti-Malware Products	ACM
39	[64]	Does refactoring improve software structural quality? A longitudinal study of 25 projects	ACM
40	[143]	On the influence of Test Smells on Test Coverage	ACM
41	[26]	Investigating the Role of Code Smells in Preventive Maintenance	ACM
42	[3]	On the Impact of Design Flaws on Software Defects	ACM
43	[90]	Investigating the Impact of Design Debt on Software Quality	ACM
44	[12]	An Exploratory Study of the Impact of Code Smells on Software Change-proneness	IEEE
45	[120]	The Evolution and Impact of Code Smells: A Case Study of Two Open Source Systems	IEEE
46	[78]	When and Why Your Code Starts to Smell Bad (and Whether the Smells Go Away)	IEEE
47	[93]	The Effect of Lexicon Bad Smells on Concept Location in Source Code	IEEE
48	[76]	Analyzing the Impact of Antipatterns on Change-Proneness Using Fine-Grained Source Code Changes	IEEE
49	[13]	Quantifying the Effect of Code Smells on Maintenance Effort	IEEE
50	[101]	Anti-pattern Mutations and Fault-proneness	IEEE
51	[94]	Investigating the Change-proneness of Service Patterns and Antipatterns	IEEE
52	[28]	Evolution of Code Smells over Multiple Versions of Softwares: An Empirical Investigation	IEEE
53	[113]	House of Cards: Code Smells in Open-Source C# Repositories	IEEE
54	[81]	Investigating the Energy Impact of Android Smells	IEEE
55	[114]	The Scent of a Smell: An Extensive Comparison between Textual and Structural Smells	IEEE
56	[30]	An Empirical Analysis on Web Service Anti-Pattern Detection using a Machine Learning Framework	IEEE
57	[65]	Bug Prediction Model using Code Smells	IEEE
58	[117]	Code Smells in Infrastructure as Code	IEEE
59	[115]	Investigating instability architectural smells evolution: an exploratory case study	IEEE
60	[102]	An exploratory study of the relationship between software test smells and fault-proneness	IEEE
61	[121]	An Empirical Study of the Impact of Two Anti-patterns, Blob and Spaghetti Code, On Program Comprehension	IEEE
62	[96]	Investigating the impact of code smells on system's quality: An empirical study on systems of different application domains	IEEE
63	[122]	Investigating the Impact of Code Smells on System's Quality:	IEEE
64	[66]	Predicting Software Change-Proneness with Code Smells and Class Imbalance Learning	IEEE
65	[84]	On the Relevance of Code Anomalies for Identifying Architecture Degradation Symptoms	IEEE
66	[103]	Are the clients of flawed classes (also) defect prone?	IEEE
67	[52]	When Code-Anomaly Agglomerations Represent Architectural Problems?	IEEE
68	[107]	Smells like Teen Spirit: Improving Bug Prediction Performance using the Intensity of Code Smells	IEEE
69	[123]	Do Code Smells Impact the Effort of Different Maintenance Programming Activities?	IEEE
70	[104]	Predicting bugs using antipatterns	IEEE
71	[53]	Do code smells reflect important maintainability aspects?	IEEE
72	[67]	Does Code Quality Affect Pull Request Acceptance? An empirical study	others
73	[54]	A Practical Guide to Support Change-proneness Prediction	others
74	[106]	Machine Learning Approaches for Predicting the Severity Level of Software Bug Reports in Closed Source Projects	others
75	[68]	Web service design defects detection: A bi-level multi-objective approach,	Others
76	[144]	On the Impact of Bad Smell Agglomerations on Software Quality	Others

Table 21: List of tools Used by All PSs

Evolution indicators	Definition of indicators	Reason of indicator
Is the study addressed any effect or impact or evolution of anti-patterns?	The selected approach must be an impact-finding study that reported the impact or effect or evolution or analysis of anti-patterns on software systems. A study is not acceptable if it only discussed detection or the relationship between smells types.	Some studies provided detailed information on the presence and detection of smells but were unable to provide an effect-finding method to predict their impacts, and did not verify their claims using unseen datasets. Such types of studies cannot be included for further synthesis.
Is the impact finding model tested with a valuable dataset?	The selected approach must propose a model that was trained and tested on different datasets. This information can be used for cross-validation and reuse.	If an approach reported the performance of its effect finding model based on training dataset only then such type of model cannot be generalized on unseen /new dataset. Therefore, these types of approaches are not recommended for further synthesis.

