Detection of Antipatterns from SOA

Presented By: Fatima Sabir

SP14-PCS-003

Supervisor: Dr. Ghulam Rasool

Co-Supervisor : Dr. Farooq Ahmad

External Co-SuperVisors

Dr. Yaan-Gael

(Head of Canadian Research Chair of Software Engineering and PTIDEJ Lab

Head of Canadian Research Chair on IOT Tier1

Professor at University of Concordia and University of Montreal)

Dr. Naouel Moha

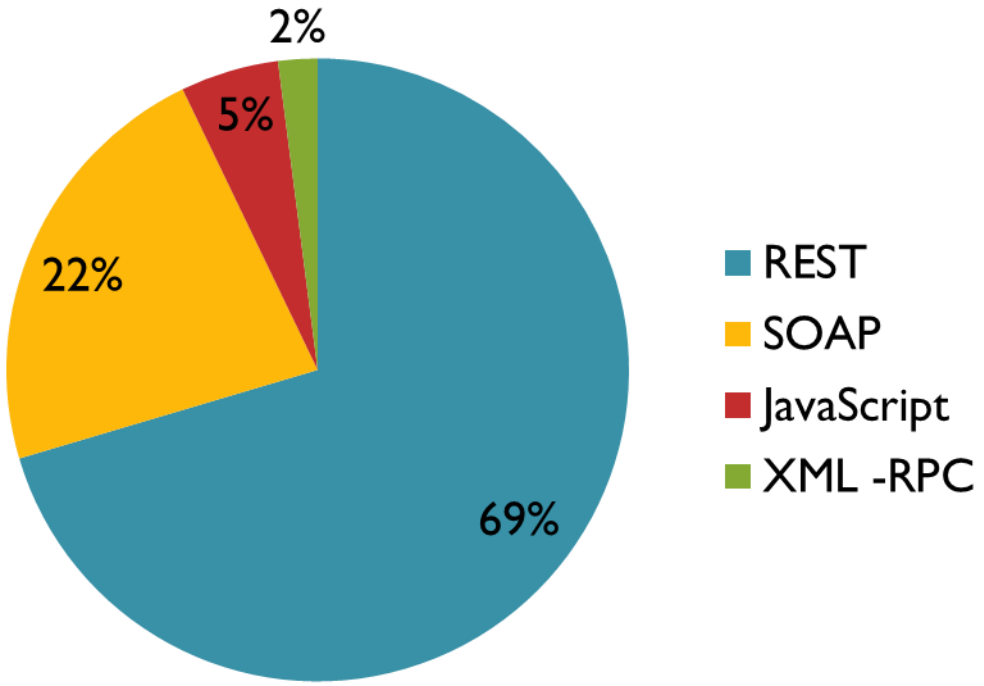
(Head of LATECE Lab , Associate Professor University of Qubic Montreal)

Outline

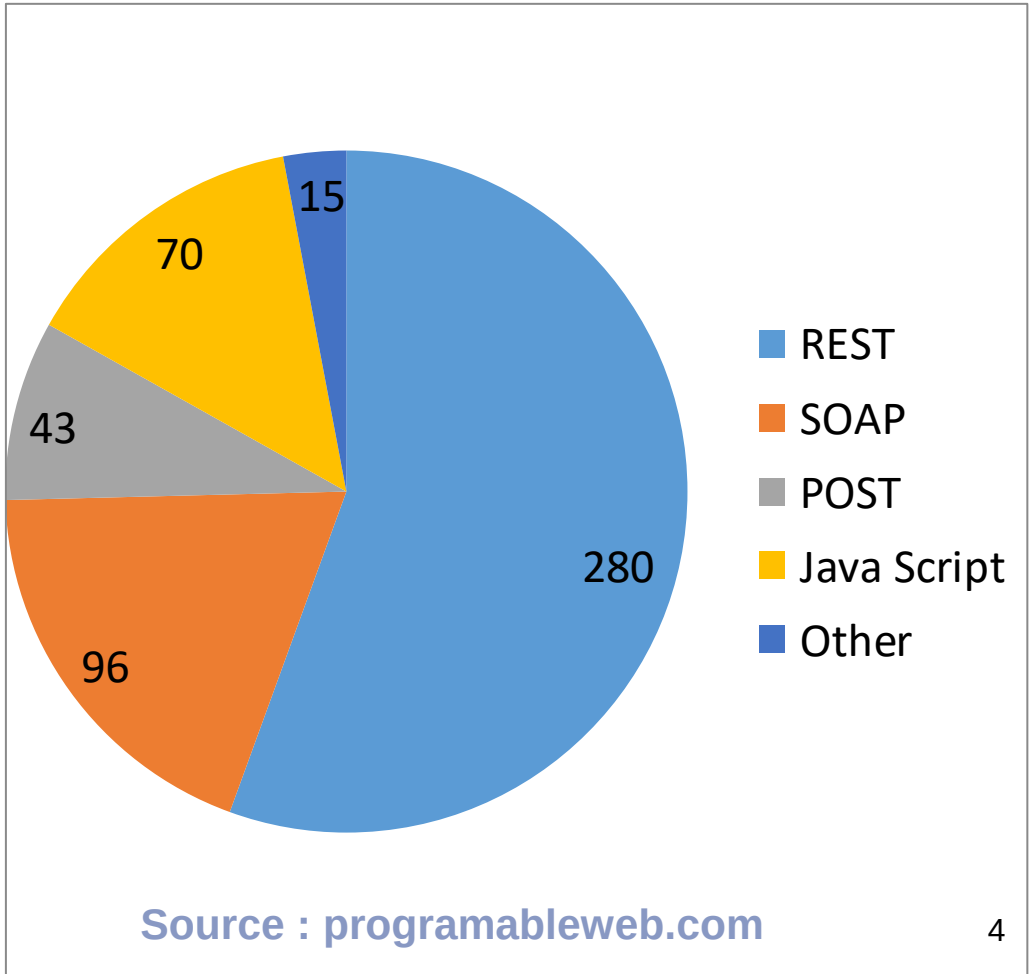
Introduction

1.Introduction

91% of the Software Projects used SOAP and REST web services



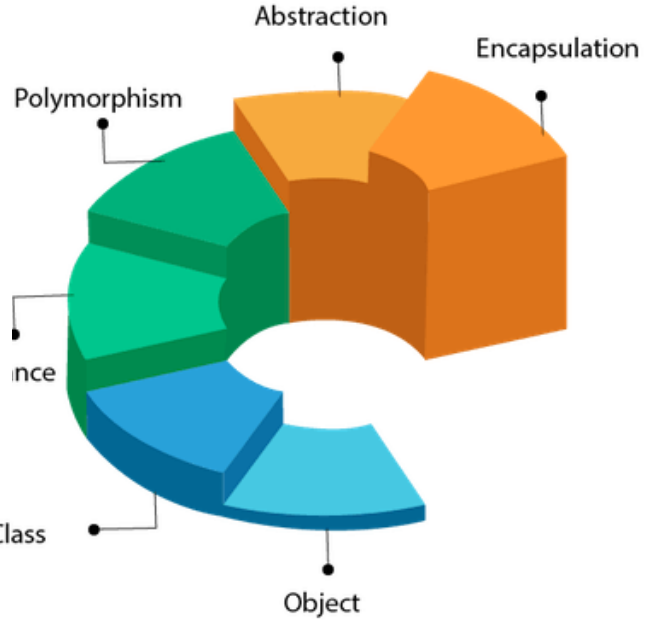
Protocols Used In Google Map APIs



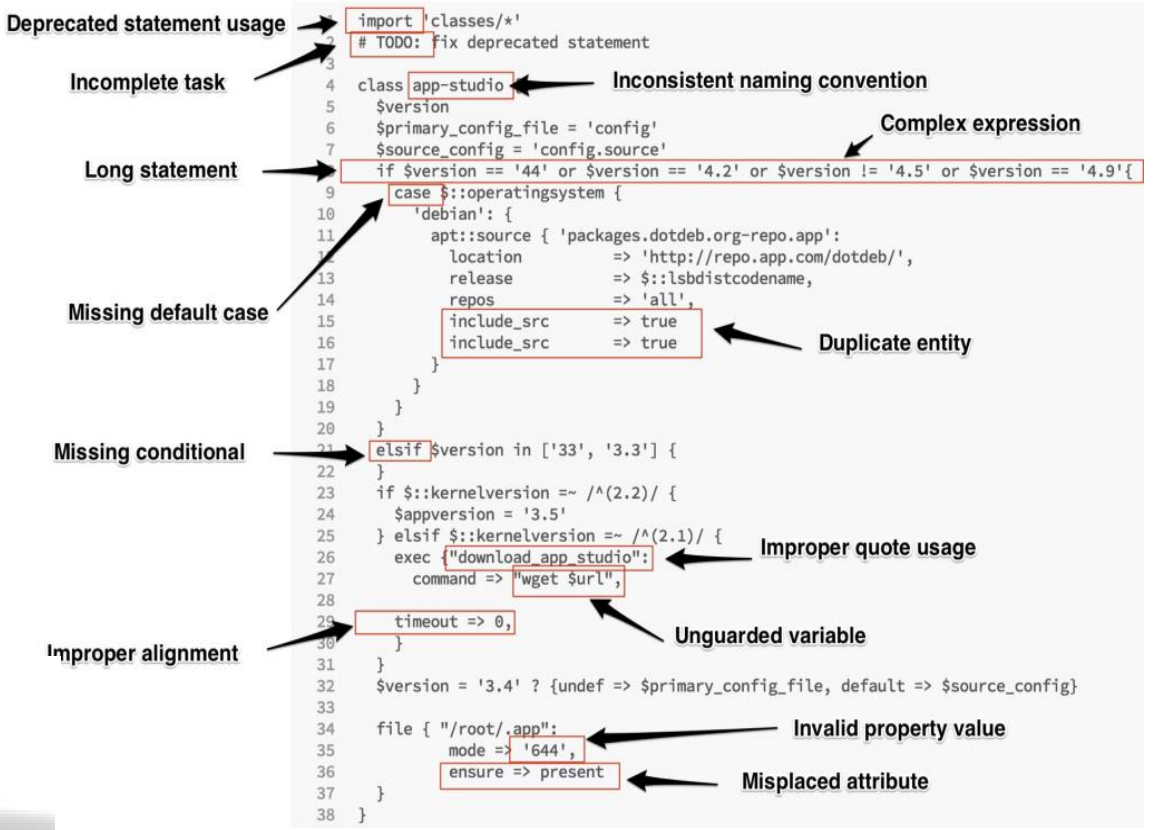
Source : programableweb.com

Removing Flaws and Improving Source Code Quality

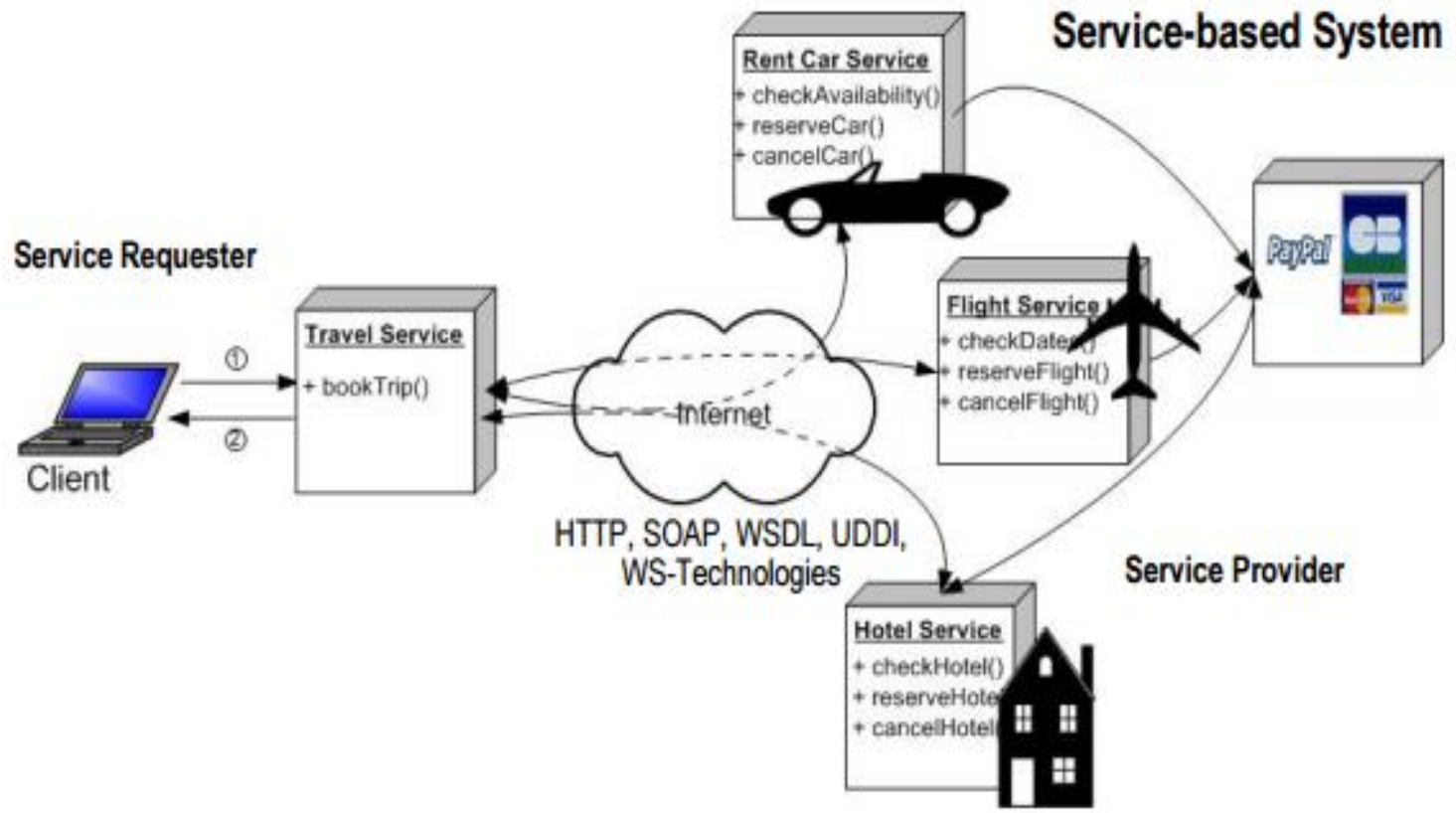
- **Design Pattern** is a general , reusable solution to a commonly occurring problem with in a given context in software design [32] .



- **Antipatterns are** commonly generated Solution of design/ code problems that may have negative impact [1].

