PhD Thesis

Unifying Service Oriented **Technologies** for the **Specification** and **Detection** of their **Antipatterns**

Francis PALMA

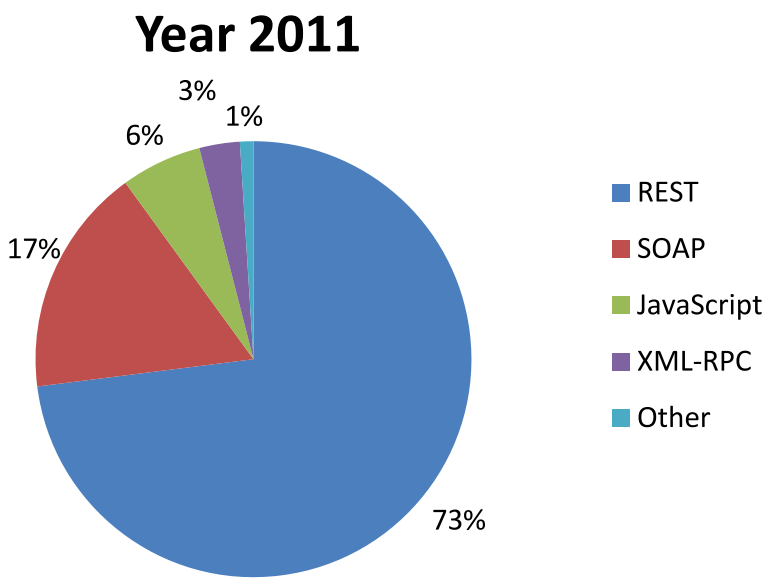
Supervised by:
Dr. Naouel Moha, Université du Québec à Montréal, Canada
Dr. Yann-Gaël Guéhéneuc, École Polytechnique de Montréal, Canada

August 19, 2015

Context: Adoption of Web Technologies and Styles

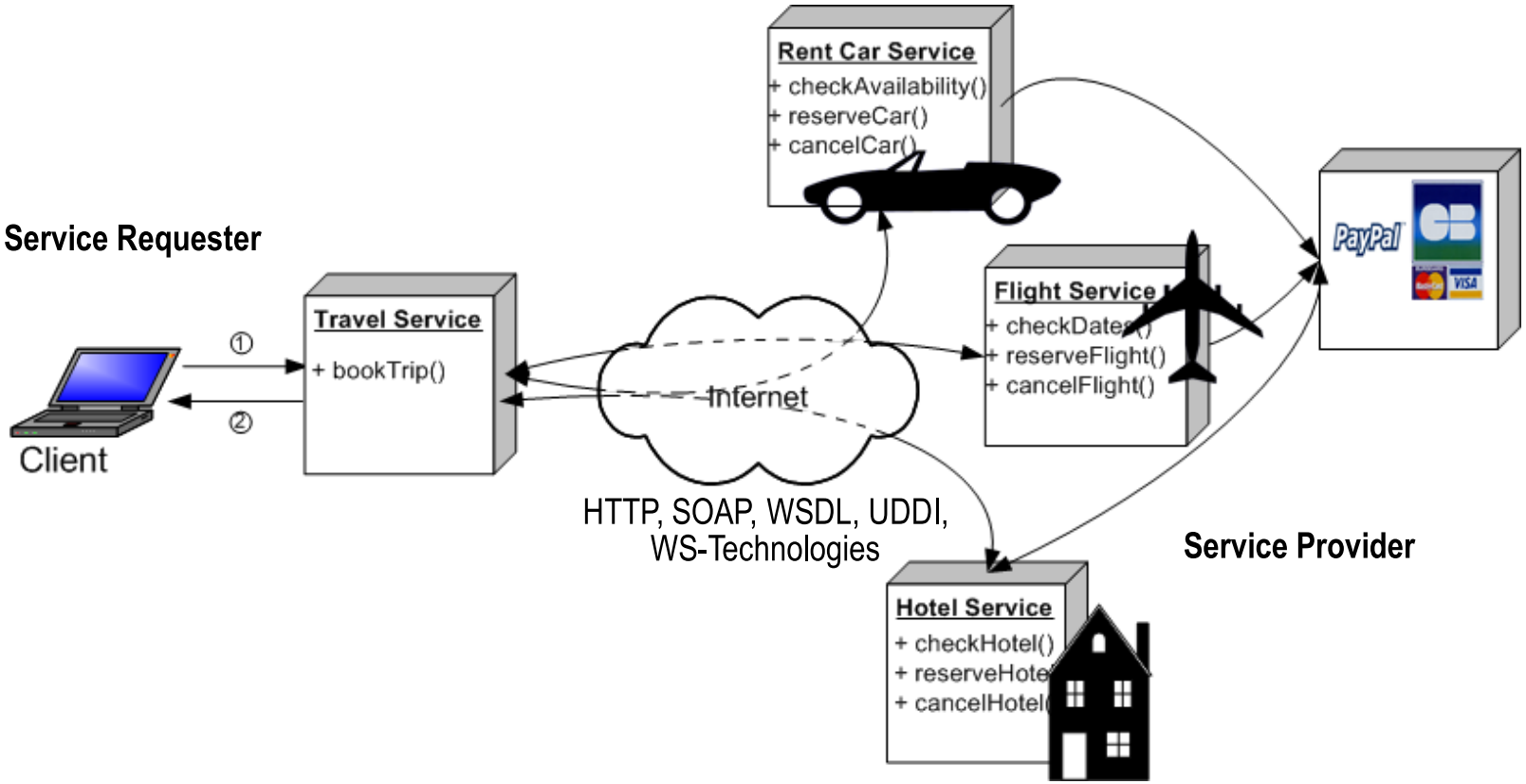
80% of software development projects will be based on SOA by 2008.

- Gartner (2006)



*Source: www.programmableweb.com

Context: Service-based Systems (SBSs)



Thesis

Poor but recurring design practices in **SBSs**  design quality and ***cause more effort*** 