Table 22: Impact Indicators Assessment

Library Name	No. of PSs	Parameter				Is the model tested?
		Impact	Effect	Evolution	Analysis	
Wiley	9	0	1	3	2	3
Sc. Direct	22	3	1	4	6	2
Springer	62	8	1	11	9	21
ACM	51	8	7	2	2	30
IEEE	55	7	10	6	7	36
Others	27	4	2	1	2	12
TOTAL	226	30	22	27	28	104

Table 23: Contextual Indicators

Indicator	Definitions of indicator	Reason for indicator
Dataset source	The source of the dataset on which the study is based should be cited. The source of the dataset may be industrial or open source. The versions of the dataset used in the study must be given as they give access to the context of the dataset.	The performance of a model changes when we apply it to different datasets. For example, some models may give good results on open-source datasets as compared to industrial datasets. For this reason, we must have a dataset source for synthesis.
Application domain	The information about the domain on which the impact is studied must be specified. e.g. OOSE, SOA, web services, and Android applications.	Few models are probably specific to the domain. Various domains used different design practices and resulted in various types of code smells. The impact of code smells may also differ across domains. Therefore, to evaluate the performance of the model the domain information must be required. In the absence of this parameter, it is very hard to interpret the findings of the study for synthesis.
Programing language	The programming language(s) of the datasets used in the testing of models must be specified. e.g. Java, C++, PHP, etc.	The impacts of smells varied in different programming languages. For instance, the procedural languages may have different code smells effects as compared to OO languages. It means that the type of language affects the performance of the model. Therefore, it is essential to record the language of the dataset used to verify the efficiency of the approach.

Table 24: Contextual Indicators Assessment

Library Name	No. of PSs	Source of dataset	Application domain	Programing language
Wiley	9	5	5	6
Science Direct	22	14	12	14
Springer	62	23	35	34
ACM	51	27	33	25
IEEE	55	31	38	31
Others	27	13	13	7
Total Paper	226	113	136	117

Table 25: Model Building Indicators

Indicator	Definition of indicator	Reason of indicator
Independent variables (IV)	The foundations on which the impact-finding model is based must be openly and visibly declared. There are different types of independent variables such as anti-patterns, number of anti-patterns, anti-pattern types, service patterns, etc.	The performance of experimental studies is measured based on their independent variable. Such types of studies must require the identification of IV explicitly. The lack of this information may significantly decrease the confidence in the study findings. This is the main barrier in examining the influence of IV on different approaches during synthesis.
Dependent variables (DV)	The selected studies should report the effects of smells on OOSE, SOA, and Android applications in terms of whether or not there is an effect of smell (as categorical DV) or the number of effects in the dataset (as continuous DV). Sometimes few continuous studies additionally report ranked results also. There are different types of dependent variables such as change proneness, Fault-proneness, memory usage, performance, and maintenance, etc.	All experimental studies must define their dependent variables clearly and explicitly. If a study lacks this information the worth of study is significantly decreased. In addition to this, the evaluation of the impact finding study highly depends upon the DV whether they are categorical or continue. This is necessary for the synthesis of our information.
Which Modeling technique is used?	The authors should mention the modeling techniques that were used to develop the model for their studies, such as experimental, survey, case study, algorithms, machine learning, etc. The tool-generated results are not acceptable unless the model of the tool is not reported in the paper.	The various modeling techniques behave differently in different circumstances. The modeling technique used by the study must be reported because without this information we cannot observe the impact of any technique on the performance of the approach.

Table 26: Model Building Assessment

Library Name	No. of PSs	Independent variables	Dependent variable	Modeling technique
Wiley	9	1	1	5
Sc. Direct	22	6	6	13
Springer	62	16	19	40
ACM	51	30	25	44
IEEE	55	24	30	37
Others	27	3	2	16
Total Paper	226	80	83	155