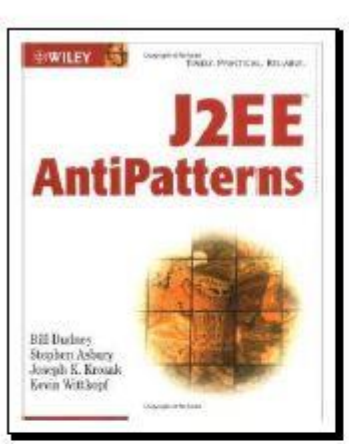


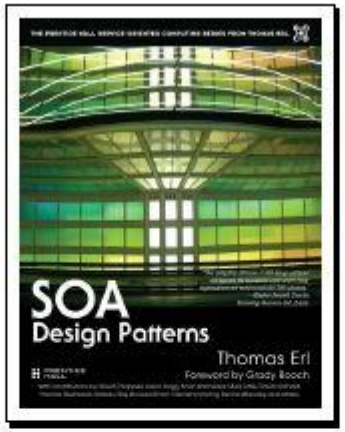
Benefits of Detecting Antipatterns from SOA

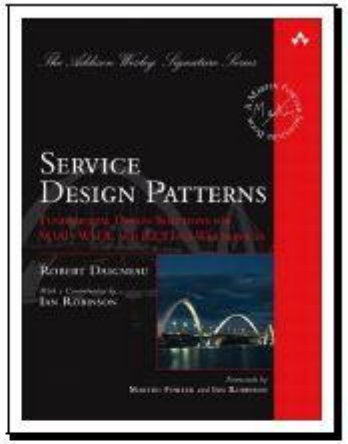


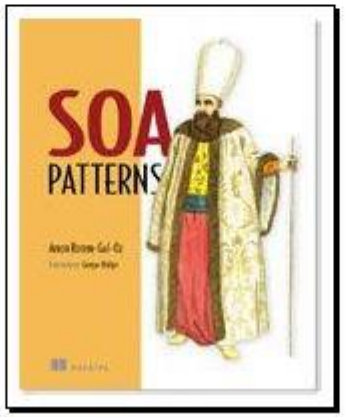
Follow the Link http://sofa.uqam.ca/soda-w/

Catalogs of Antipatterns for SOA

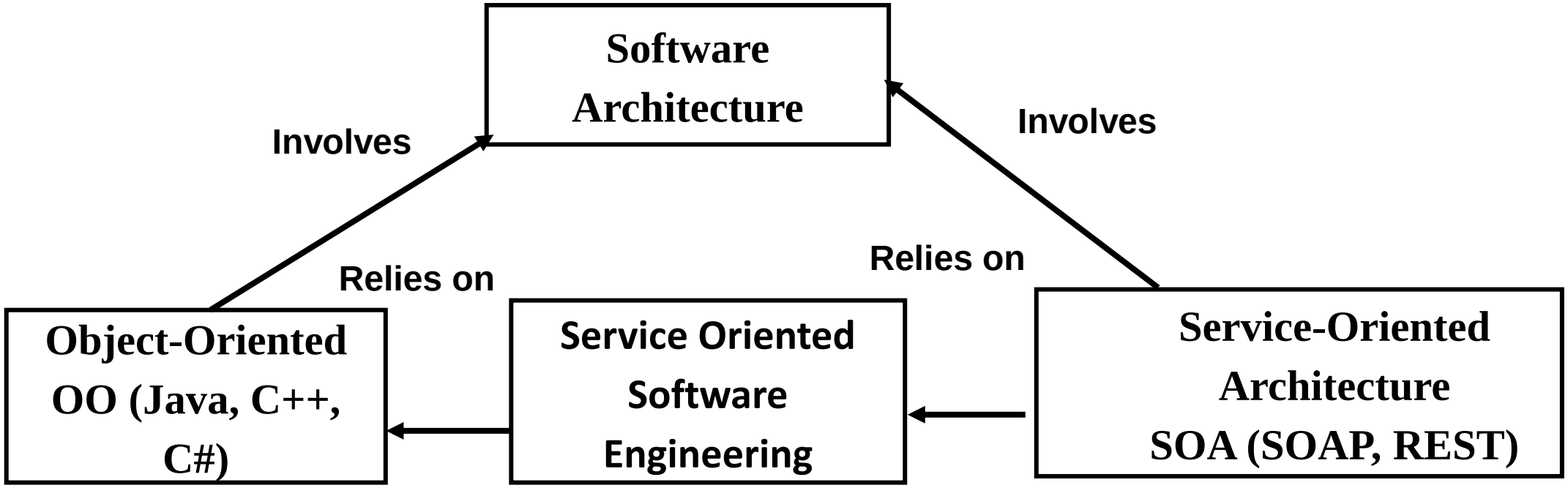








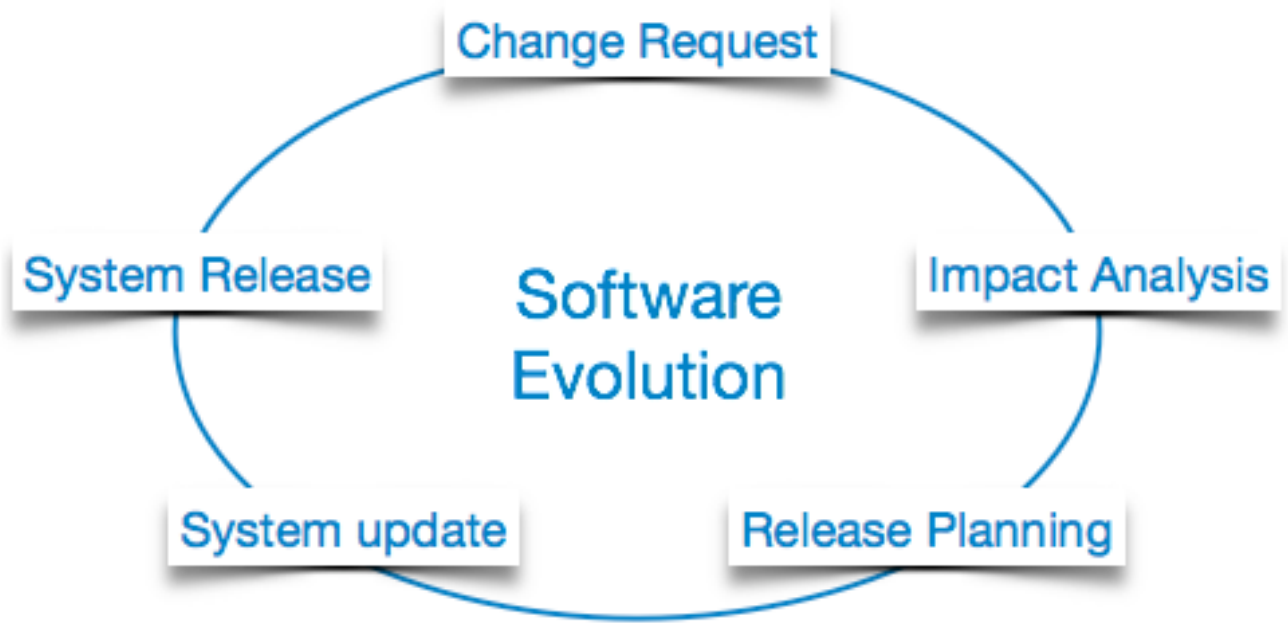
Relationship between Software Architecture and Programming Paradigm



Related Work

Software Evolution
OOSE vs SOA

Object Oriented Software engineering [10,11,12,13,19, 22,26,29,31]



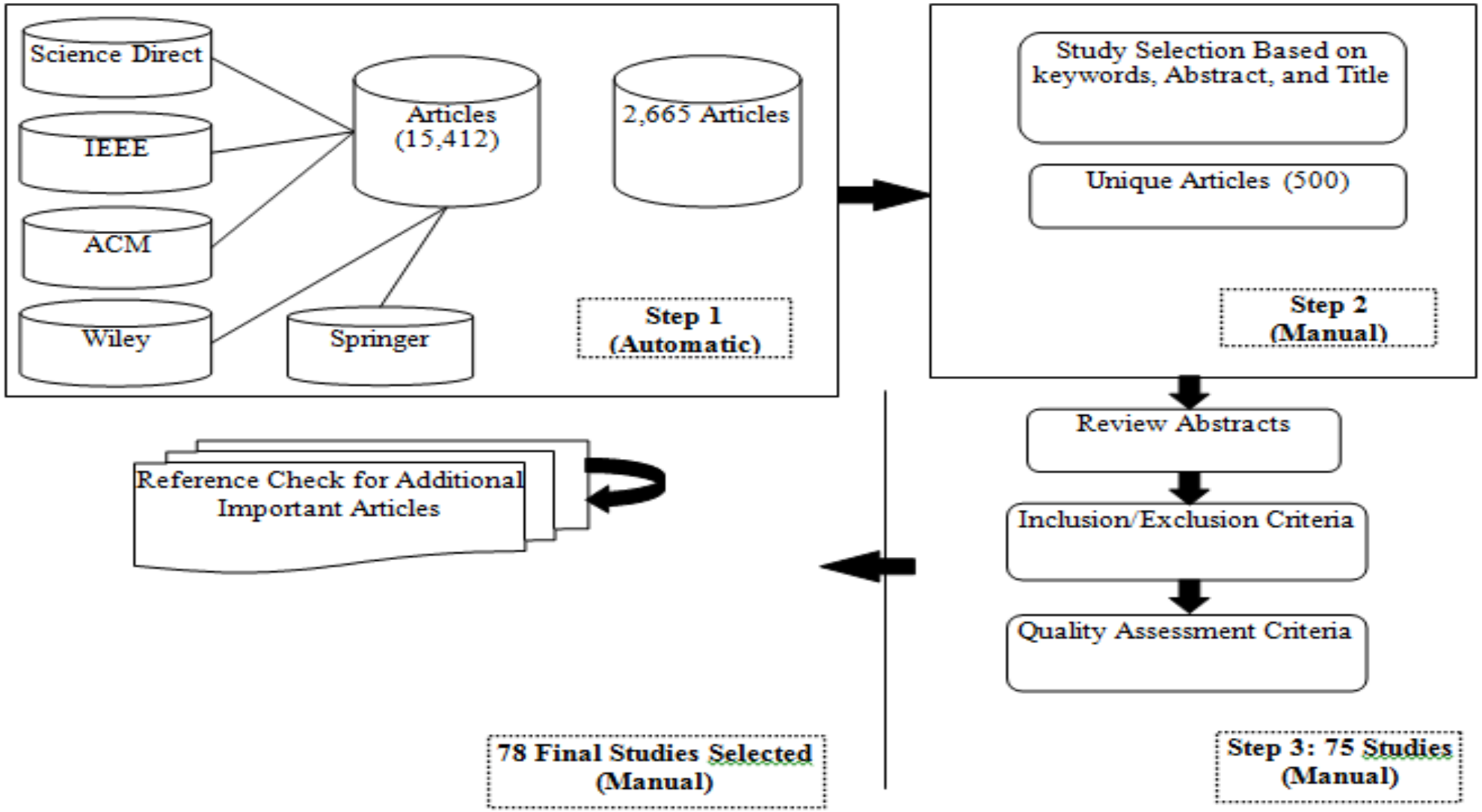
Service Oriented Architecture [21,23,25,30]

?

First Published papers on SOA Antipatterns reported in 2013 by Moha [2].

Palma, F., Nayrolles, M., Moha, N., Guéhéneuc, Y. G., Baudry, B., & Jézéquel, J. M. (2013). Soa Antipatterns: An approach for their specification and detection. ***International Journal of Cooperative Information Systems*****,** ***22*****(04), 1341004.**

Systematic Literature Review for Problem Identification



RQ1: ***What are the Classifications of the State-of-the-Art Techniques Employed in the Detection of Bad Smells?***

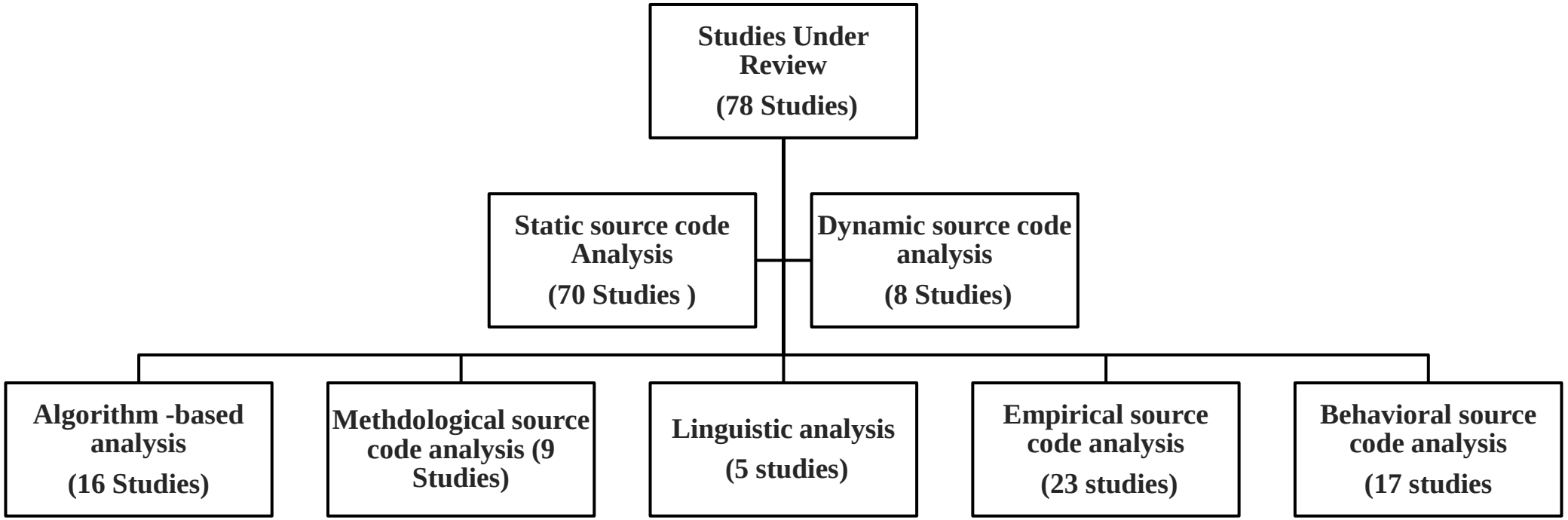
RQ2: ***How the State-of-the-art Approaches Evolved across Different Domains Starting from Object Oriented to Service-based Systems?***

RQ3: ***What are the Smells that are Reported for a Specific Domain?***

RQ4: ***What is the Correlation between Smells across Domains?***

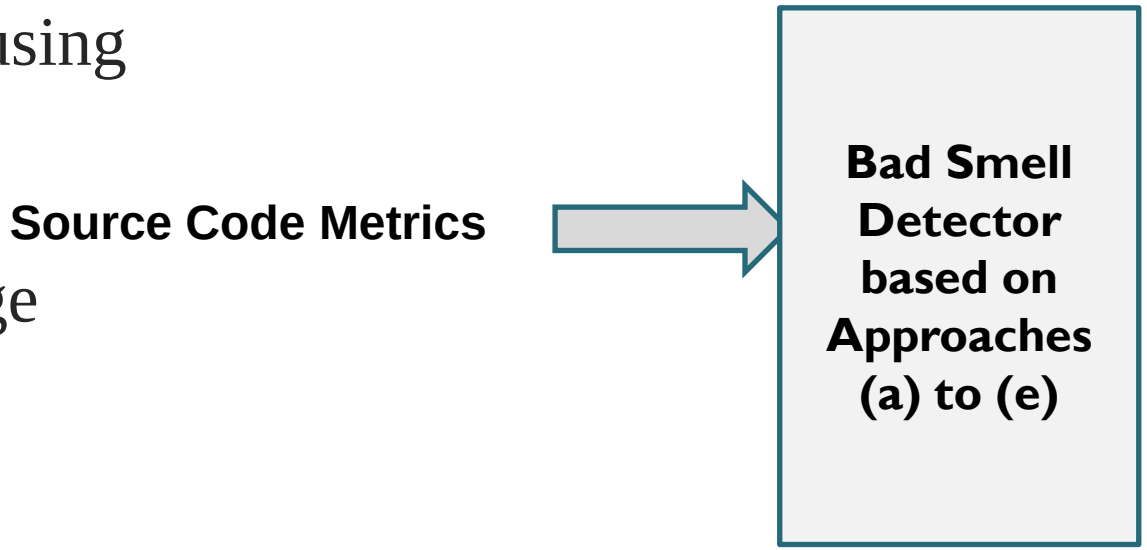
RQ5: ***What are the Trends in Research for Bad Smells from the Year 2000 to 2017?***