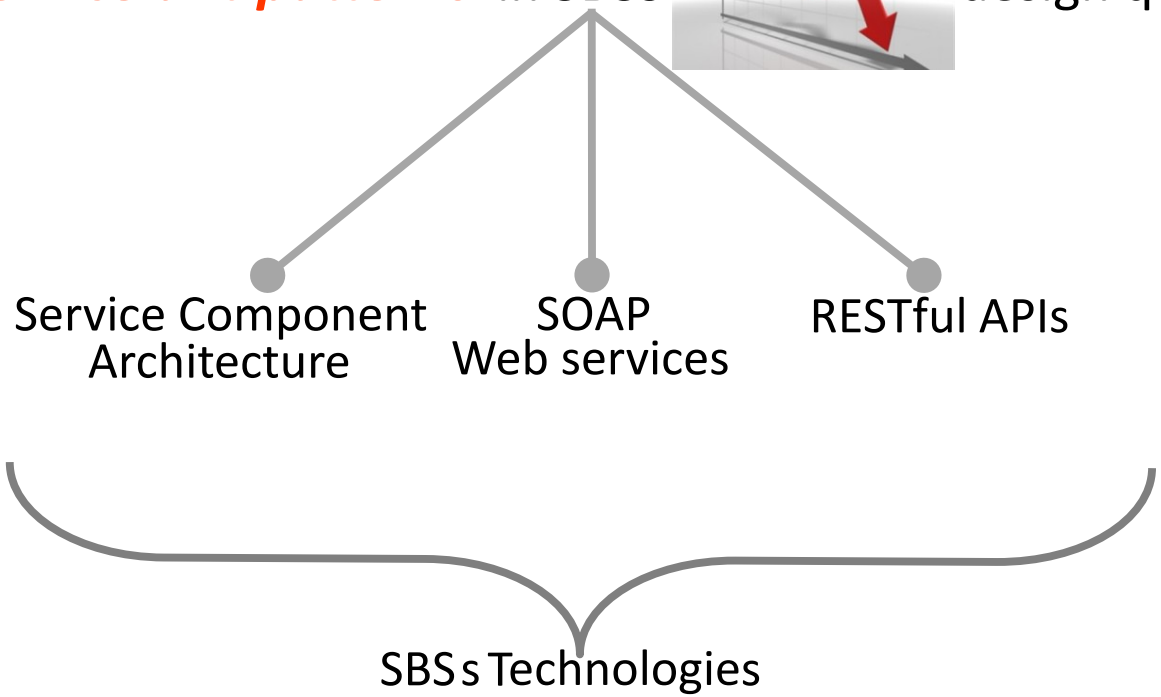
Thesis

Service antipatterns in **SBSs**  design quality and ***hinder*** 

Thesis

Service antipatterns in **SBSs**

design quality and ***hinder***



Thesis

Service antipatterns in **SBSs**

design quality and ***hinder***

Service Component Architecture

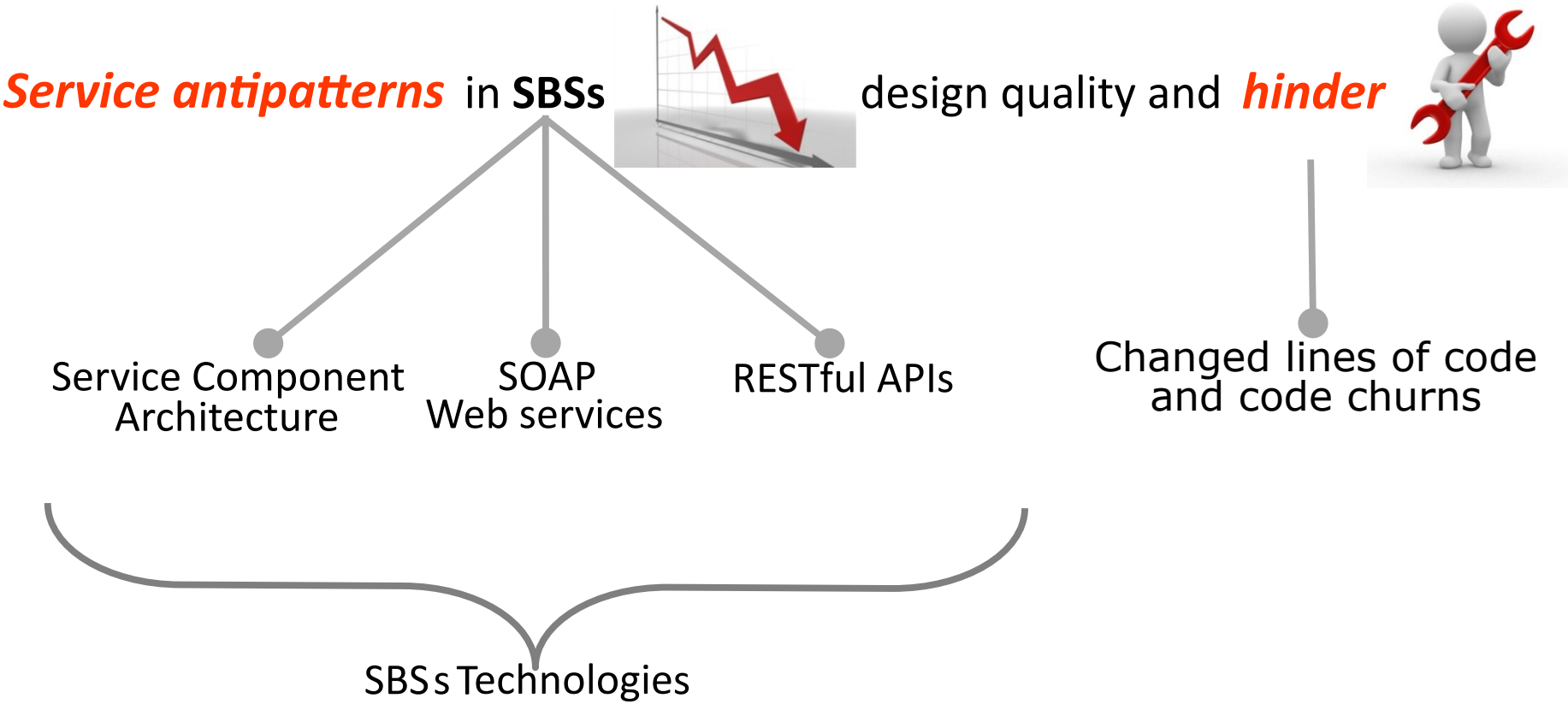
SOAP Web services

RESTful APIs

SBSs Technologies

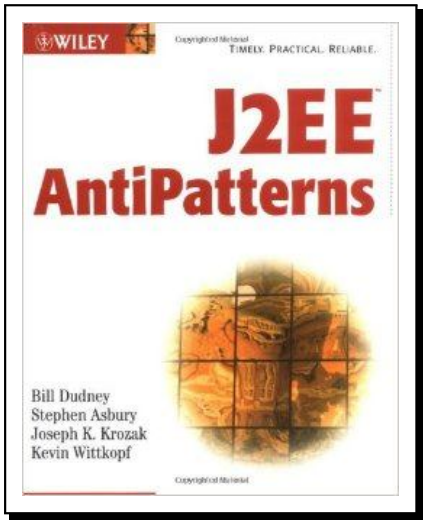
Changed lines of code and code churns

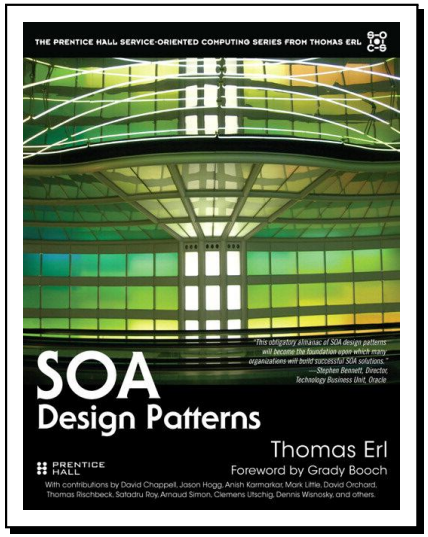
Thesis

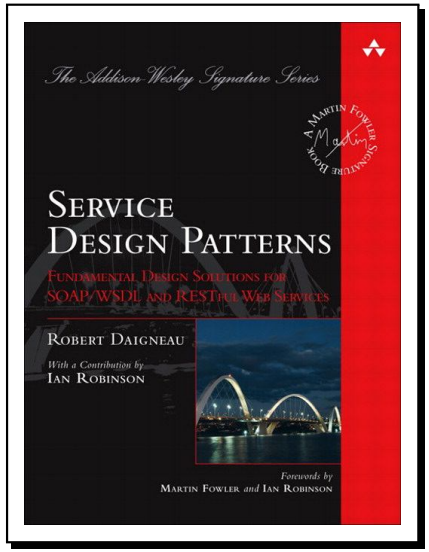


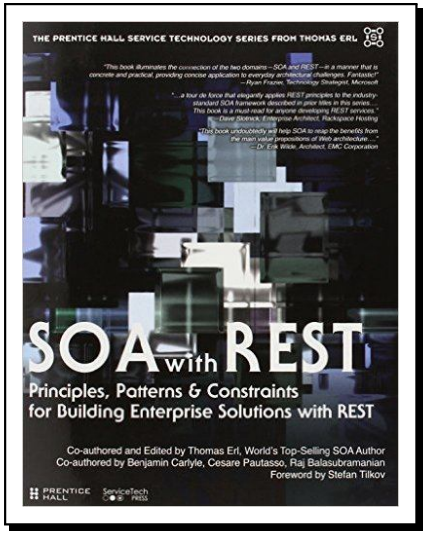
Proposing a unified approach to specify and detect service antipatterns by assessing the quality of design and quality of service of SBSs.