Table 27: Model Building Indicators

Indicator	Definition of indicator	Reason of indicator
Attainment process	The processes used by the study to obtain the dataset should be reported. The core steps taken to obtain the dataset will be sufficient in this regard. If the dataset has been obtained from a third party, then the main procedure of this dataset attainment must be reported or referenced.	The confidence in the dataset used by the approach in the experimentation can be built by providing information about the procedure adapted to the collected dataset. The dataset is fundamental to ensure the quality of the approach and the data collection process has a huge impact on the quality of the dataset. A study's results cannot be reliable if the origin of the dataset is unknown. We cannot select any paper for our SLR that was unable to give any indication of how the dataset was achieved.
Dataset size	The size of the system or dataset being studied should be given in no. of classes, methods, or KLOC. The total size of the system is required. It is not necessary to mention the size of each component of the dataset separately.	The behavior of the systems is mostly influenced by the size of the datasets. Therefore, the impact of code smells may be varied in systems of different sizes. Different types of models will behave differently with different sizes of systems. Therefore, to interpret the study findings for synthesis, it is necessary to record the size of the dataset on which the model of the system was tested.
Link available	The link of the dataset that is used by the approach in experimentation must be given in the paper. The dataset may be placed on GitHub, research group site, etc.	The link of the dataset will be used for the revaluation of an approach. This link can also be used to reuse the same dataset by another approach for experimentation or comparison between the approaches' findings.

Table 28: Model Building Assessment

Library Name	No. of PSs	Independent variables	Dependent variable	Modeling technique
Wiley	9	1	1	5
Sc. Direct	22	6	6	13
Springer	62	16	19	40
ACM	51	30	25	44
IEEE	55	24	30	37
Others	27	3	2	16
Total Paper	226	80	83	155

Table 29: Dataset Indicators

Indicator	Definition of indicator	Reason of indicator
Attainment process	The processes used by the study to obtain the dataset should be reported. The core steps taken to obtain the dataset will be sufficient in this regard. If the dataset has been obtained from a third party, then the main procedure of this dataset attainment must be reported or referenced.	The confidence in the dataset used by the approach in the experimentation can be built by providing information about the procedure adapted to the collected dataset. The dataset is fundamental to ensure the quality of the approach and the data collection process has a huge impact on the quality of the dataset. A study's results cannot be reliable if the origin of the dataset is unknown. We cannot select any paper for our SLR that was unable to give any indication of how the dataset was achieved.
Dataset size	The size of the system or dataset being studied should be given in no. of classes, methods, or KLOC. The total size of the system is required. It is not necessary to mention the size of each component of the dataset separately.	The behavior of the systems is mostly influenced by the size of the datasets. Therefore, the impact of code smells may be varied in systems of different sizes. Different types of models will behave differently with different sizes of systems. Therefore, to interpret the study findings for synthesis, it is necessary to record the size of the dataset on which the model of the system was tested.
Link available	The link of the dataset that is used by the approach in experimentation must be given in the paper. The dataset may be placed on GitHub, research group site, etc.	The link of the dataset will be used for the revaluation of an approach. This link can also be used to reuse the same dataset by another approach for experimentation or comparison between the approaches' findings.

Table 30: Dataset Assessment

Library Name	No. Of PSs	Attainment process	Dataset size	Link available
Wiley	9	2	3	5
Sc. Direct	22	12	5	13
Springer	62	28	14	31
ACM	51	33	25	30
IEEE	55	31	30	32
Others	27	13	9	12
Total Paper	226	119	86	123

Table 31: Evaluation Assessment

Library Name	Total	Evaluation
Wiley	9	3
Science Direct	22	12
Springer	62	23
ACM	51	21
IEEE	55	25
Others	27	10
Total Paper	226	94

Table 32: Selected Primary Studies

Library name	Final Selected
Wiley	1
Science Direct	12
Springer	12
ACM	21
IEEE	26
Others	4
Total selected	76

Studies Selection

```
1: for all studies  $i=1$  to 226 do
2:   if (impact=Yes OR effect=Yes OR evoulution =Yes OR anayais=Yes)
   and (Is model test=Yes) then
3:     "Prediction Criteia is satisfied then check for Conext Criteria"
4:     if (Source of dataset=Yes AND Application domain=Yes AND Pro-
       graming language=Yes) then
5:       conext is satisfied check for model
6:       if Independent variables OR Dependent variable=Yes OR Modelling
       technique=Yes) then
7:         "Modeling criteria is satisfied then check for Data criteria"
8:         if (Attainment process=Yes AND Dataset size=Yes AND Link
           available=Yes) then
9:           "Data criteria is satisfied then check for Evaluation of study "
10:          if Evaluation = Yes then
11:            "All required parameters are satisfied study is qualified for
              primary studies collection list "
12:            "Otherwise ignore the study and go to test for next study "
13:          end if
14:        end if
15:      end if
16:    end if
17:  end if
18: end for
```

Figure 24: Primary Studies Selection Algorithm