Finding of RQ1: Classification of State-of-the-Art Techniques



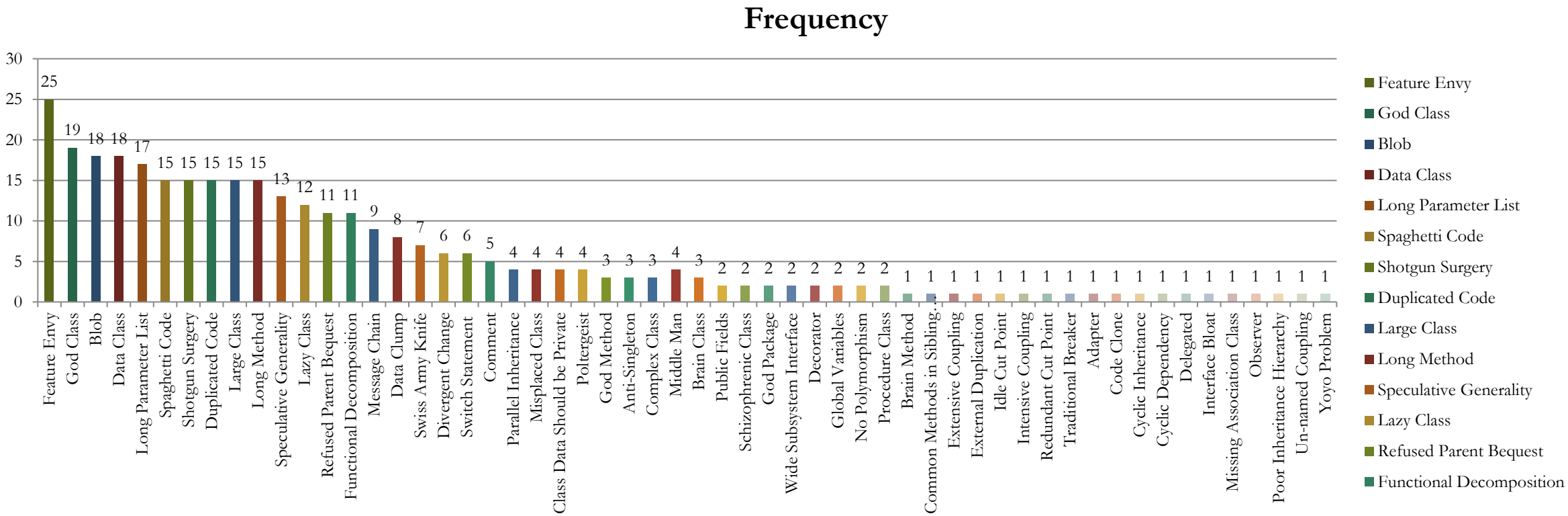
Findings RQ2:Evolution of State of the Art Approaches

a) Source Code Metrics
b) Mining the Source Code using SVN
c) Genetic Algorithm
d) Domain Specific Language
e) PE-A

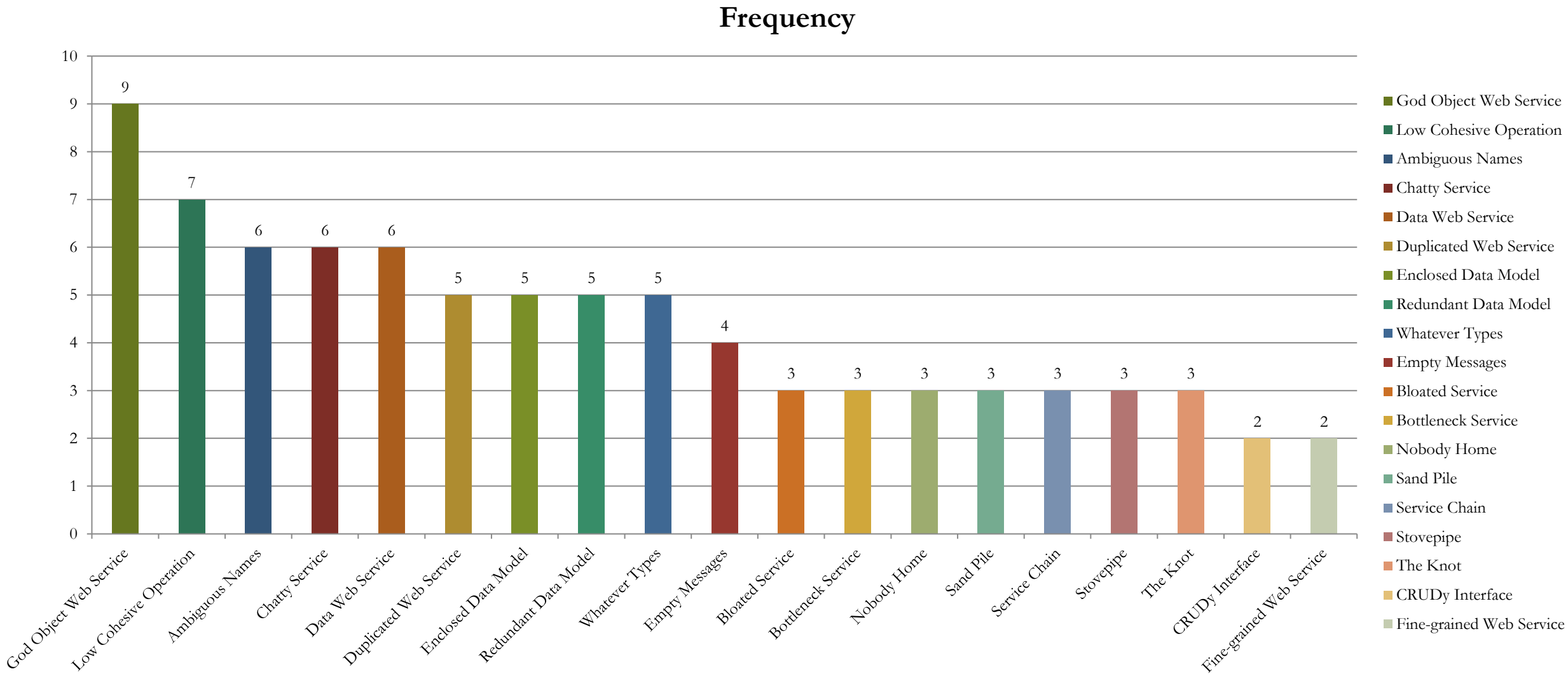


RQ3: Smells Reported most of the Time in Literature

a)OOSE



b)Smells Reported for SOA

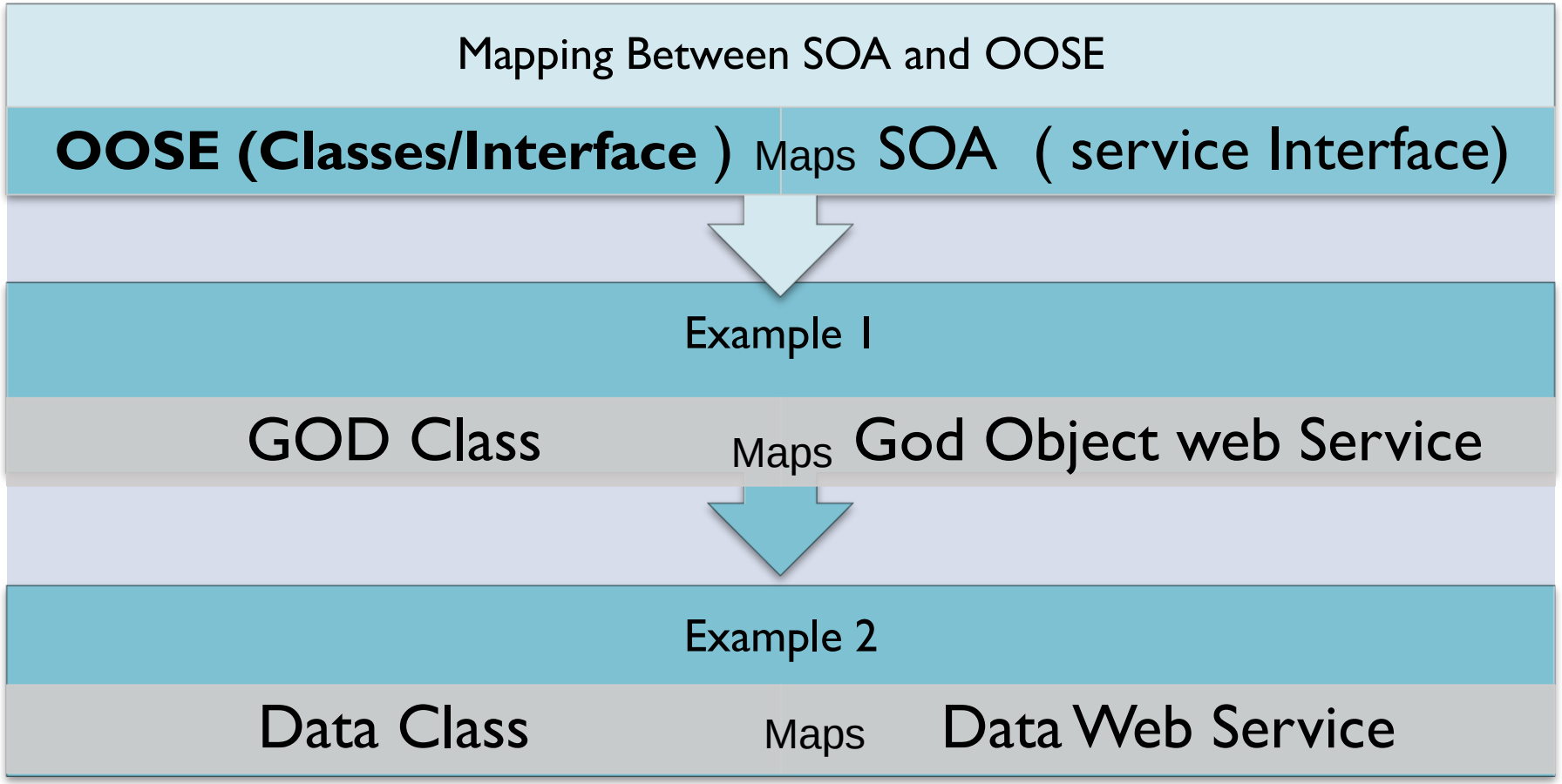


c) Smells Reported First time in Literature

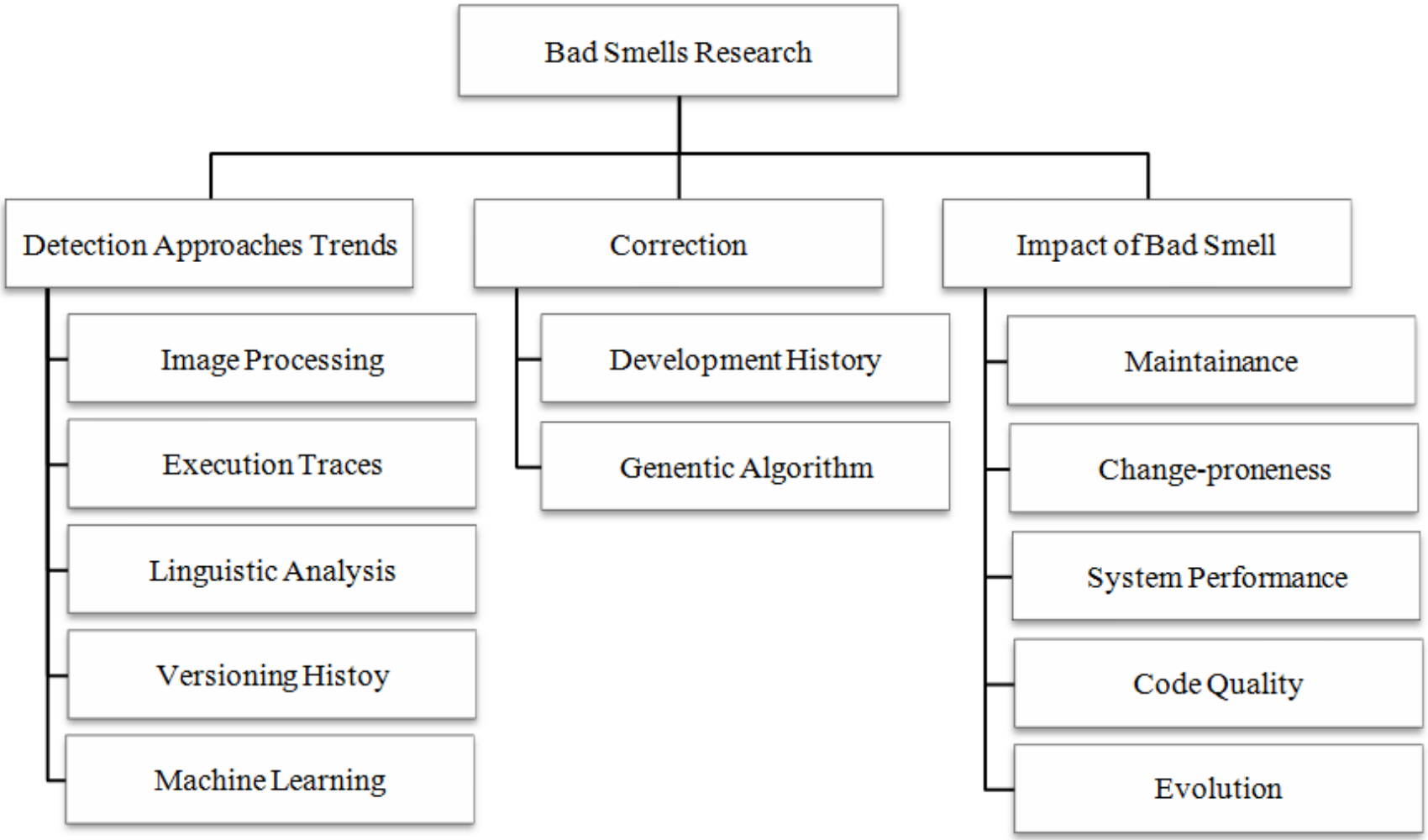
Ref	BSD	CN	CvCLRN	DNU	DOR	DSSS	FH	HvNHN	IC	IMT	ISC	ILC	LDFINT	LFDS	LFEBAOD	LFRDD	LSDASI	LFCWSDL	MC	MS	NS	RPT	RC	SINS	SPN	TVA	TTG	TTP	UCFISM	VCU
[5]		√					√		√	√	√	√				√	√	√		√	√	√		√		√	√		√	
[6]	√	√	√	√	√	√		√					√	√	√										√	√				√

*BSD (Breaking Self Descriptiveness), CN (Content Negotiation), CvCLRN (Contextualized vs. Contextless Resource Name), IMT (Ignoring Mime Type), IC (Ignoring Cache), ISC (Ignoring Self Descriptiveness), FH (Forgetting Hypermedia), HvNHN (Hierarchal vs. Non-Hierarchical Node), TTG (Tunneling Through Get), TTP (Tunneling Through Post), SPN (Singularized vs. Pluralized Nodes), MC (Misusing Cookies).

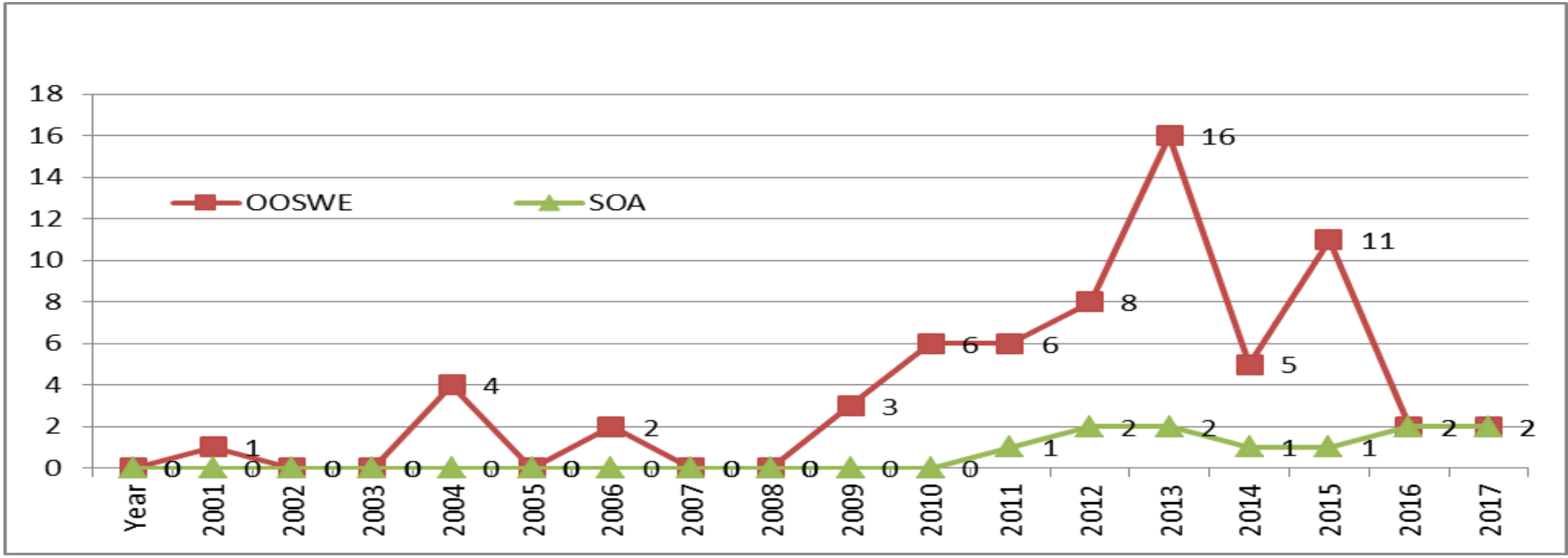
RQ4: What is the Correlation between Smells across the Paradigms?