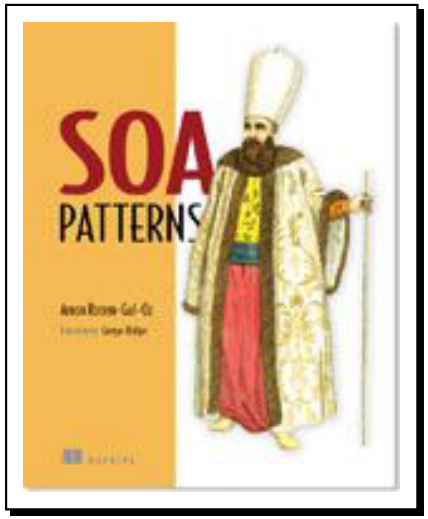
Service Antipatterns Catalog: Books



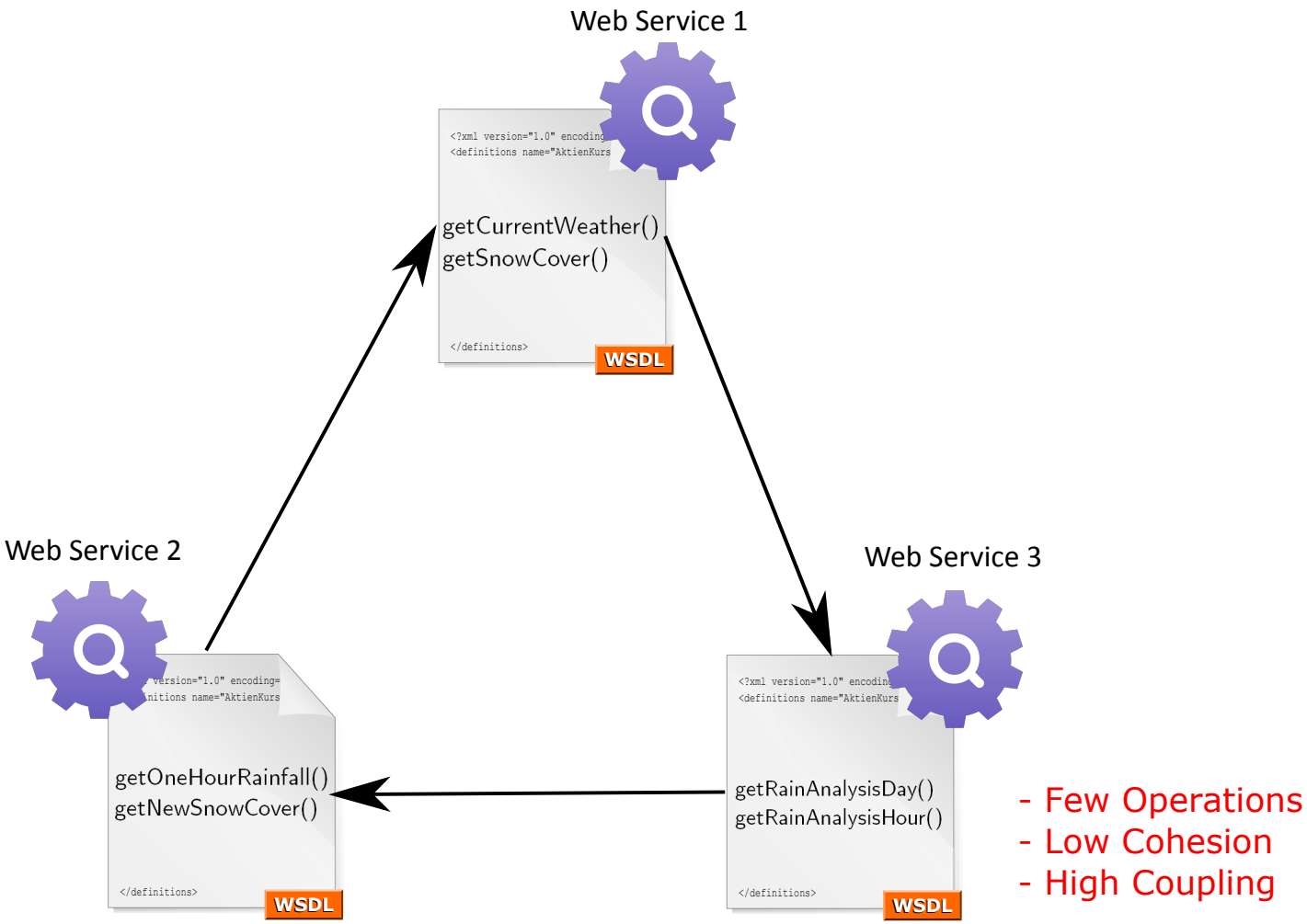




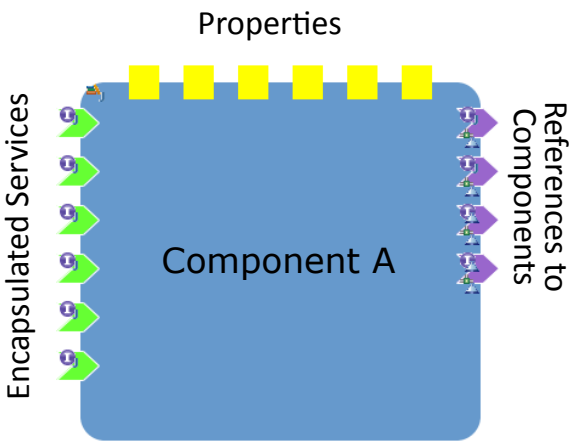




Tiny Service Antipattern in SOAP Web Services



God Component Antipattern in SCA




```
package org.ow2.frascati.component.factory.api;

import org.objectweb.fractal.api.Component;
import org.objectweb.fractal.api.type.ComponentType;

import org.osoa.sca.annotations.Service;

@Service
public interface ComponentFactory
{

    void generateMembrane(ComponentFactoryContext context, ComponentType componentType, String
  membraneDesc,
        String contentClass) throws FactoryException;


   void generateScaPrimitiveMembrane(ComponentFactoryContext context, ComponentType
  componentType, String classname)
       throws FactoryException;


   void generateScaCompositeMembrane(ComponentFactoryContext context, ComponentType
  componentType)
       throws FactoryException;


   Component createComponent(ComponentFactoryContext context, ComponentType componentType,
  String membraneDesc,
       Object contentClass) throws FactoryException;

   Component createScaPrimitiveComponent(ComponentFactoryContext context, ComponentType
  componentType, String classname)
       throws FactoryException;

   Component createScaCompositeComponent(ComponentFactoryContext context, ComponentType
  componentType)
       throws FactoryException;

   Component createScaContainer(ComponentFactoryContext context)
       throws FactoryException;

}
```

Forgetting Hypermedia Antipattern in RESTful APIs

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",
        "root": "dropbox",
        "mime_type": "text/plain",
        "size": "4 bytes "
    }
],
"size": "0 bytes "
}
```

No links to follow...

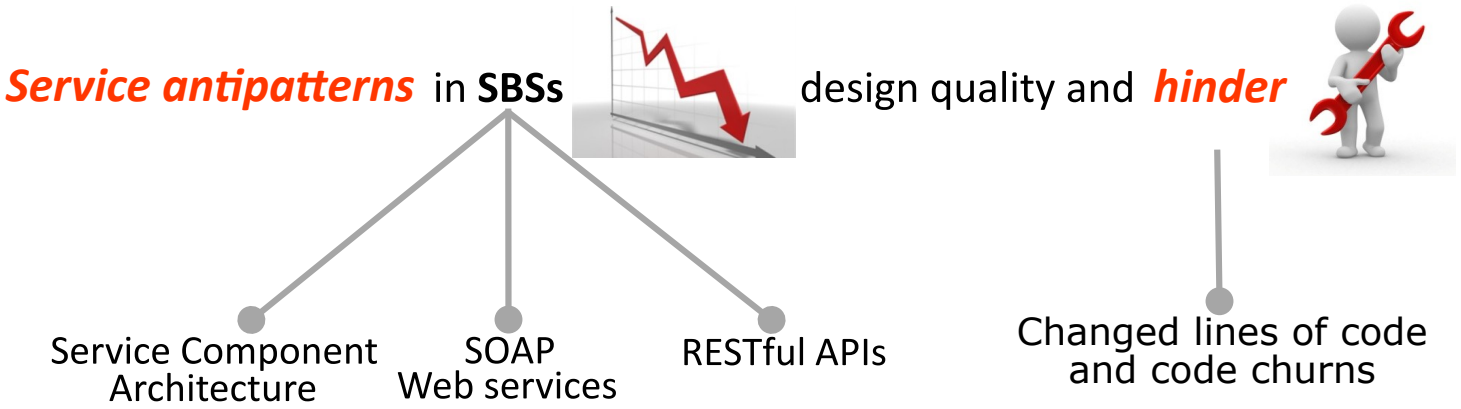
No links to follow...

DropBox Server Response 2:

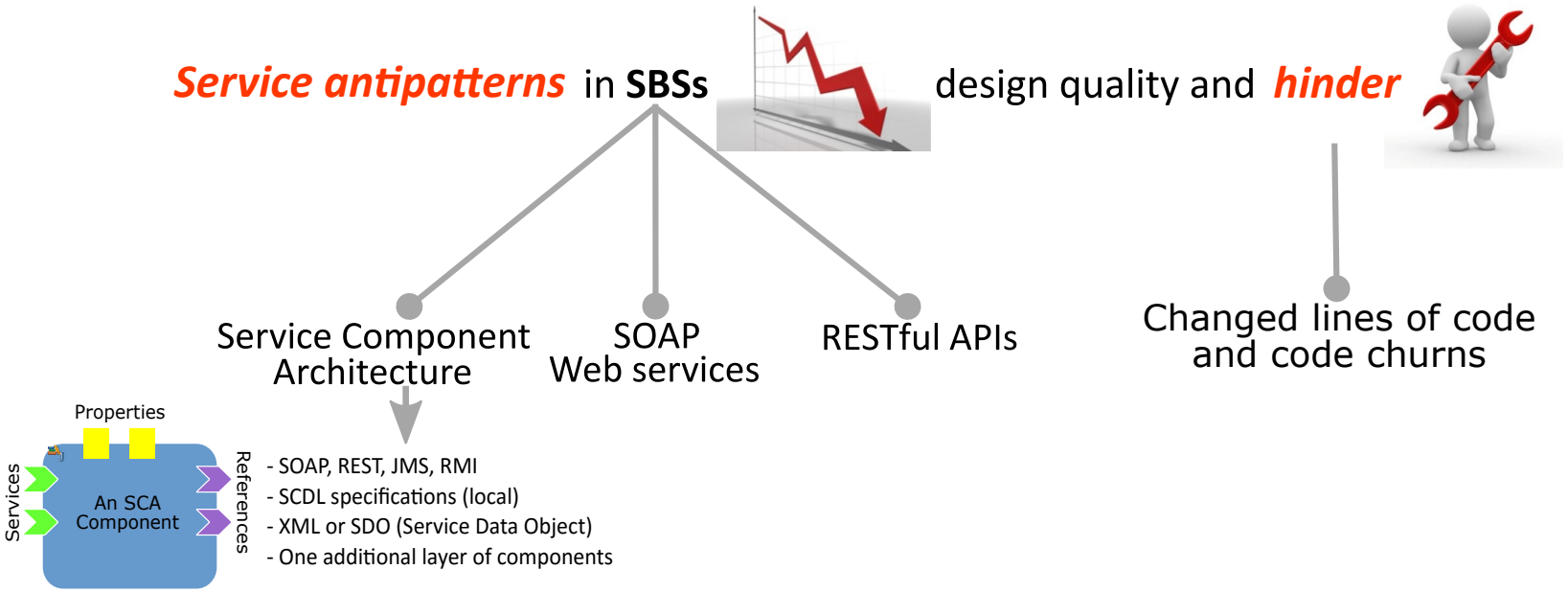
```
Header: {
    x-frame-options=[SAMEORIGIN],
    x-dropbox-request-id=[cd12e1e844327464485842b11b530071],
    Connection=[keep-alive],
    Server=[nginx],
    pragma=[no-cache],
    cache-control=[no-cache],
    x-server-response-time=[110],
    x-dropbox-http-protocol=[None],
    set-cookie=[gvc=MzIwNTkxODQzNjQy.....;
    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...

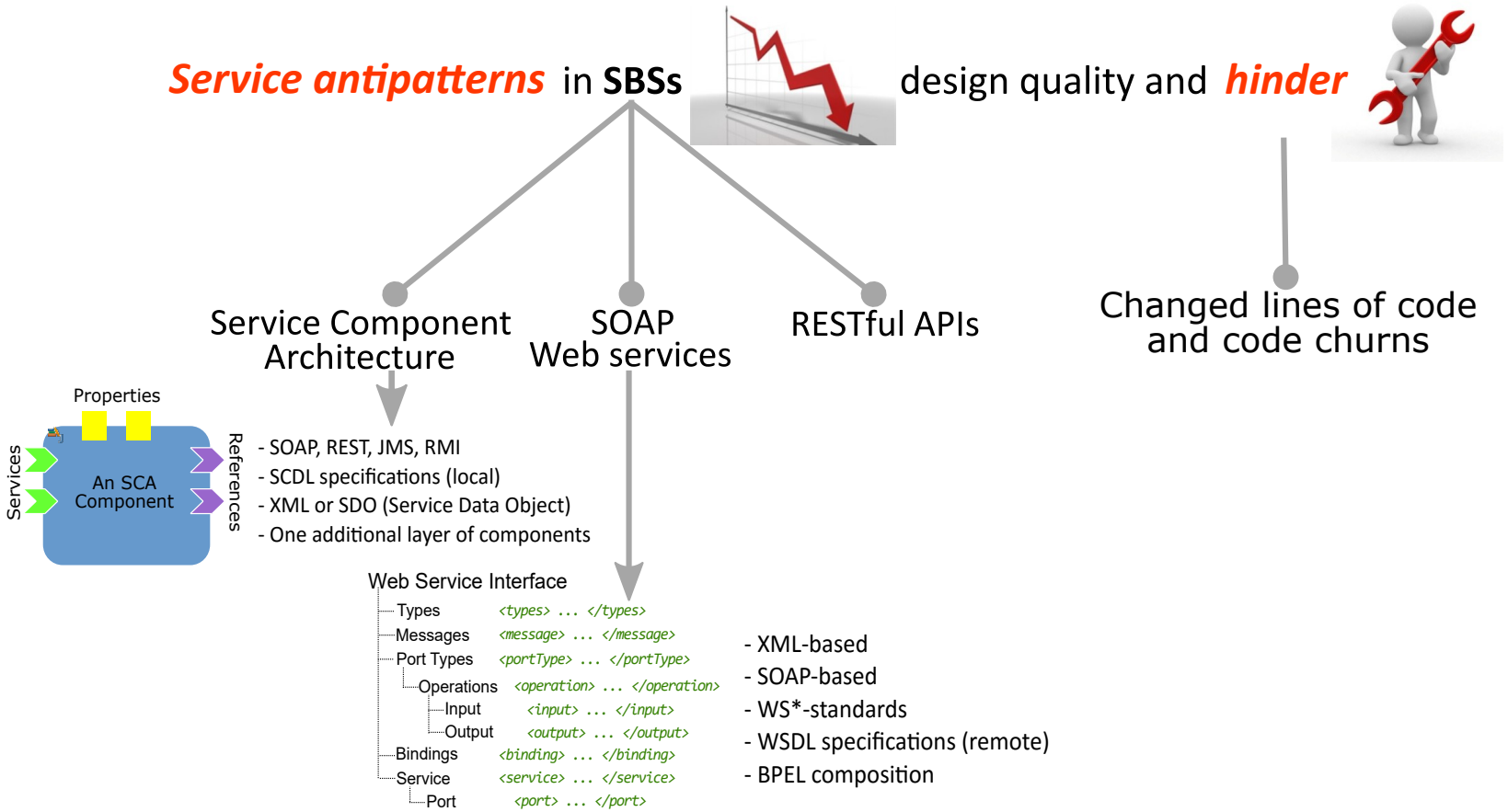
Challenges and Problems



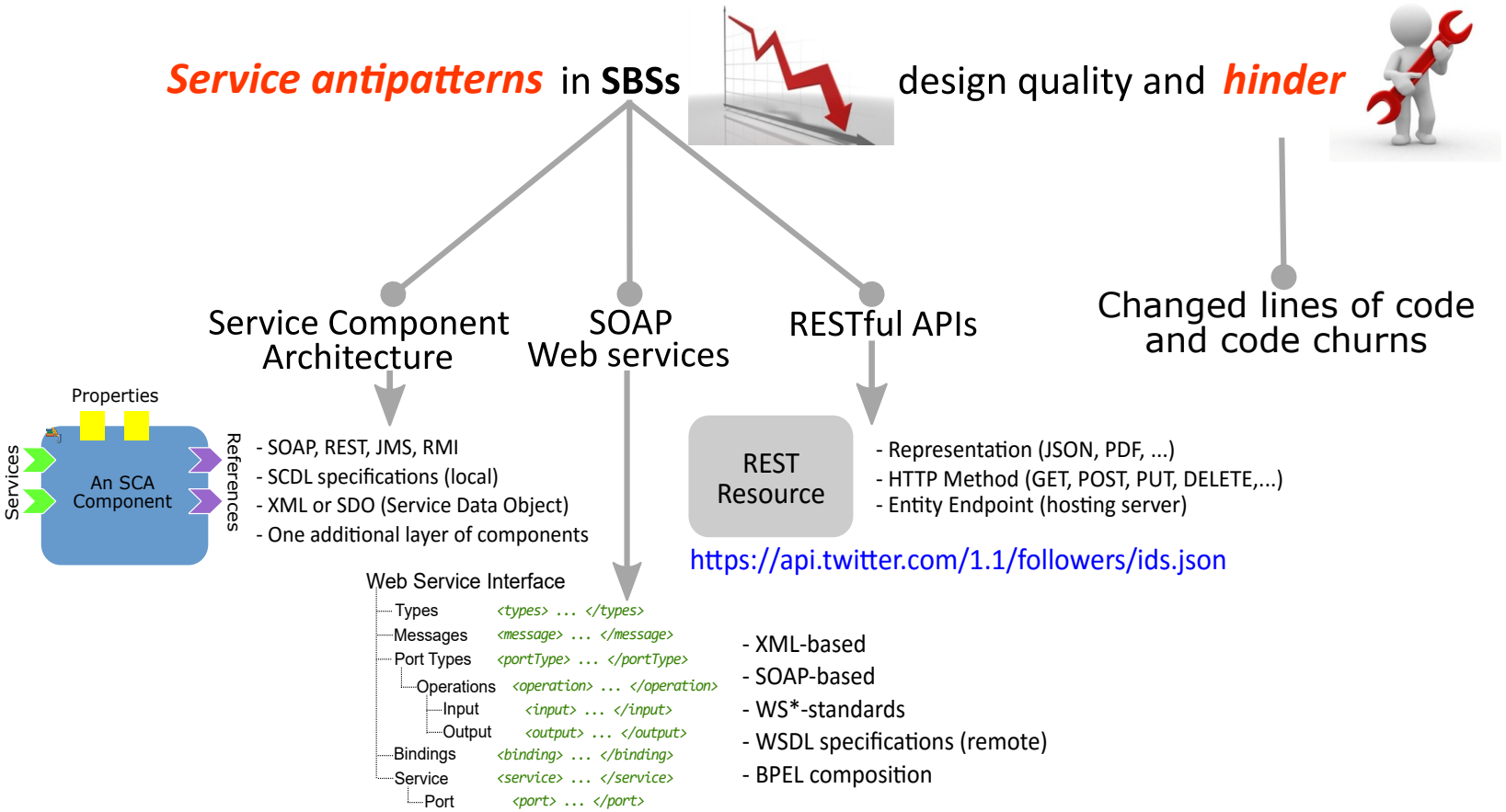
Challenges and Problems



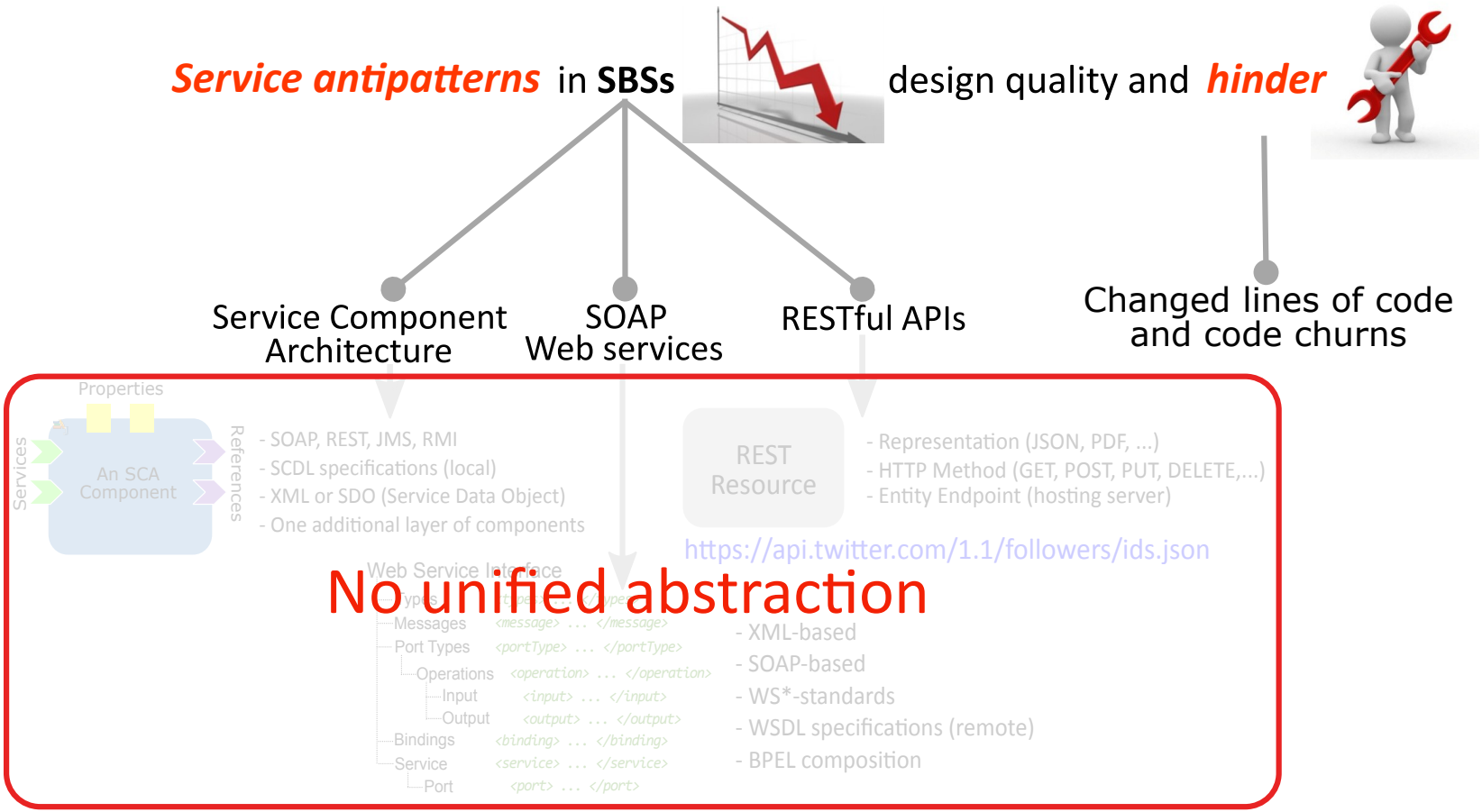
Challenges and Problems



Challenges and Problems



Challenges and Problems



Challenge 1

Challenges and Problems

Challenge 2

- Only textual descriptions
- No specifications

Service antipatterns in **SBSs**

design quality and ***hinder***

Service Component Architecture

SOAP Web services

RESTful APIs

Changed lines of code and code churns

Properties

Services

An SCA Component

References

- SOAP, REST, JMS, RMI
- SCDL specifications (local)
- XML or SDO (Service Data Object)
- One additional layer of components

REST Resource

- Representation (JSON, PDF, ...)
- HTTP Method (GET, POST, PUT, DELETE,...)
- Entity Endpoint (hosting server)

https://api.twitter.com/1.1/followers/ids.json

Web Service Interface

- Types `<types> ... </types>`
- Messages `<message> ... </message>`
- Port Types `<portType> ... </portType>`
  - Operations `<operation> ... </operation>`
    - Input `<input> ... </input>`
    - Output `<output> ... </output>`
- Bindings `<binding> ... </binding>`
- Service `<service> ... </service>`
  - Port `<port> ... </port>`

- XML-based
- SOAP-based
- WS*-standards
- WSDL specifications (remote)
- BPEL composition

Challenge 1

Challenges and Problems

Challenge 2

- Only textual descriptions
- No specifications

Challenge 3

- No dedicated unified approach
- No unified framework

Service antipatterns in **SBSs**

design quality and ***hinder***

Changed lines of code and code churns

Service Component Architecture

SOAP Web services

RESTful APIs

Properties

Services

An SCA Component

References

- SOAP, REST, JMS, RMI
- SCDL specifications (local)
- XML or SDO (Service Data Object)
- One additional layer of components

REST Resource

- Representation (JSON, PDF, ...)
- HTTP Method (GET, POST, PUT, DELETE,...)
- Entity Endpoint (hosting server)

https://api.twitter.com/1.1/followers/ids.json

Web Service Interface

Types	<types> ... </types>
Messages	<message> ... </message>
Port Types	<portType> ... </portType>
Operations	<operation> ... </operation>
Input	<input> ... </input>
Output	<output> ... </output>
Bindings	<binding> ... </binding>
Service	<service> ... </service>
Port	<port> ... </port>

- XML-based
- SOAP-based
- WS*-standards
- WSDL specifications (remote)
- BPEL composition