RQ5: What are the Research Trends in the Domain of Bad Smells?

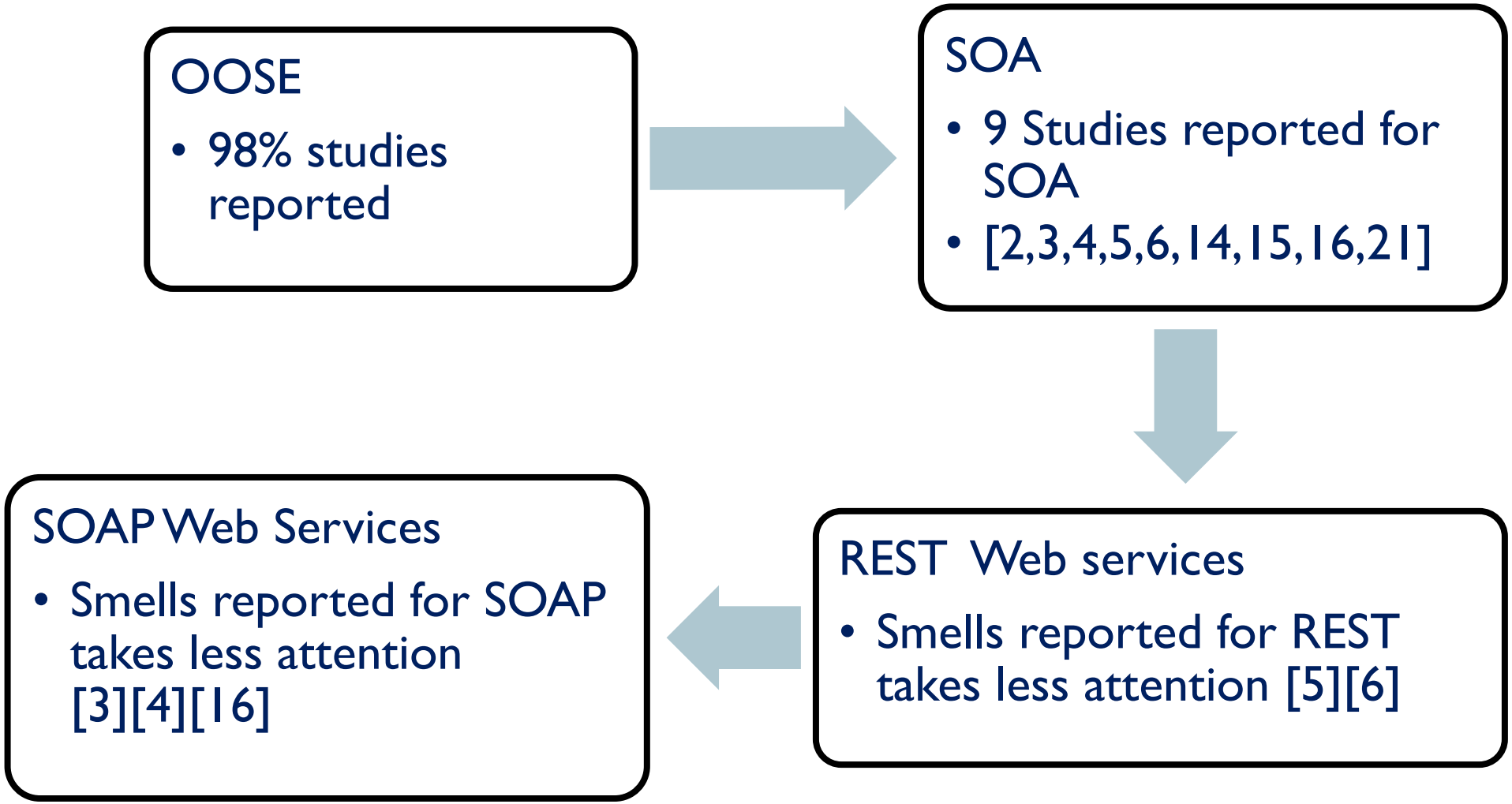


Findings from SLR



Sabir, F., Palma, F., Rasool, G., Guéhéneuc, Y. G., & Moha, N. (2019). A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems. Software: Practice and Experience, 49(1), 3-39. Impact Factor 1.33

3. Research Gaps



Problem Statement

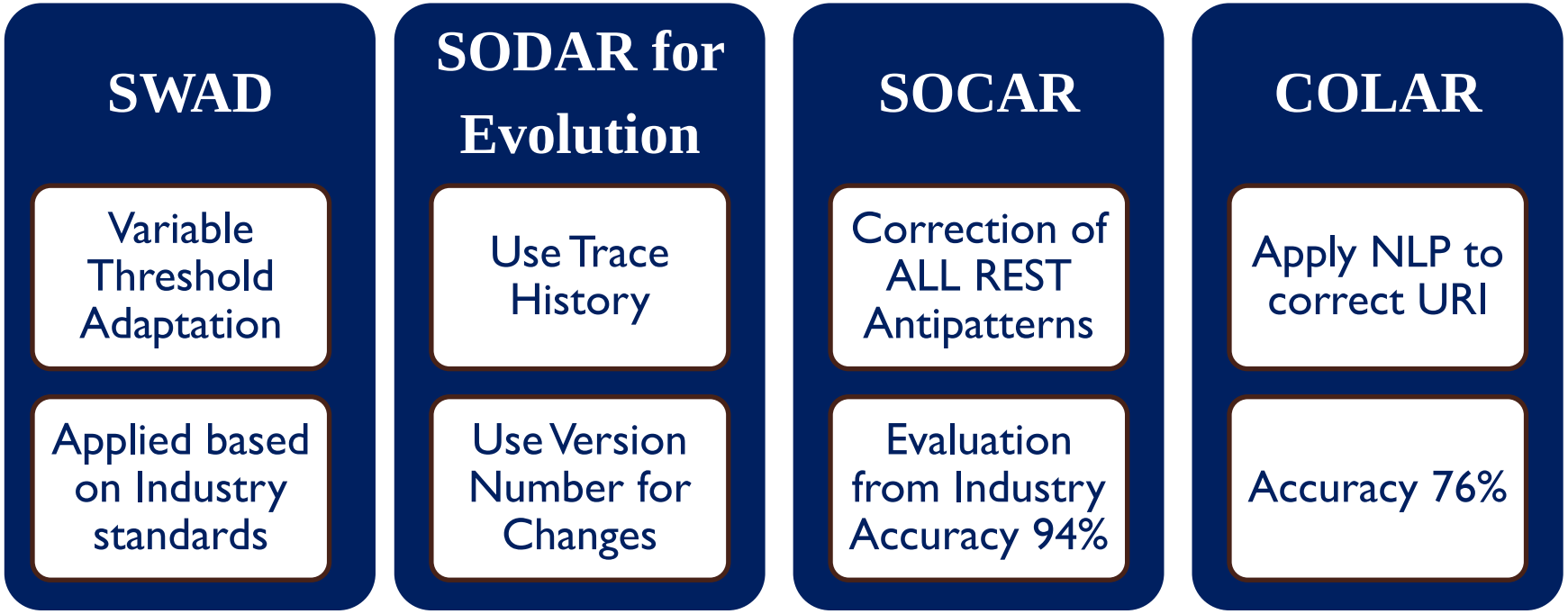
4. Problem Statement

Specification, Detection, Evolution and Correction of Antipatterns for Web Services

Problem 1	Variable threshold adaptation for multiple services
Problem 2	Evolution of Services
Problem 3	Correction of REST Antipatterns
Problem 4	Correction of REST Linguistic Antipatterns

Proposed Methodology

5. Proposed Methodology

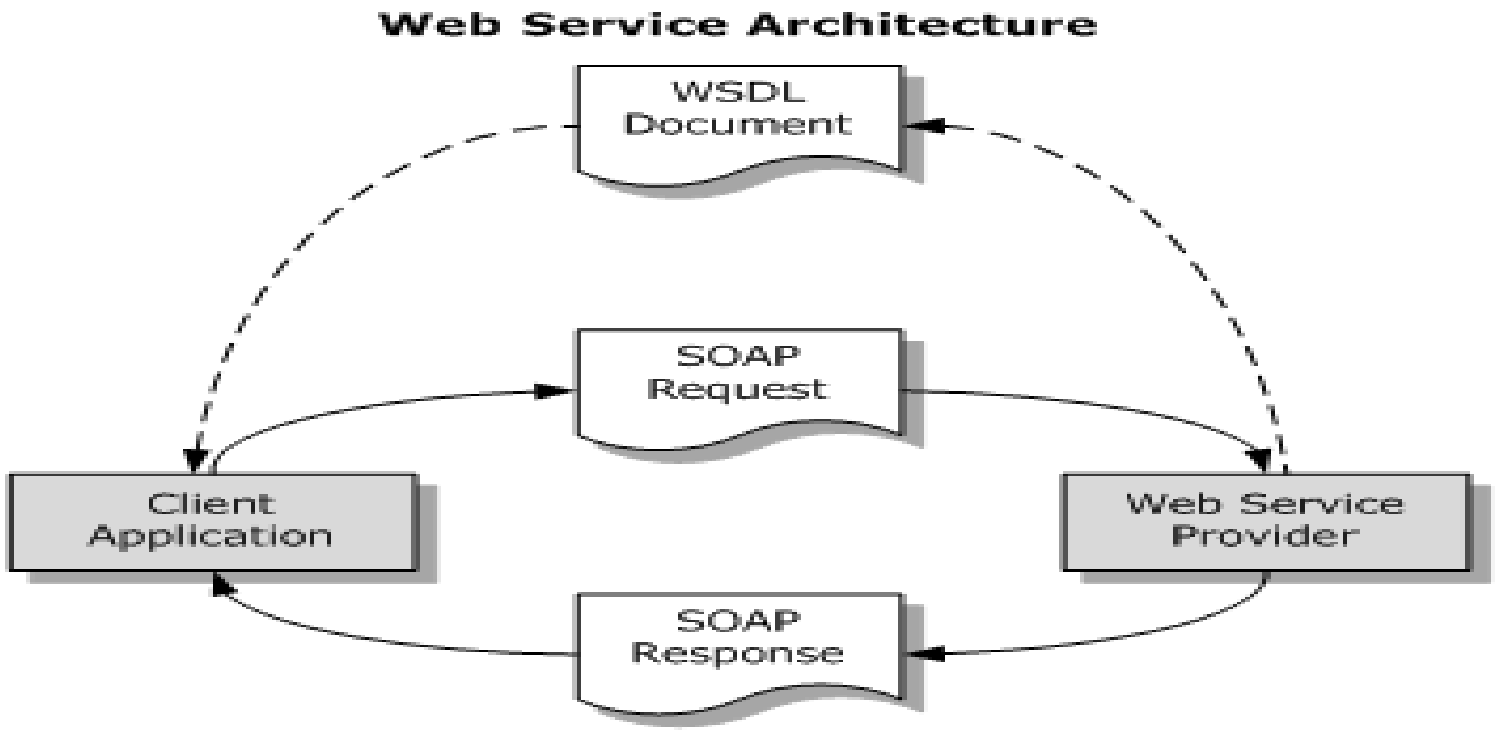


SWAD (Specification of Web Service Antipatterns Detection), SOCAR (Service Oriented Correction of Antipatterns for REST APIs), COLAR (Correction of REST Linguistic Antipatterns)

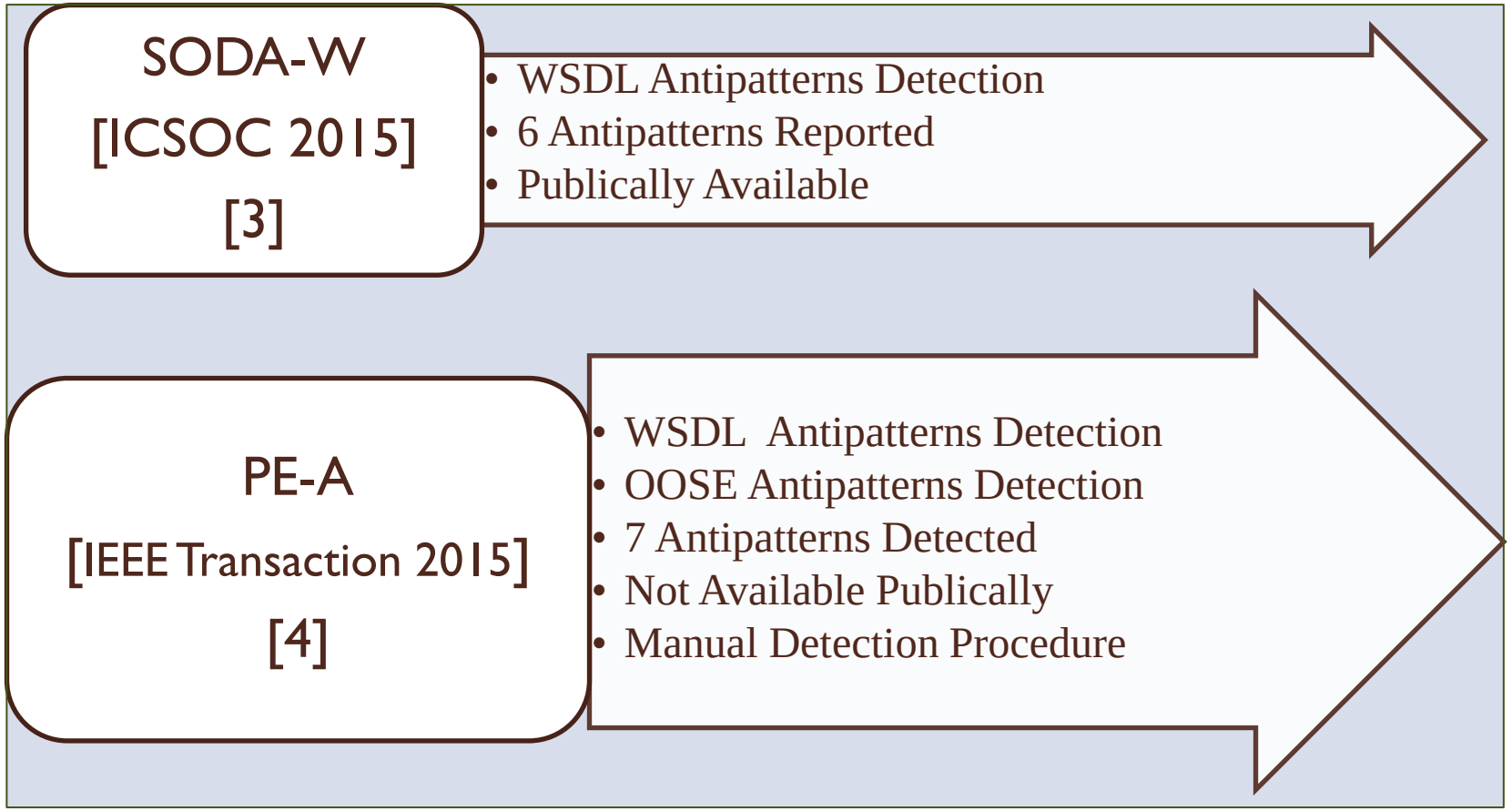
Available on SOFA.UQAM.CA
Available on PTIDEJ
COMSATS Software Engineering Research Group

Problem 1. Variable Threshold Adaptation from Multiple Technologies.