Challenge 1

Challenges and Problems

Challenge 2

- Only textual descriptions
- No specifications

Challenge 3

- No dedicated unified approach
- No unified framework

Challenge 4

- No empirical evidence

Service antipatterns in **SBSs**

design quality and ***hinder***

Service Component Architecture

SOAP Web services

RESTful APIs

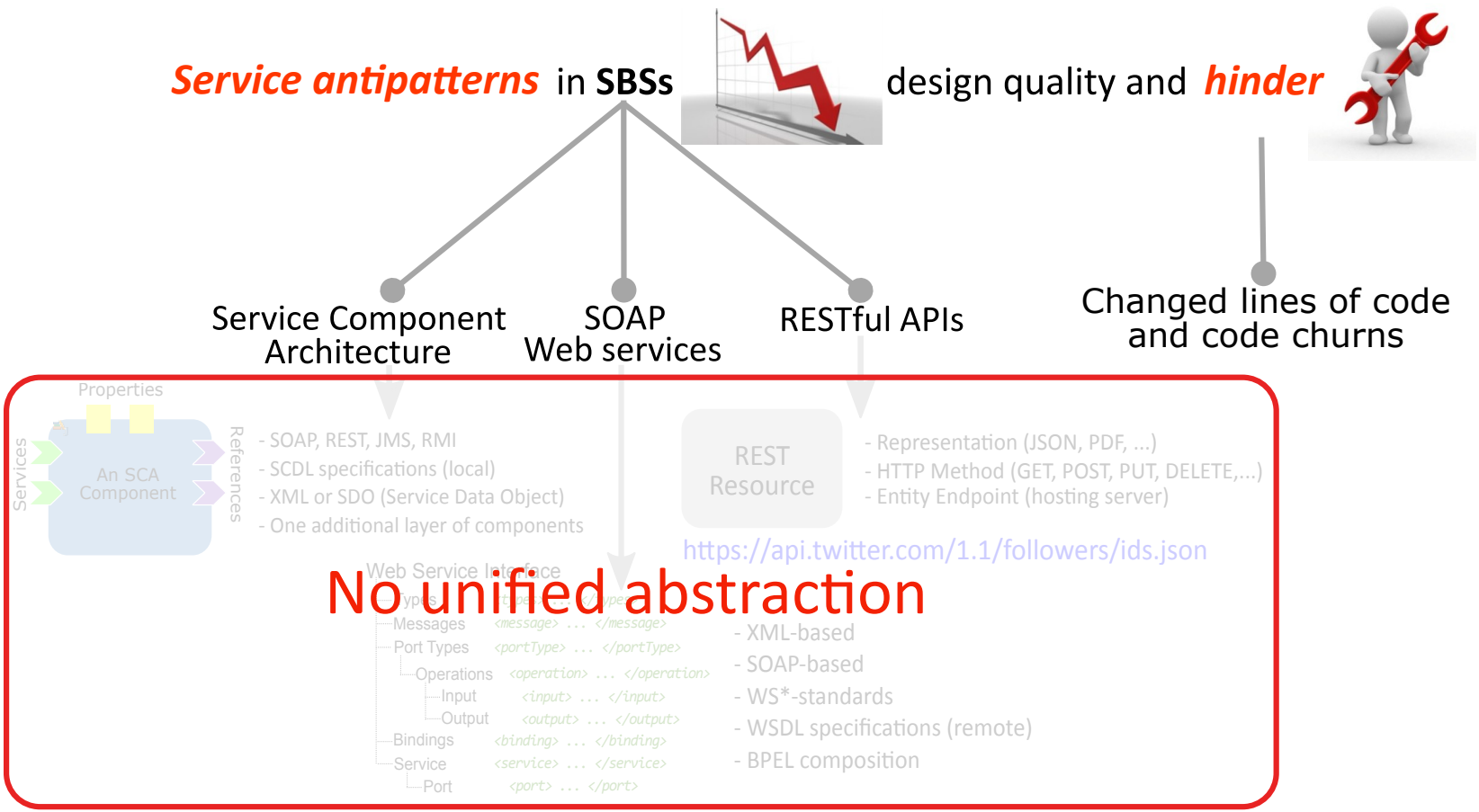
Changed lines of code and code churns

Challenge 1

Antipatterns Detection: Relevant Works in the Literature

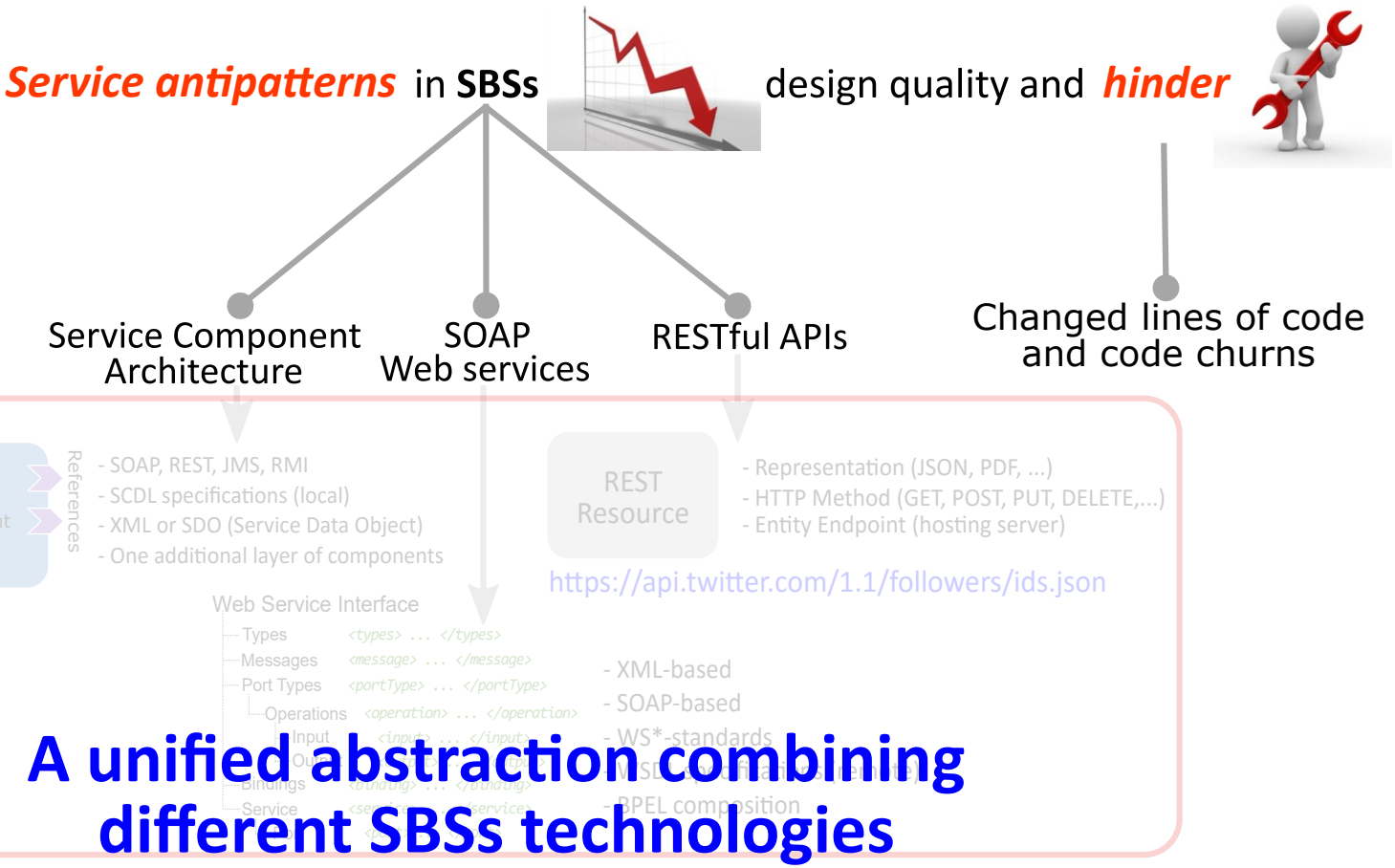
Component-based Systems (CBS)	Object-Oriented Systems (OO)	Service Component Architecture (SCA)	SOAP Web services	RESTful APIs	Unified
Trubiani et al. (2014)	Peiris et Hill (2014)	Nayrolles et al. (2013)	Ouni et al. (2015)		
Wert et al. (2014)	Cortellessa et al. (2014)		Anchuri et al. (2014)		
Zhang et al. (2012)	Marco et Trubiani (2014)		Torkamani et Bagheri (2014)		
Cortellessa et al. (2010)	Ouni et al. (2013)		Coscia et al. (2013)		
Garcia et al. (2009)	Maiga et al. (2012)		Tripathi et al. (2014)		
Parsons et Murphy (2008)	Cortellessa et al. (2012)		Rodriguez et al. (2013)		
	Khomh et al. (2011)		Mateos et al. (2011)		
	Stoianov et Sora (2010)		Rodriguez et al. (2010b)		
	Moha et al. (2010)		Král et Žemlička (2009)		
	Salehie et al. (2006)		Král et Žemlička (2008)		
	Smith et Williams (2002)		Král et Žemlička (2007)		
	Smith et Williams (2000)		Zheng et Krause (2006)		

Thesis Contributions



Challenge 1

Thesis Contributions



Thesis Contributions

Challenge 2

- Only textual descriptions
- No specifications

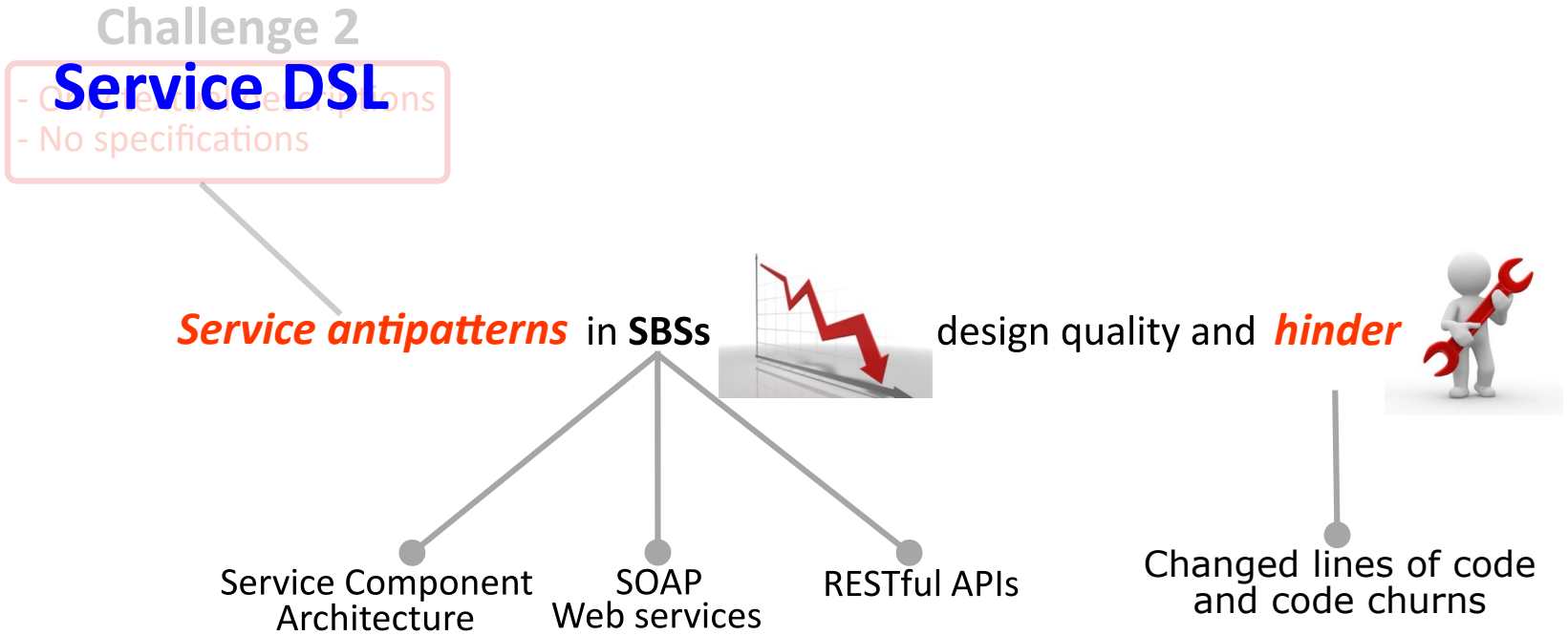
Service antipatterns in **SBSs**

design quality and ***hinder***

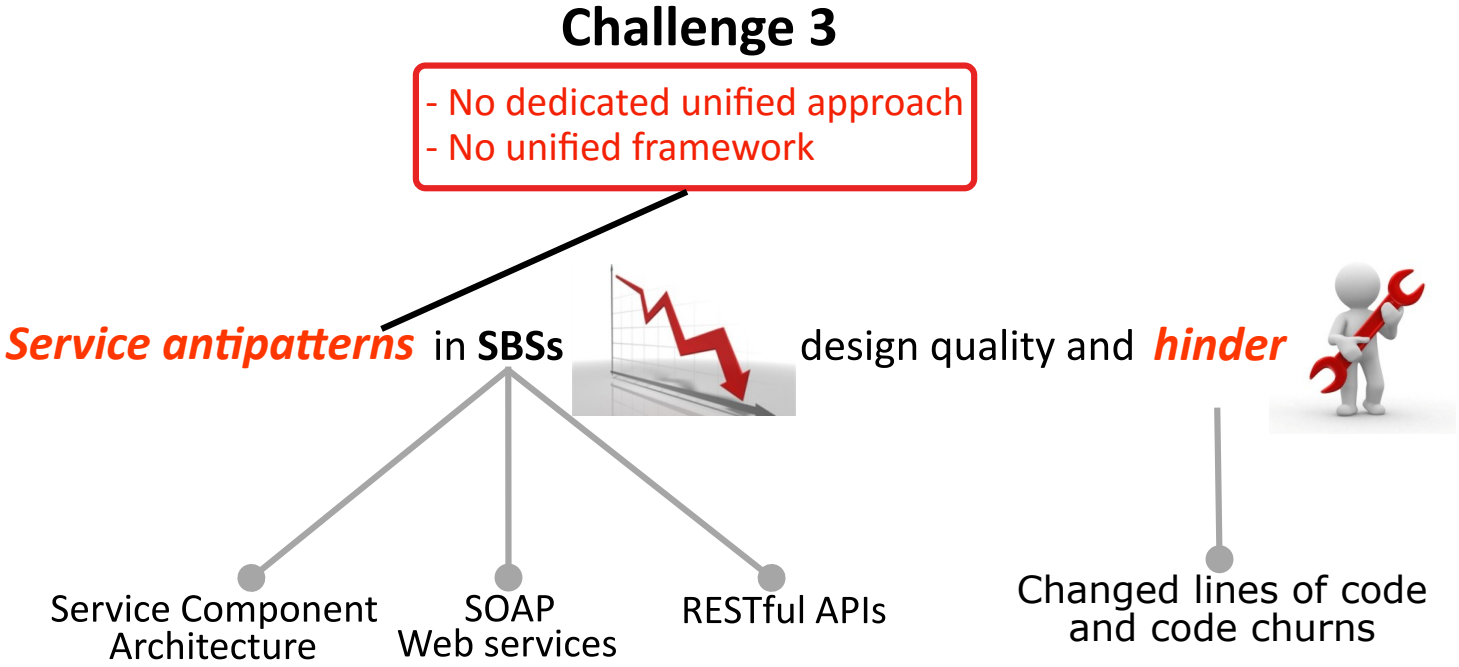
- Service Component Architecture
- SOAP Web services
- RESTful APIs