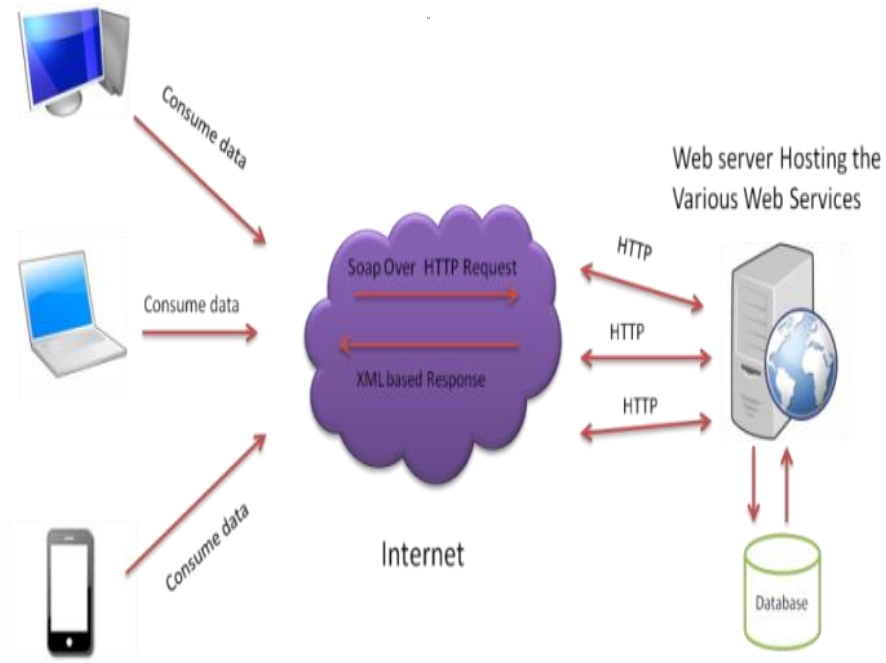
Specification and Detection from SOAP Web Services Antipatterns .

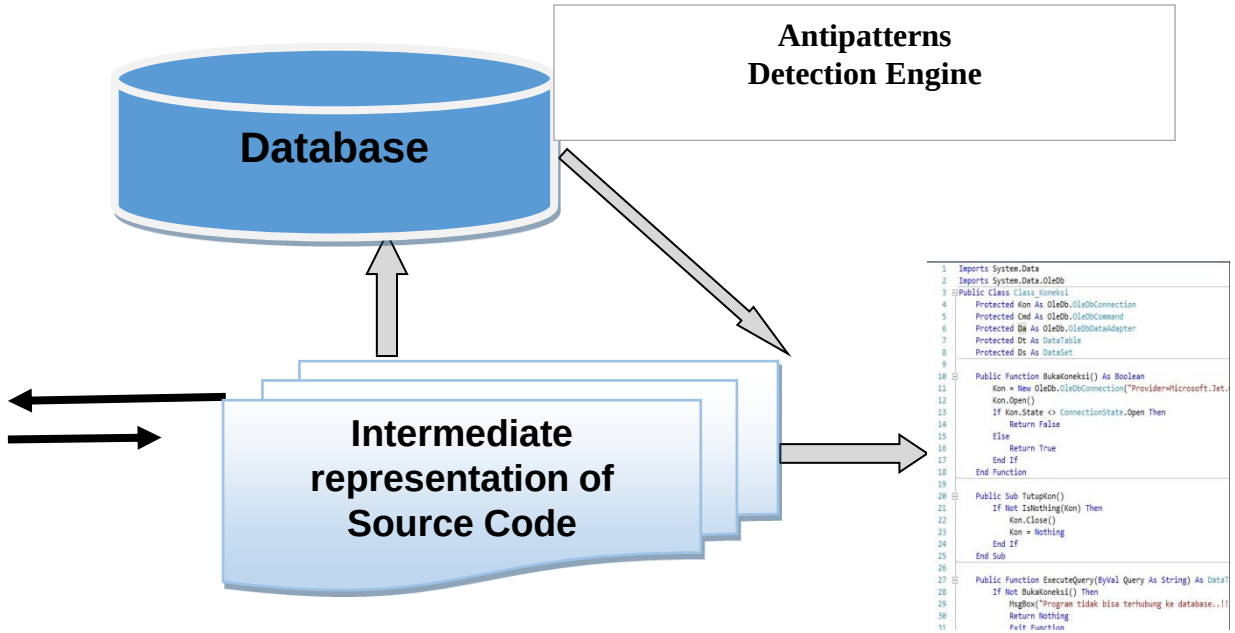


State of the Art Approaches
Antipatterns Detection from SOAP Web Services

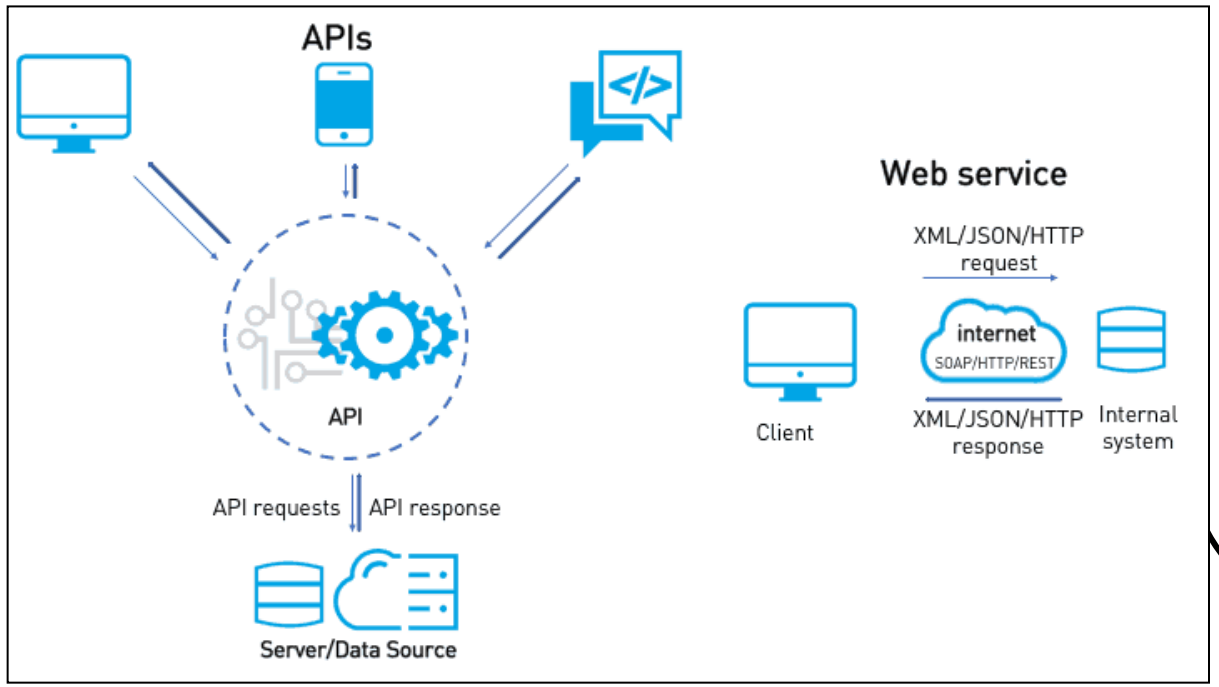


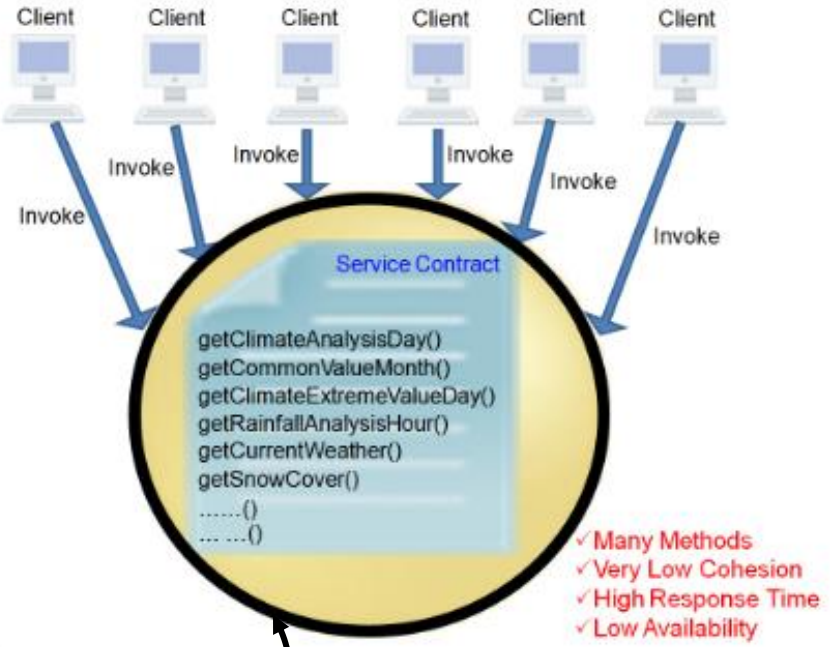
SWAD(Specification of Web Service Antipatterns Detection)

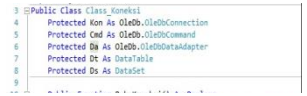




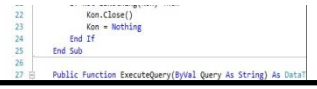
Example of God Object Web Service







Antipatterns Detection Engine



Antipatterns Detection based on rules

SWAD Key Features

- Variable threshold adaptation
- Multilingual analysis of Code-First and Contract-First
- Adaptability for NLP techniques for Chinese ,Korean and French Dialect.
- Industrial usage for Code-First and Contract-First
- Precision 89% and Recall 85%
- One new Antipattern reported in already existing Catalog.

Fatima Sabir, Ghulam Rasool, Maria Yousaf (2017), "A Lightweight Approach for Specification and Detection of SOAP Anti-Patterns ", International Journal of Advanced Computer Science and Applications, pp: 455-467, Vol: 8, Issue: 5, Standard: 2156-5570

Problem 2. Evolution of Antipatterns with Evolution of Services .

Forgetting Hypermedia Antipattern

DropBox Server Response 1:

```
Header: {
    x-frame-options=[SAMEORIGIN],
    x-dropbox-request-id=[b9a25269beb2c75fa7d7e21e1638bb9d],
    Connection=[keep-alive],
    Server=[nginx],
    pragma=[no-cache],
    cache-control=[no-cache],
    x-server-response-time=[64],
    x-dropbox-http-protocol=[None],
    set-cookie=[gvc=MjExODUyMTE…..
    expires=Tue, 26 Mar 2019 18:34:14 GMT],
    Transfer-Encoding=[chunked],
    Date=[Thu, 27 Mar 2014 18:34:14 GMT],
    Content-Type=[application/json],
    X-RequestId=[c64da98881e565a90a5dd9aecea9f049]
}

Body: {
    "hash": "f9d780e7655fe43261b4de9ec9a926eb",
    "revision": 2,
    "rev": "21e8a5a19",
    "thumb_exists": false,
    "bytes": 0,
    "modified": "Tue, 28 Jan 2014 21:45:31 +0000",
    "path": "/test",
    "is_dir": true,
    "icon": "folder",
    "root": "dropbox",
    "contents": [ {
        "revision": 3,
        "rev": "31e8a5a19",
        "thumb_exists": false,
        "bytes": 4,
        "modified": "Tue, 28 Jan 2014 21:46:30 +0000",
        "client_mtime": "Tue, 28 Jan 2014 21:46:30",
        "path": "/test/test.txt",
        "is_dir": false,
        "icon": "page_white_text"
```

No links to follow…

No links to follow…

DropBox Server Response 2:

```
Header: {
    x-frame-options=[SAMEORIGIN],
    x-dropbox-request-id=[cd12ele844327464485842b11b530071],
    Connection=[keep-alive],
    Server=[nginx],
    pragma=[no-cache],
    cache-control=[no-cache],
    x-server-response-time=[110],
    x-dropbox-http-protocol=[None],
    set-cookie=[gvc=MzIwNTkxODQzNjQy.....;
    expires=Sat, 06 Apr 2019 22:11:47 GMT;
    Transfer-Encoding=[chunked],
    Date=[Mon, 07 Apr 2014 22:11:47 GMT],
    Content-Type=[application/json],
    X-RequestId=[d509463440ada422459335fd3c71d309]
}

Body: {
    "referral_link": "https://db.tt/AaWjP9HP",
    "display_name": "Francis Palma",
    "uid": 118690394,
    "country": "CA",
    "quota_info": {
        "datastores": 0,
        "shared": 293074019,
        "quota": 2147483648,
        "normal": 1661304356
    },
    "team": null,
    "email": "francis.polymtl@yahoo.ca"
}
```

Links to follow…

Evolution of Antipatterns in REST APIs

- Detection of Antipatterns from REST APIs already reported in literature
- As per the findings of SLR no study reported for the evolution of Antipatterns across different REST APIs.
- There is a need to identify that how Antipatterns are evolved w.r.t time .
- How major REST API providers refactor them ?

Evolution History for Antipatterns Identification

RQ1 :When Antipatterns are introduced ?

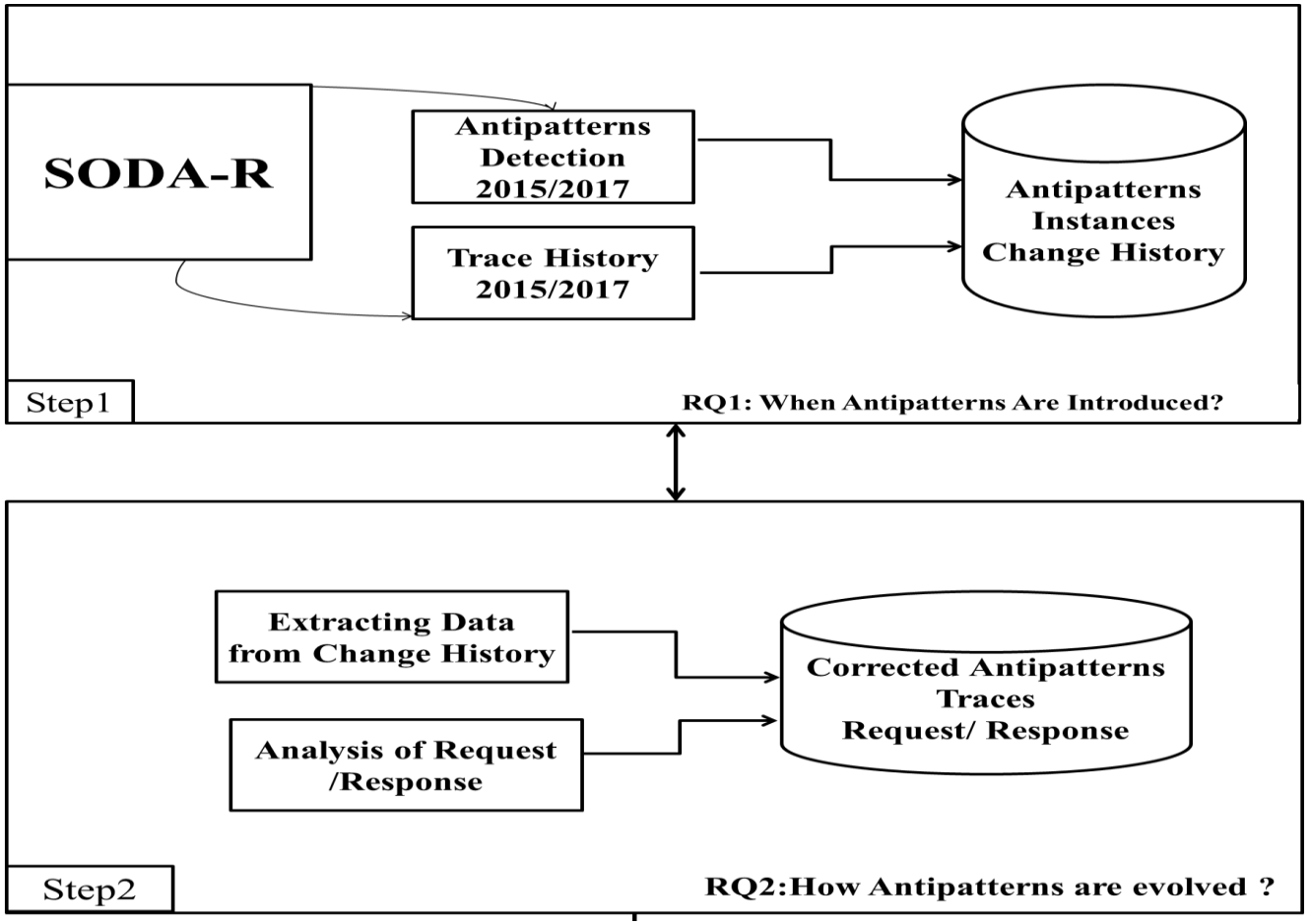
RQ2: How Antipatterns are evolved from 2015 to 2017 ?

RQ3:How Major Service providers remove Antipatterns?