- Changed lines of code and code churns

Thesis Contributions



Thesis Contributions



Thesis Contributions

Challenge 3

Unified SODA Approach
SOFA Framework
Validation of SODA

Service antipatterns in **SBSs**

design quality and ***hinder***

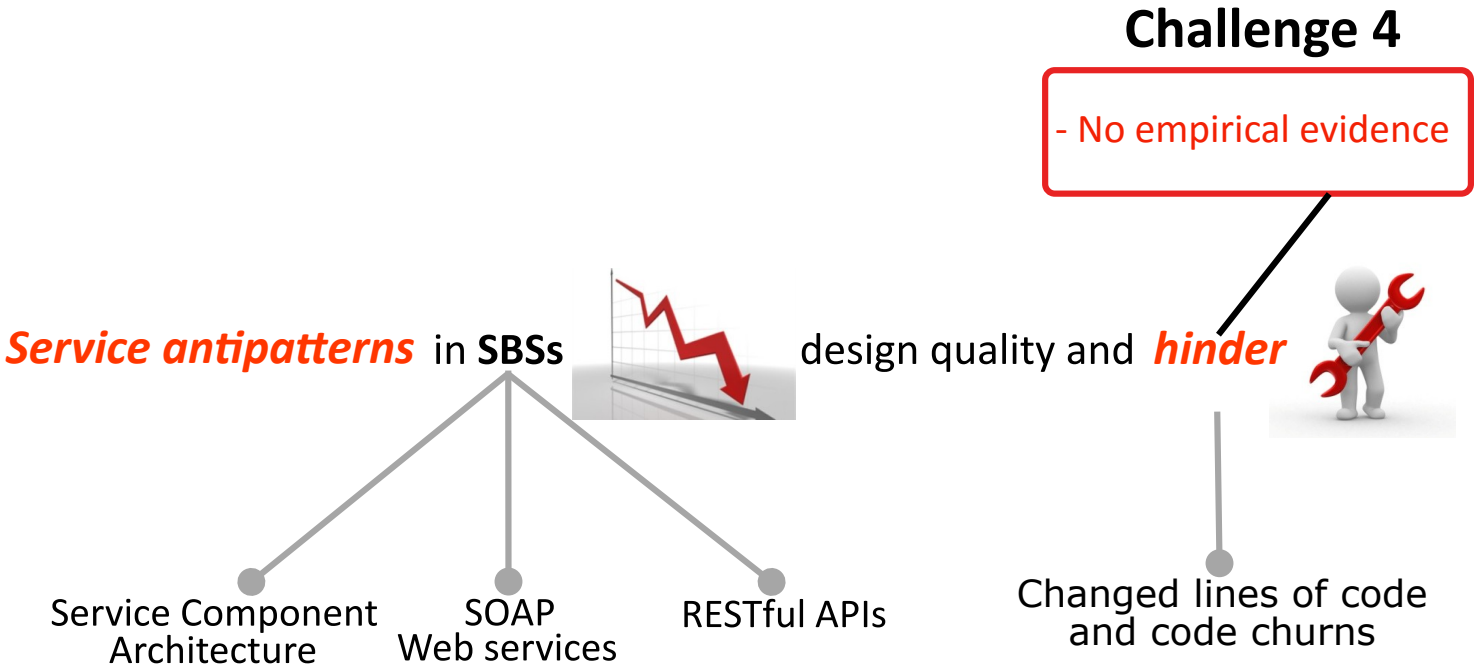
Service Component Architecture

SOAP Web services

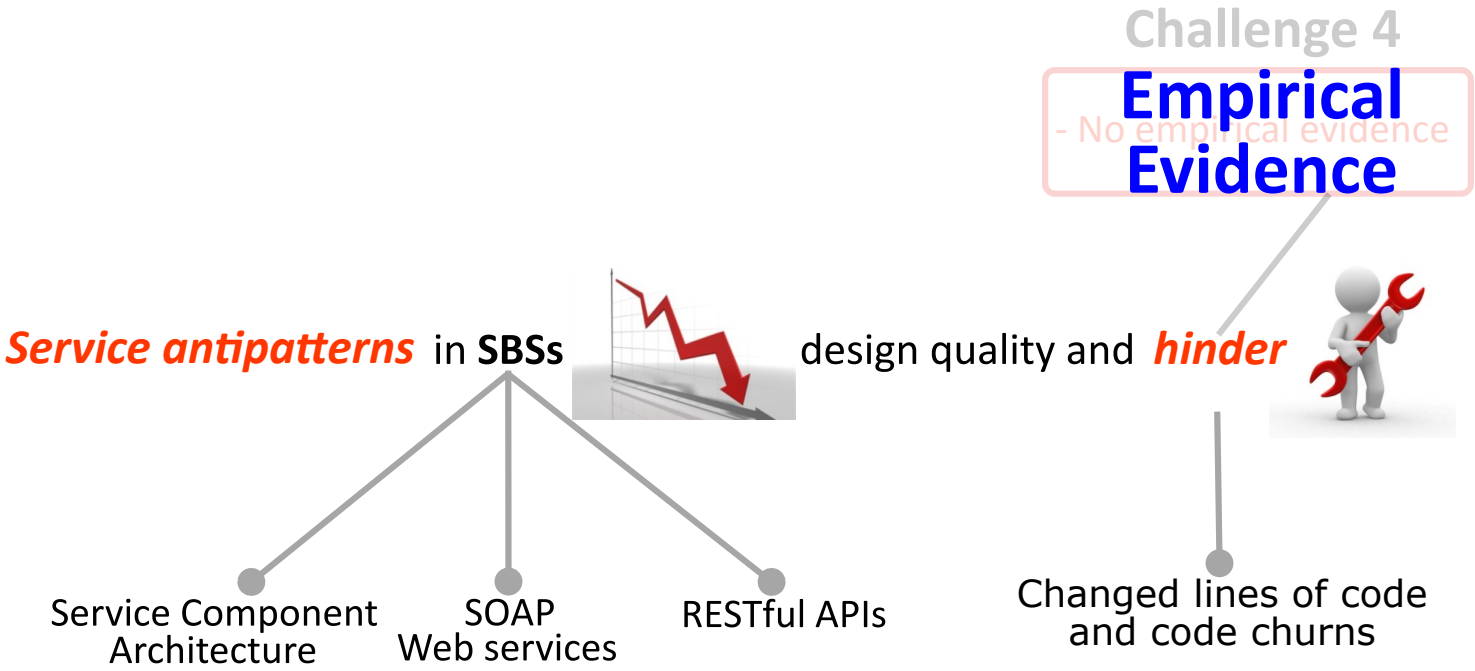
RESTful APIs

Changed lines of code and code churns

Thesis Contributions



Thesis Contributions