Characteristics of Ecosystem Under Analysis

Sr.No	REST API Name	Available version	Monthly Active User	Online Changelog
1	Facebook.com	version 2.3 to version 2.10	1.94 billion	Facebook Changelog
2	Youtube.com	Revision history from 2013 to 2017	1 billion	youtube change history
3	Alchemy.com	version 1	Not Available	Alchemy API documentation
4	Bitly.com	version 3 is available	13,530	Bitly Documentation
5	Charlieharvey.com	Version 1	Not Avaialble	Charlihavery documentation
6	Ohloh.com	Version 1.0	669,601	Openhub documentation
7	musicgraph.com	version 2	1 billion	Music graph documentation
8	Dropbox.com	version 1 and version 2	500 million	Drop box change log
9	Instagram.com	version 2 is available with changelog	319 million	Insta gram changelog
10	Twitter.com	version 1 with complete changelog	600 million	Twitter Changelog
11	Teamviewer.com	complete complete change log	300 million	Team Viewer
12	Stackaexchange	version 2.0,2.1,2.2 is available	345 million	Stack Exchange Changelog

Research Methodology used for REST Antipatterns Evolution



When Antipatterns are Introduced?

Step 1

Sr.No	API Name	Year	Breaking Self Descrip-tiveness	Forgetting Hyper-media	Ignoring Mime Type	Ignoring Status Code	Ignoring Cache	Misusing Cookie	Tunneling	Relative Change (%)
1	Alchemy	2015	0	1	2	1	7	0	5	94
		2017	9	1	2	1	9	0	9	
2	Music Graph	2015	0	1	2	1	7	0	5	94
		2017	9	1	2	1	9	0	9	
3	Bitly	2015	0	2	3	0	0	0	2	243
		2017	0	5	15	0	0	0	4	
4	DropBox	2015	12	9	0	0	12	0	5	18
		2017	17	14	0	0	8	0	6	
5	Twitter	2015	10	3	9	0	0	0	0	225
		2017	25	6	25	6	14	0	2	
6	Youtube	2015	9	3	9	0	0	0	0	76
		2017	17	3	14	0	3	0	0	
7	CharliHavery	2015	4	0	4	0	0	0	0	50
		2017	12	0	0	0	0	0	0	
8	Facebook	2015	67	29	8	2	0	0	0	Not Applicable
		2017	21	21	12	2	4	4	0	

How Antipatterns are Evolved?
Step 2

API Name	Version Number	Breaking Self Descrip-tivenes S	Forgettin g Hyper-media	Ignoring Mime Type	Ignoring Status Code	Ignoring Cache	Misusing Cookie	Tunneling
StackEx change	2.0	0	19	53	0	0	0	1
	2.1	0	24	53	0	0	0	1
	2.2	0	26	53	0	0	0	1

Real Time Traces for Antipatterns Evolution

Service name: ca.uqam.sofa.alchemy.api.Alchemy
Method name: URLGetRankedNamedEntities
Path: /calls/url/URLGetRankedNamedEntities

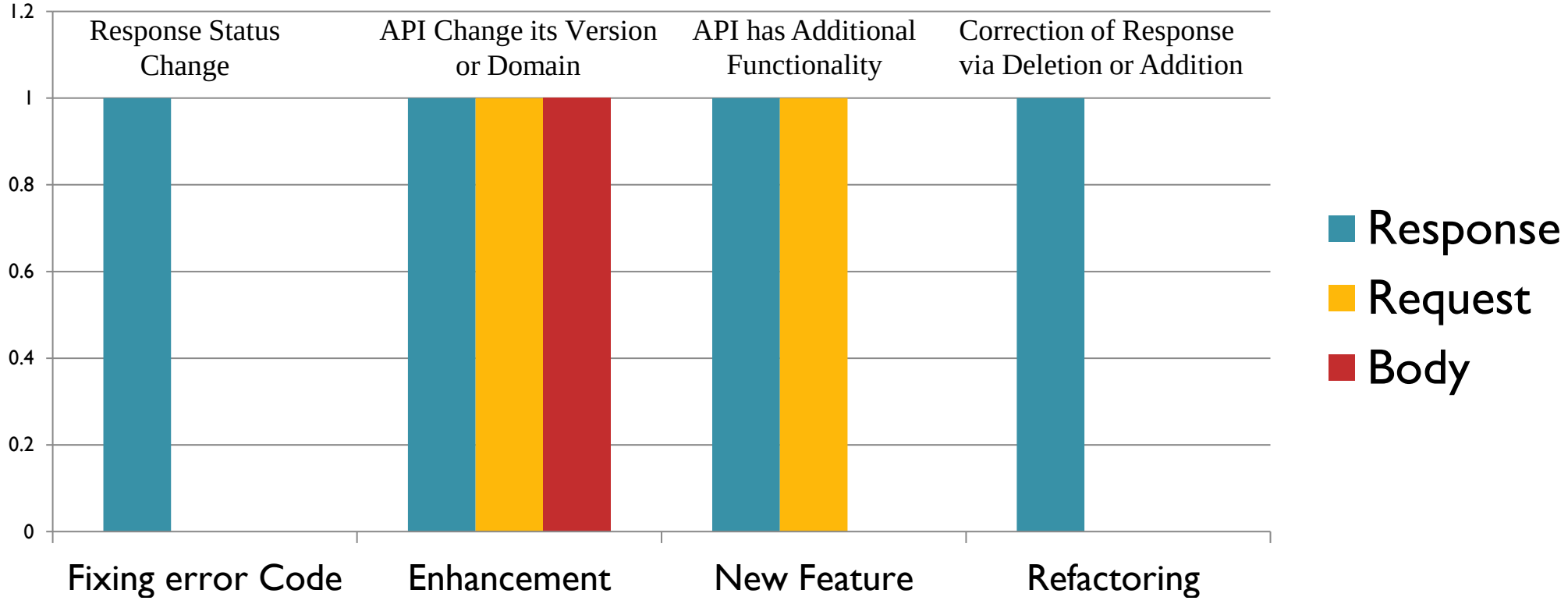
Response:
Status Code : 301

Header: {x-frame-options=[DENY], content-type=[text/html], connection=[keep-alive], etag=["595ebaa0-303"], location=[https://gateway-a.watsonplatform.net/calls/url/URLGetRankedNamedEntities

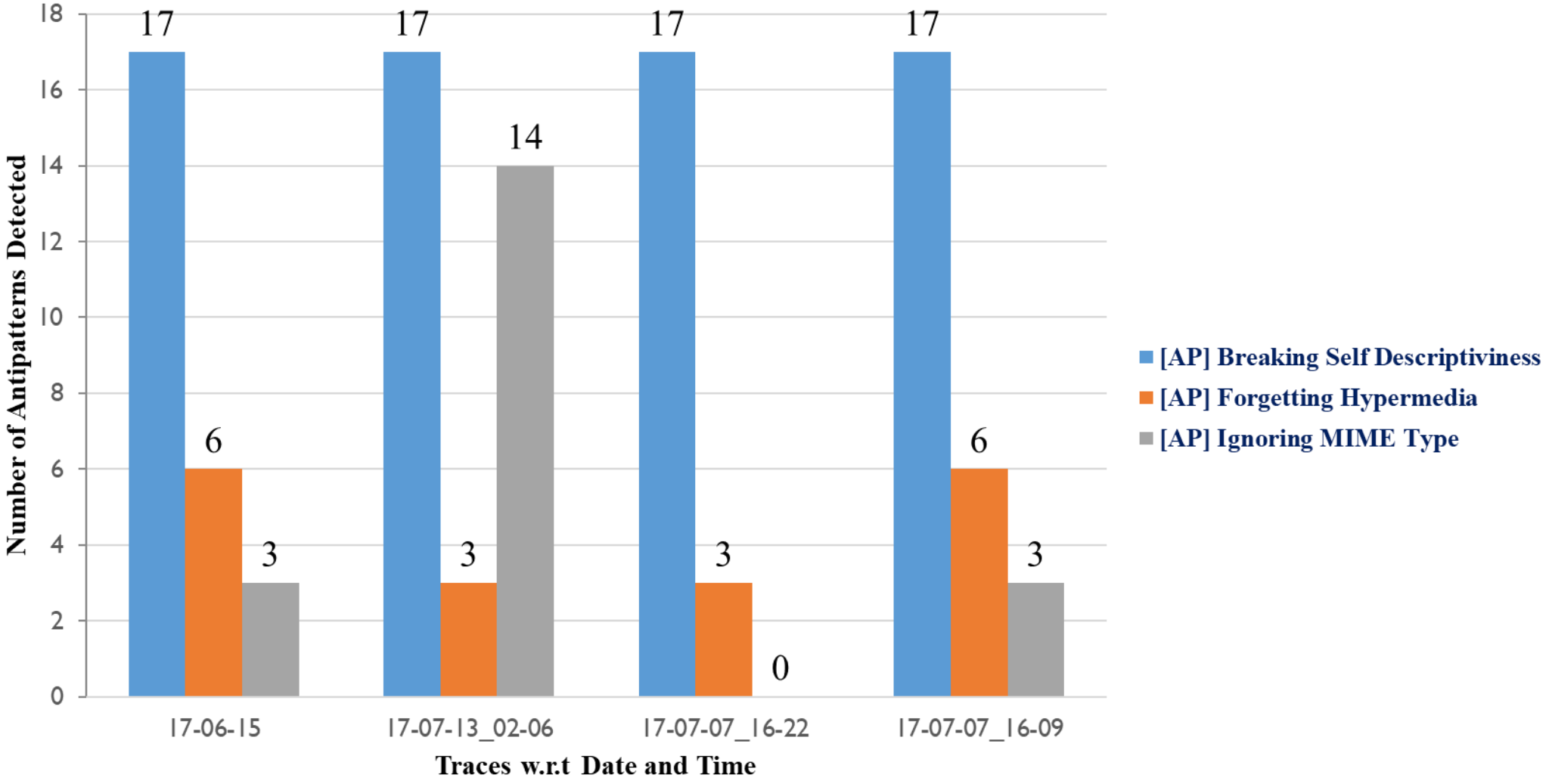
Body: <PRE>Dear AlchemyAPI User,This is an important reminder about critical action that needs to be taken before June 28th. To increase the security of our service, we are discontinuing HTTP support for all AlchemyAPI endpoints.
This update requires you to migrate all HTTP AlchemyAPI requests to HTTPS.
To ensure your AlchemyAPI code continues to work properly, replace all instances of "http://access.alchemyapi.com" with "https://gateway-a.watsonplatform.net". To give you another chance to migrate your AlchemyAPI requests, since our February notification, we have pushed this action deadline back to June 28th at 12:00pm MDT. If you have any questions or concerns, please contact the Alchemy support team at support@alchemyapi.com.Regards,IBM Watson</PRE>
Request:

Header: {cache-control=[no-cache], content-type=[application/xml], connection=[keep-alive], host=[access.alchemyapi.com], accept=[application/xml], get /calls/url/urlgetrankednamedentities?url=http%3a%2f%2fwww.cnn.com%2f2011%2f09%2f28%2fus%2fmassachusetts-pentagon-plot-arrest%2findex.html%3fhpt%3dhp_t1&apikey=01de81f60117e5a4d2880da5ede143b69b4ce973&outputmode=json http/1.1=[null], user-agent=[Apache CXF 2.7.5], pragma=[no-cache]}

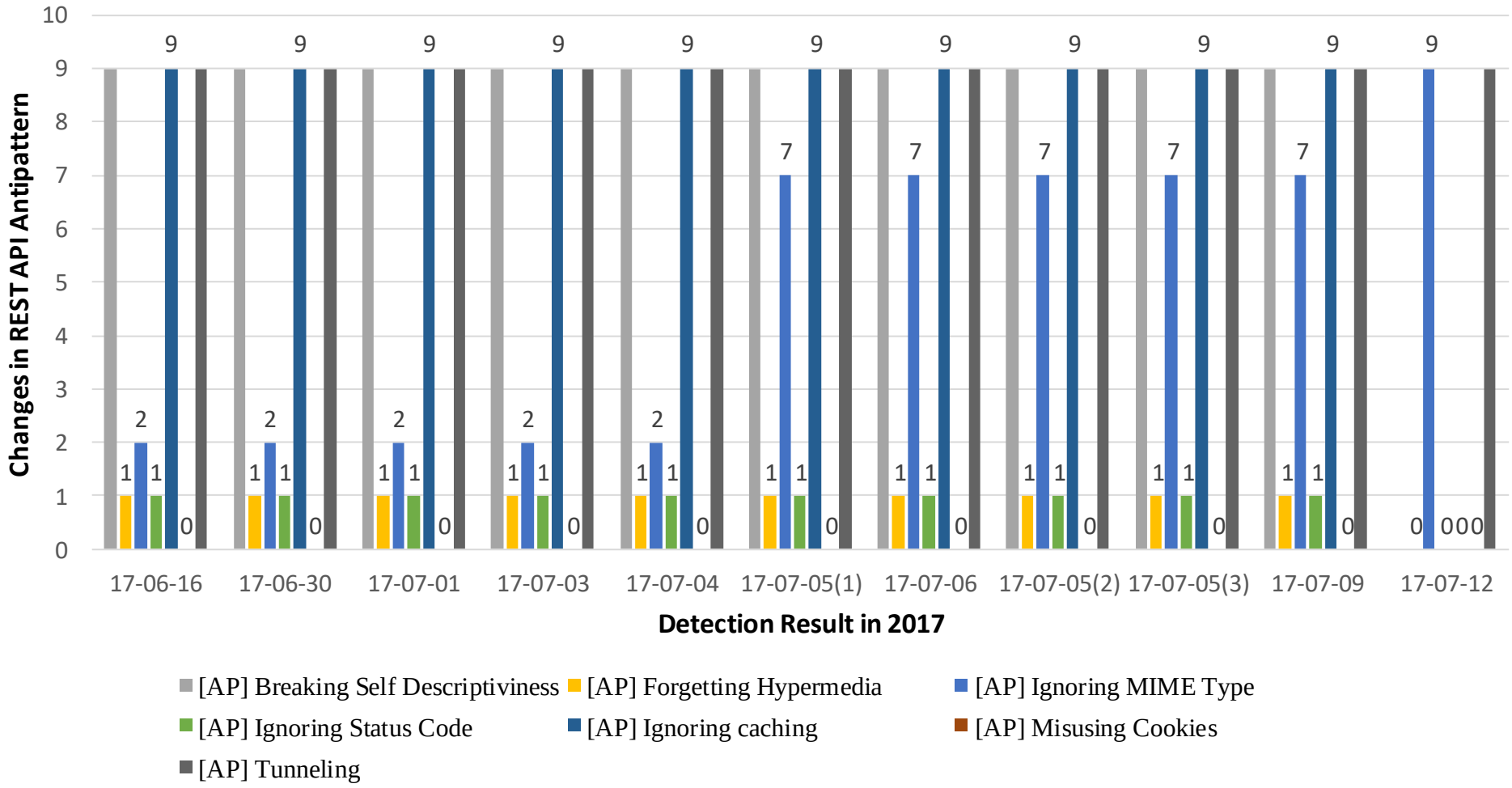
Operations Performed in Evolution



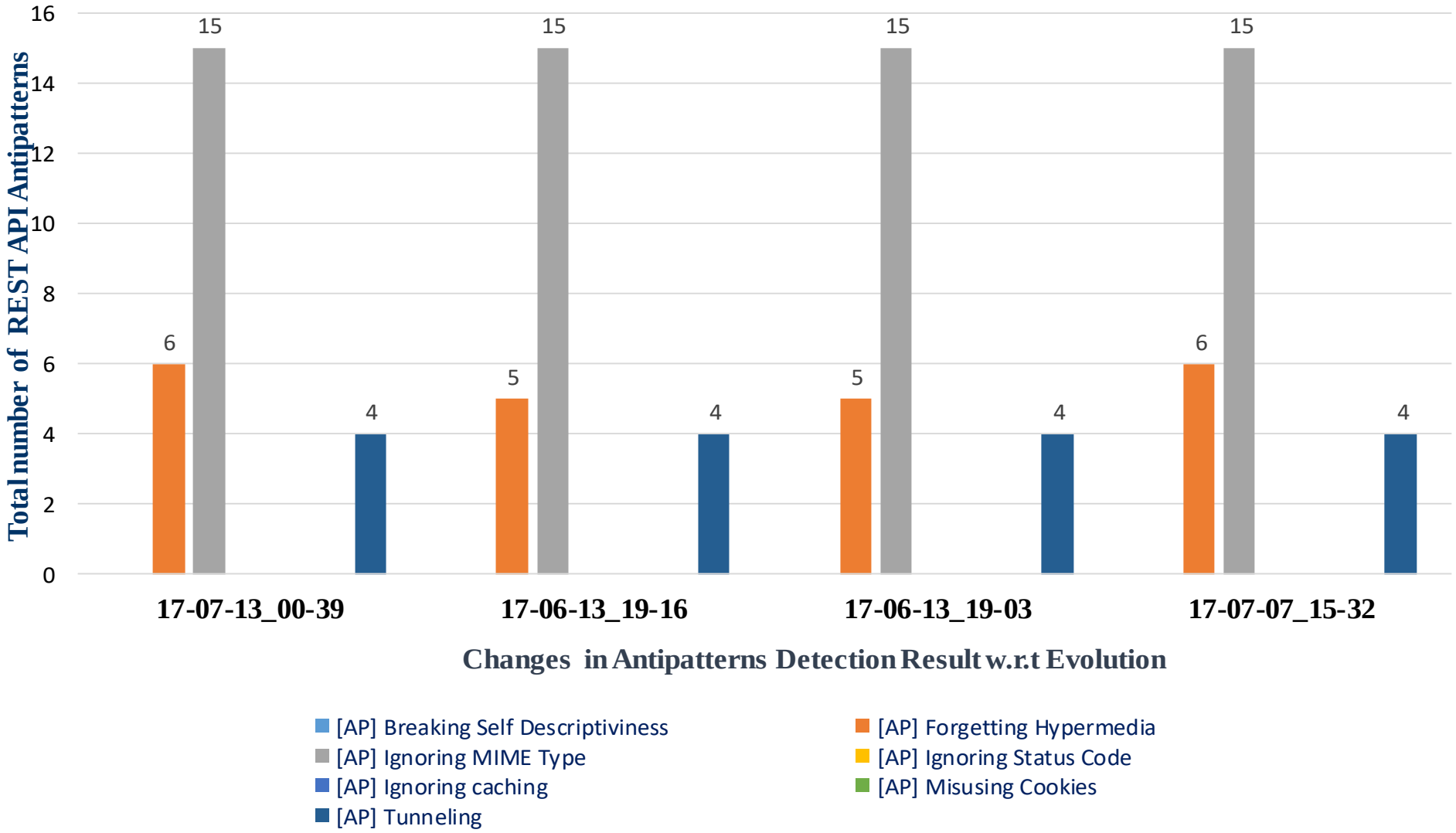
Antipatterns Evolution for Youtube API



Antipatterns Evolution for Alchemy API



Antipatterns Evolution for Bitly API



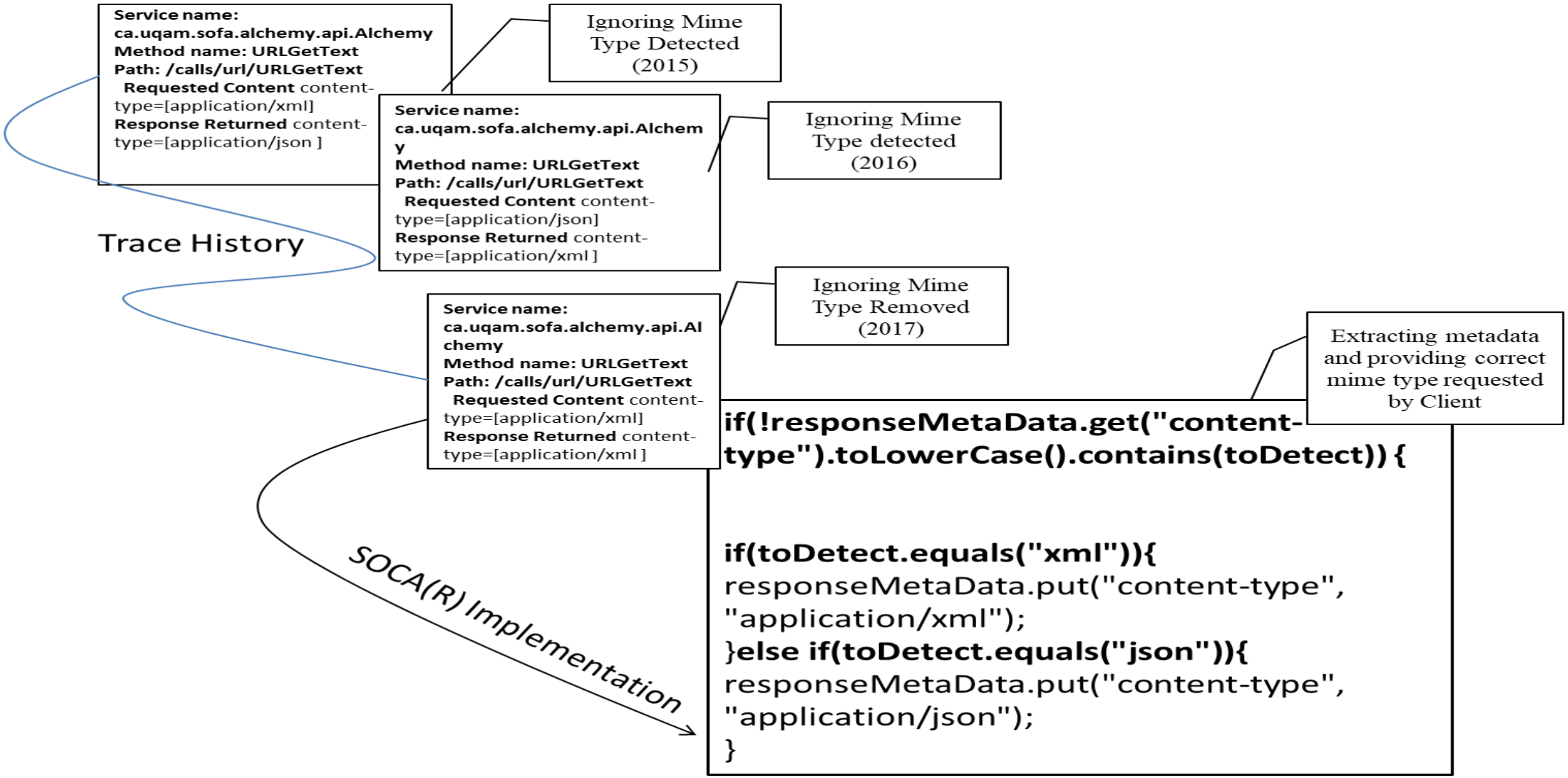
Findings

- Antipatterns are not removed by major service providers.
- Most of the Antipatterns are increased w.r.t time
- Correction of tunneling antipattern definition collected for Bitly.
- Correction of Mime type Antipatterns collected from Alchemy .
- Correction of Misusing Cookie and Breaking Self Descriptiveness from INFO Q.
- Real time traces will help to maintain request, response and body of the REST API..

Problem 3. Correction of Antipatterns for REST web Services.

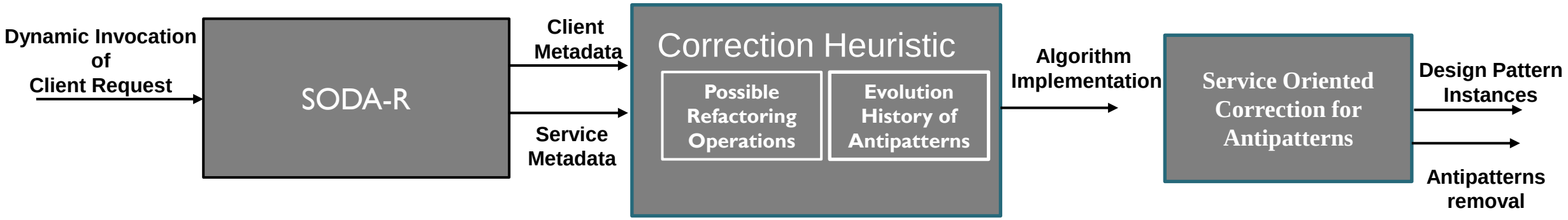
How Antipatterns are Corrected for REST APIs?

Evolution of Ignoring MIME Type Antipattern in Alchemy



SOCAR

Service Oriented Correction for Antipatterns for RESTful APIs



Refactoring Operations Performed for Antipatterns Correction

Sr.No	Antipattern Name	Properties	Refactoring Operations	Effect
1	Ignoring Mime Type	Accept,Content Type	Add Content-Type	Content Negotation Pattern
2	Ignoring Cache	Cache-Control,ETag	Add Cache Control, Generate Unique E-Tag for Request	Response Pattern
3	Forgetting Hypermedia	http-methods,entity link, location	add links,metadata info, status code	**Entity Link Pattern**
4	BreakingSelf-descriptiveness	Request-header field,response-header field	Remove non standardized Header from Response	**Antipattern Remove**
5	Ignoring Status Code	http method, status, stan-dardized status code de-scription,	Status code number and description change, replace method for code description and number	**Antipattern Remove**
6	Misusing Cookie	Cookie ,Set Cookie	Remove set-Cookie, cookie from response metadata	**Antipattern Remove**
7	Tunneling Through post	http-method,request-uri	remove access,update and delete from resource uri	**Antipattern remove**
8	**Tunneling Through get**	**http-method,request-uri**	**remove access,update and delete from resource uri**	**Antipattern remove**

Correction Algorithm For Forgetting Hypermedia Antipattern

```
Input:  http method ,entitylinks format,  entity location
Output:  Entity Link Design Pattern
http method←—Get, Put, Post, Delete;
entity links format ←—xml,json,pdf,rdf.; entity
  link←—link, location;
if response.getstatus()= "4K" Or response.getstatus()=
  "5K" then
 |   remove from detection and correction ;
end
if response.getBody() and response.getmetadata() ≠ Null
  then
 |   if checkLinksBody()←—xml,json,pdf,rdf then
 |  |   if checkLinkMetaData()←—location,link then
 |  |  |   Link detected ;
 |  |  |   if
 |  |  |     checkLinkMetaData().contains≠location,link
 |  |  |     then
 |  |  |  |   metadata.add(Add URL dynamically);
 |  |  |   end
 |  |   end
 |   end
end
  " Forgetting Hypermedia Antipattern Correction
Adding link in response meta data"
"Entity Link Pattern detected"
```

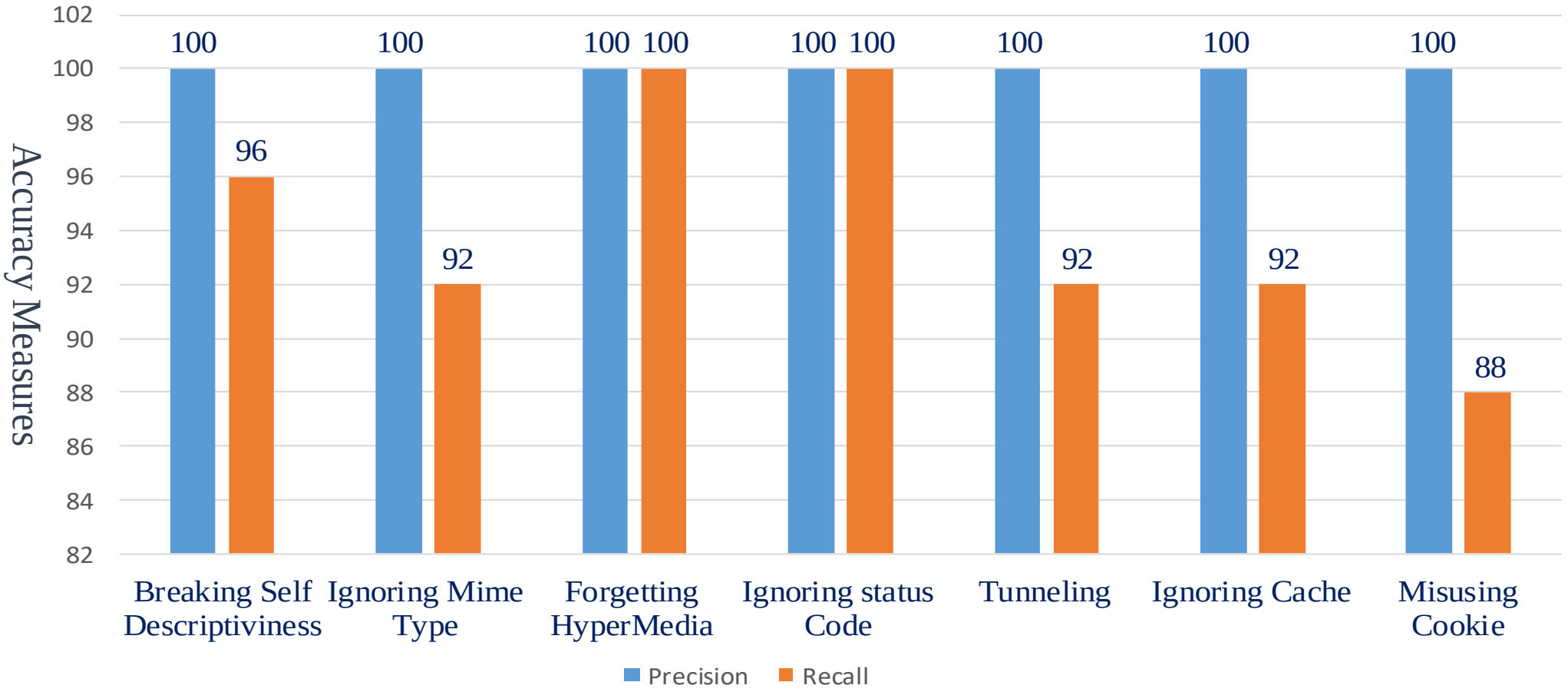
Contributions

- Real time traces of two years that provide information of different attributes used by REST API providers.
- A complete list of error code with Description available online.
- Content description of each REST API providers is also available online.
- Mining of trace history will be available on Github that can be used by industry/ academia.

RESULTS


AlchemyAPI
BEST BUY
bitly
CHARLIEHARVEY
MusicGraph
Ohloh
TeamViewer
YouTube
Zappos
Instagram
StackExchange

Accuracy of SOCA-R



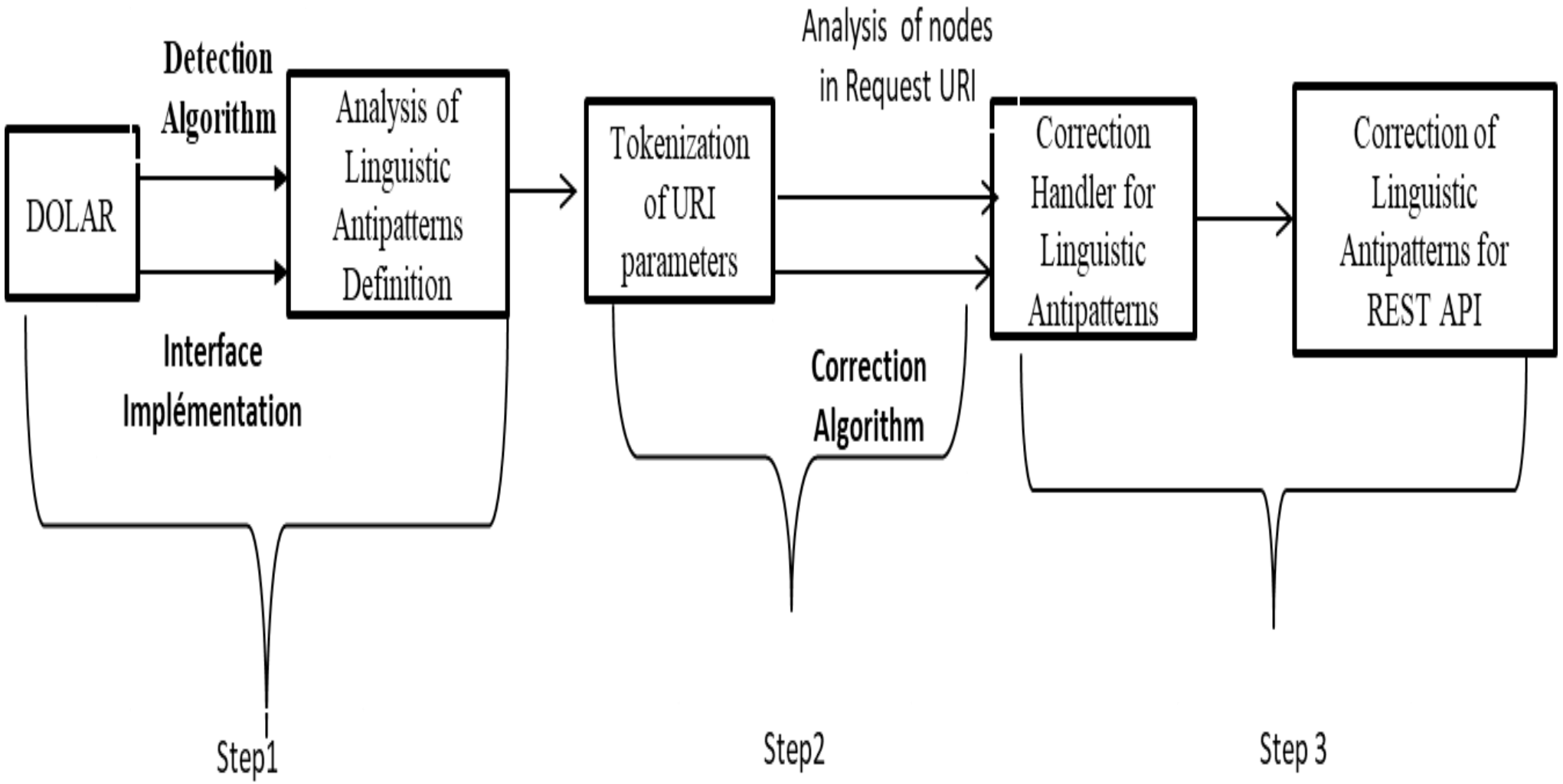
Average Precision 100%
Average Recall 94%

Problem 4. Correction of Linguistic Antipatterns for REST Web Services

Linguistic Antipatterns Problem

- Linguistic Antipatterns Detection and Correction is reported in literature for OOSE[7,8,9].
- Linguistic Antipatterns detection is Reported by Palma with a tool support DOLAR for REST APIs [6].
- Linguistic Antipatterns Correction is not reported for REST APIs.

Research Methodology Used for Correction



Linguistic Antipatterns for REST APIs

- Definition of Linguistic Antipatterns
  - **Singularized vs. Pluralized Nodes**
- Example of Linguistic Antipatterns
  - **https://www.abcexample.com/university/faculty/profile**
  - **Corrected URI Example**
  - **https://www.abcexample.com/university/faculty/profiles**

Corrected Trace of Linguistic Antipatterns for Youtube

```
https://www.googleapis.com/youtube/v3" /videos/rates
```

Refactoring operation “replace ” performed

```
https://www.googleapis.com/youtube/v3"
/videos/rate
```

Tidy vs Amorphous URI Example

Amorphous URI Antipattern

https:// *www.abcexample.com/University/Faculty/pic.jpg*

Replace and Remove Refactoring Operations Performed

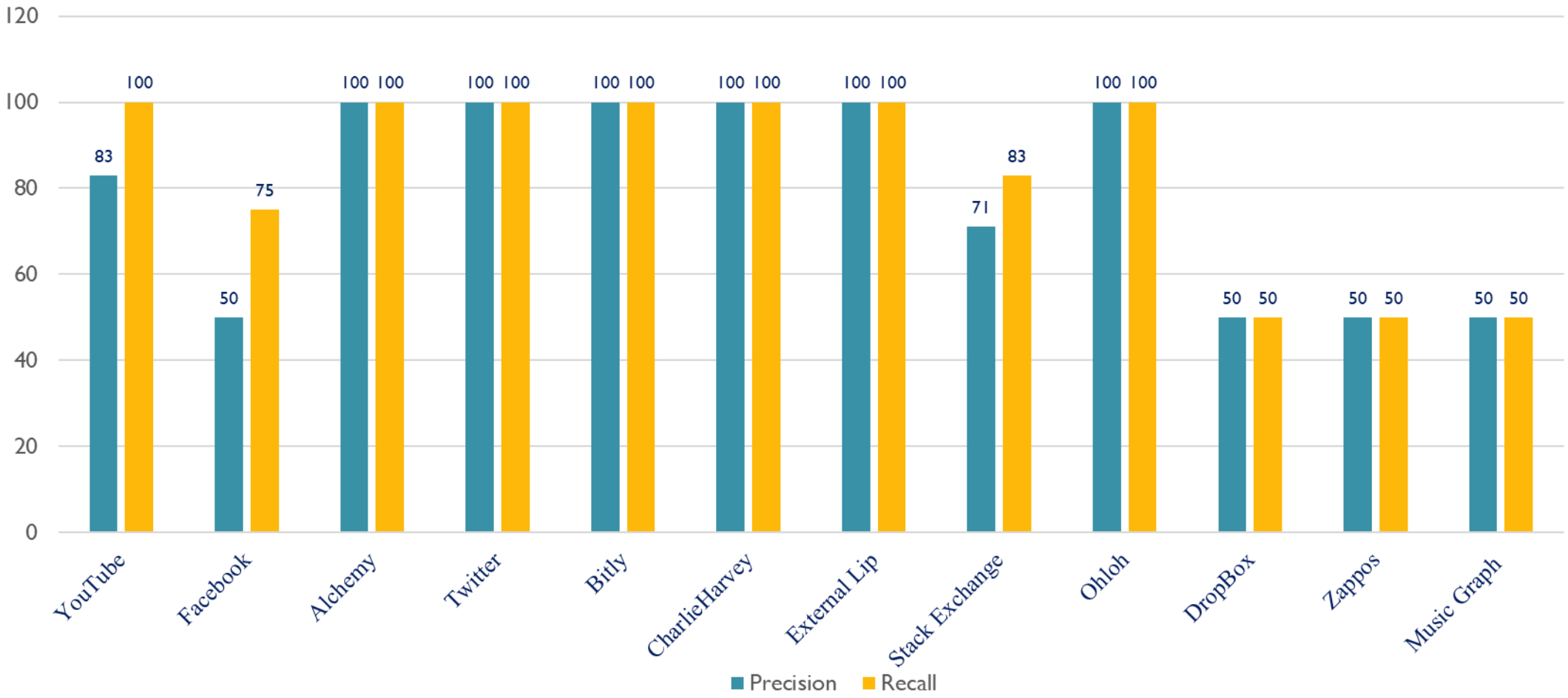
Tidy URI Design pattern

https:////*www.abcexample.com/university/faculty/profile/biodata*

Amorphous Antipattern Correction Algorithm

```
   Input:  URIparameters
   Output:  Tidy Design Pattern
 1 URIparameters ←— Hyphenated URI, UpperCase URI, Trailing Slash
     URI, Extension URI ;
 2 if URIparamters "contains" Hyphen then
 3 |   correctedURI= path.replace(_ ,-) else
 4 |   if URIparameters contains UpperCase URI then
 5 |   |   correctedURI= correctedURI.toLowerCase
 6 |   |   if URIparameters contains Trailing slash URI then
 7 |   |   |   correctedURI= correctedURI.remove (/,\\)
 8 |   |   |   if URIparameters contains Extension URI then
 9 |   |   |   |   if correctedURI.contains(extension.toLowercase() OR
   |   |   |   |     extension.touppercase()) then
10 |   |   |   |   |   correctedURI=correctedURI.remove(extension)
11 |   |   |   |   end
12 |   |   |   end
13 |   |   end
14 |   end
15 end
16   " Amorphous Antipatterns removed "
17 "Tidy Design pattern detected"
```

Accuracy of SOCAR



Average Precision 79%
Average Recall 75%

6. Conclusion

1. SWAD detect Antipatterns for industry both for code-first and contract-first.
2. A new antipattern introduced in the list of already defined Antipatterns
3. Evolution History is covered for REST APIs versions .
4. This history reports original traces and logs that can further be used in the field of Data mining.
5. SOCA-R is available that correct all famous REST API providers problems dynamically.

6. Standardized list of Request and Response codes available in XML form at University of Qubic Montreal web site.
7. SOCAR tool is open source and can easily be used by the new researchers for implementation of text mining techniques for improved URI .
8. SOCAR use dynamic source code analysis so services can be analyzed at any time incase user find problem to access specific services.
9. SWAD helps to detect the location of errors in services . This will inform developers to find bugs at early stages.
10. COLAR tool helps to improve the URI of REST APIs for better analysis.

7. Future Directions

- We have real time traces till end of the year 2017. This trace history may be extended and can be used for the identification of error prone services.
- The clients of REST API providers can be improved by removing the falws from major service providers. If the services are error prone then are the clients too?
- We are planning to extend SOFA(service Oriented Framework for Analysis) to compare the maintenance cost of effected services as compare to those that are not effected.
- We are planning to use SWAD tool for cross cultural analysis amongst SOAP API providers.

8. List of Publications

1. Fatima Sabir, Ghulam Rasool, Maria Yousaf (2017), "A Lightweight Approach for Specification and Detection of SOAP Anti-Patterns ", International Journal of Advanced Computer Science and Applications, pp: 455-467, Vol: 8, Issue: 5, Standard: 2156-5570 . (ISI Index).
2. Fatima Sabir, Francis Palma, Ghulam Rasool, Yann-Gaël Guéhéneuc, Naouel Moha (2018), "A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems", Software: Practice and Experience, pp: 1-37, Standard: 1097-024X, Impact Factor: 1.33
3. Fatima Sabir,Ghulam Rasool, ,Francis Palma, Yann-Gaël Guéhéneuc, Naouel Moha, Hassan Akhtar (2019). Correction of REST Antipatterns using Evolution History. Submitted in Journal of Empirical Software Engineering .Impact Factor 2.933 (under review).

References

1. Fowler, M. (1999). *Refactoring: improving the design of existing code*. Pearson Education India.
2. F. Palma, M. Nayrolles, N. Moha, Y.-G. Guéhéneuc, B. Baudry, and J.-M. Jézéquel, *"SOA Antipatterns: An Approach for Their Specification and Detection"* International Journal of Cooperative Information Systems, vol. 22, p. 1341004, 2013.
3. A. Ouni, M. Kessentini, K. Inoue, and M. O. Cinnéide, *"Search-Based Web Service Antipatterns Detection" IEEE Transactions on Services Computing*, 2015.
4. F. Palma, N. Moha, G. Tremblay, and Y.-G. Guéhéneuc, *"Specification and Detection of SOA Antipatterns In Web Services"* in European Conference on Software Architecture 2014, pp. 58-73.
5. F. Palma, J. Dubois, N. Moha, and Y.-G. Guéhéneuc, *"Detection of REST Patterns and Antipatterns: A Heuristics-Based Approach"* in International Conference on Service-Oriented Computing, 2014, pp. 230-244.
6. F. Palma, J. Gonzalez-Huerta, N. Moha, Y.-G. Guéhéneuc, and G. Tremblay, *"Are Restful APIs Well-Designed? Detection of Their Linguistic (Anti) Patterns"* in International Conference on Service-Oriented Computing, 2015, pp. 171-187.
7. V. Arnaoudova, M. Di Penta, G. Antoniol, and Y.-G. Gueheneuc, *"A New Family of Software Antipatterns: Linguistic Antipatterns"* in Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on, 2013, pp. 187-196.
8. V. Arnaoudova, M. Di Penta, G. Antoniol, and Y.-G. Gueheneuc, Repent: Analyzing the nature of identifier renamings, *IEEE Transactions on Software Engineering*, *40*(5), pp.502-532.2016
9. V. Arnaoudova, M. Di Penta, G. Antoniol, and Y.-G. Gueheneuc, "Linguistic antipatterns: What they are and how developers perceive them." *Empirical Software Engineering*21.1 (2016): 104-158.

References

10. F. Khomh, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "An Exploratory Study of The Impact of Antipatterns On Class Change-And Fault-Proneness" Empirical Software Engineering, vol. 17, pp. 243-275, 2012.
11. T. Hall, M. Zhang, D. Bowes, and Y. Sun, "Some Code Smells Have A Significant But Small Effect On Faults" ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 23, p. 33, 2014.
12. H. Liu, Q. Liu, Z. Niu, and Y. Liu, "Dynamic And Automatic Feedback-Based Threshold Adaptation For Code Smell Detection" IEEE Transactions on Software Engineering, vol. 42, pp. 544-558, 2016.
13. Jaafar, Fehmi, Angela Lozano, Yann-Gaël Guéhéneuc, and Kim Mens. "Analyzing software evolution and quality by extracting Asynchrony change patterns." Journal of Systems and Software 131 ,pp. 311-322,2017.
14. Petrillo, F., Merle, P., Moha, N., & Guéhéneuc, Y.-G. "Are REST APIs for cloud computing well-designed? An exploratory study". In International Conference on Service-Oriented Computing, pp. 157-170. Springer, Cham, 2016.
15. Wang, Hanzhang, Marouane Kessentini, and Ali Ouni. "Prediction of Web Services Evolution." In International Conference on Service-Oriented Computing, pp. 282-297. Springer, Cham, 2016.
16. Baghdadi, Y. Service-oriented software engineering: a guidance framework for service engineering methods. International Journal of Systems and Service-Oriented Engineering (IJSSOE), 2015, 5(2), 1-19.
17. M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, et al., "When and Why Your Code Starts to Smell Bad" in Proceedings of the 37th International Conference on Software Engineering-Volume 1, 2015, pp. 403-414.

18. Lehnert, S. (2011, September). A taxonomy for software change impact analysis. In Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th annual ERCIM Workshop on Software Evolution(pp. 41-50). ACM.
19. Mäntylä, Mika V., and Casper Lassenius. "Subjective evaluation of software evolvability using code smells: An empirical study." Empirical Software Engineering 11, no. 3: 395-431,2006.
20. Kaur, H., & Kaur, P. J.. A Study on Detection of Antipatterns in Object-Oriented Systems. International Journal of Computer Applications,93(5),2014.
21. F. Petrillo, P. Merle, N. Moha, and Y.-G. Gueh´eneuc,´ "Towards a rest cloud computing lexicon," in 7th International Conference on Cloud Computing and Services Science, CLOSER , 2017.
22. M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, and D. Poshyvanyk, "When and why your code starts to smell bad (and whether the smells go away)," IEEE Transactions on Software Engineering, 2017.
23. F. Haupt, F. Leymann, A. Scherer, and K. Vukojevic-Haupt, "A frame-work for the structural analysis of rest apis," in Software Architecture (ICSA),IEEE International Conference on. IEEE, 2017, pp. 55– 58.
24. M. Athanasopoulos and K. Kontogiannis, "Extracting rest resource models from procedure-oriented service interfaces," Journal of Systems and Software, vol. 100, pp. 149–166, 2015.
25. G. Bavota, R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, "Methodbook: Recommending move method refactorings via relational topic models," IEEE Transactions on Software Engineering, vol. 40, no. 7, pp. 671–694, 2014.
26. B. F. dos Santos Neto, M. Ribeiro, V. T. da Silva, C. Braga, C. J. P. de Lucena, and E. de Barros Costa, "Autorefactoring: A platform to build refactoring agents," Expert Systems with Applications, vol. 42, no. 3, pp. 1652–1664, 2015.

27. T. Espinha, A. Zaidman, and H.-G. Gross, “Web api growing pains: Loosely coupled yet strongly tied,” Journal of Systems and Software, vol. 100, pp. 27–43, 2015.

28. G. Salvatierra, C. Mateos, M. Crasso, and A. Zunino, “Towards a computer assisted approach for migrating legacy systems to soa,” Computational Science and Its Applications–ICCSA 2012, pp. 484–497, 2012.

29. F. Jaafar, Y.-G. Gueh´eneuc,´ S. Hamel, F. Khomh, and M. Zulkernine, “Evaluating the impact of design pattern and antipattern dependencies on changes and faults,” Empirical Software Engineering, vol. 21, no. 3,896–931, 2016.

30. B. Costa, P. F. Pires, F. C. Delicato, and P. Merson, “Evaluating rest architectures—approach, tooling and guidelines,” Journal of Systems and Software, vol. 112, pp. 156–180, 2016.

31. Kuttal, S. K., Sarma, A., & Rothermel, G. “On the benefits of providing versioning support for end users: an empirical study”. ACM Transactions on Computer-Human Interaction (TOCHI), 21(2), 9,2014.

32. Jézéquel, Jean-Marc, Michel Train, and Christine Mingins. *Design Patterns with Contracts*. Addison-Wesley Longman Publishing Co., Inc., 1999.

Questions ?