Thesis Contributions

Challenge 2

Service DSL

Challenge 3

Unified SODA Approach
SOFA Framework
Validation of SODA

Challenge 4

Empirical Evidence

Service antipatterns in **SBSs**  design quality and ***hinder*** 

Service Component Architecture

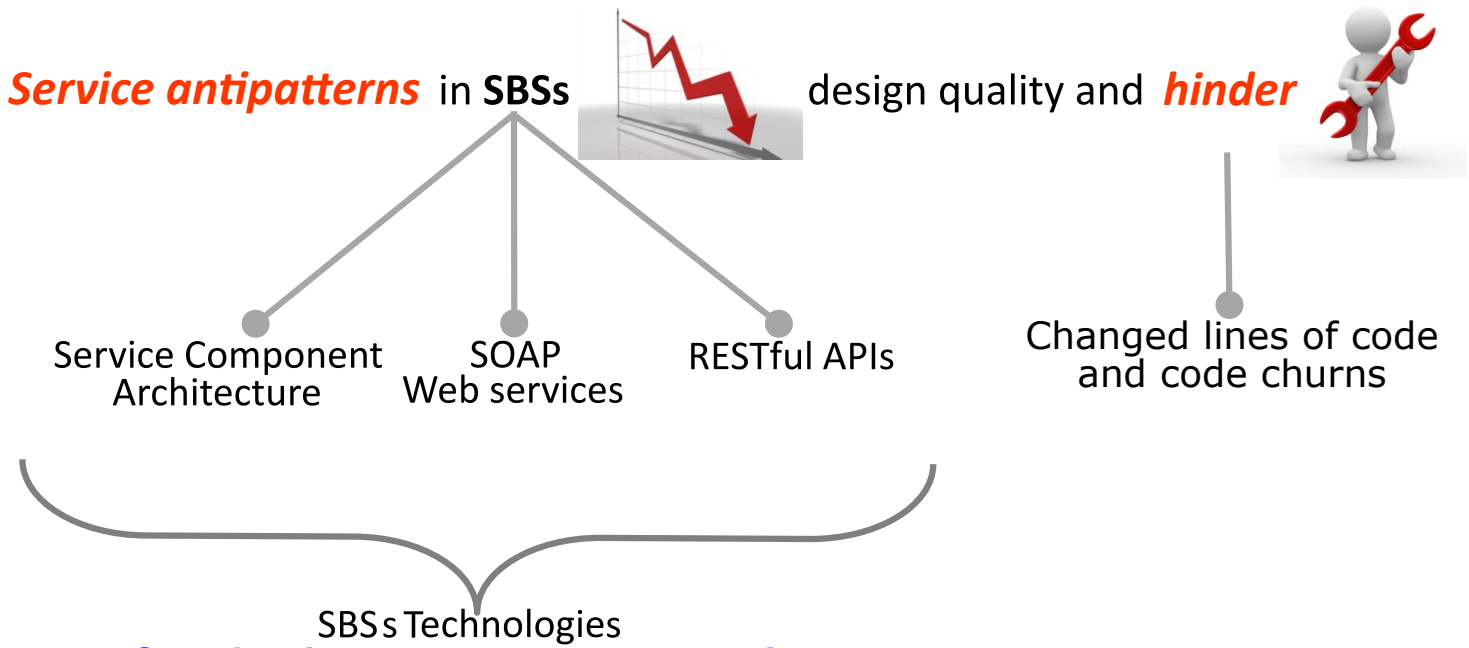
SOAP Web services

RESTful APIs

Changed lines of code and code churns

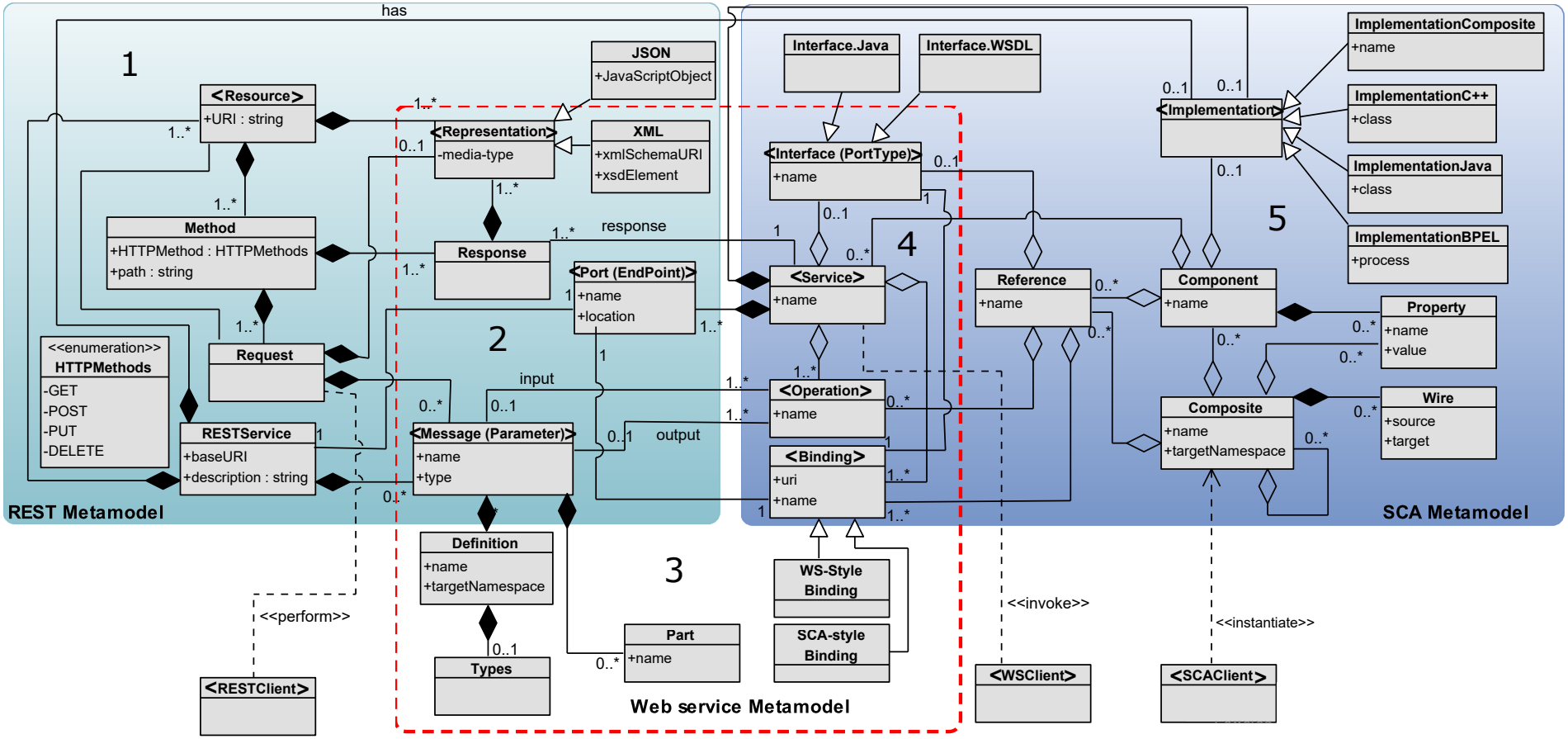
A unified abstraction combining different SBSs technologies

Contribution 1

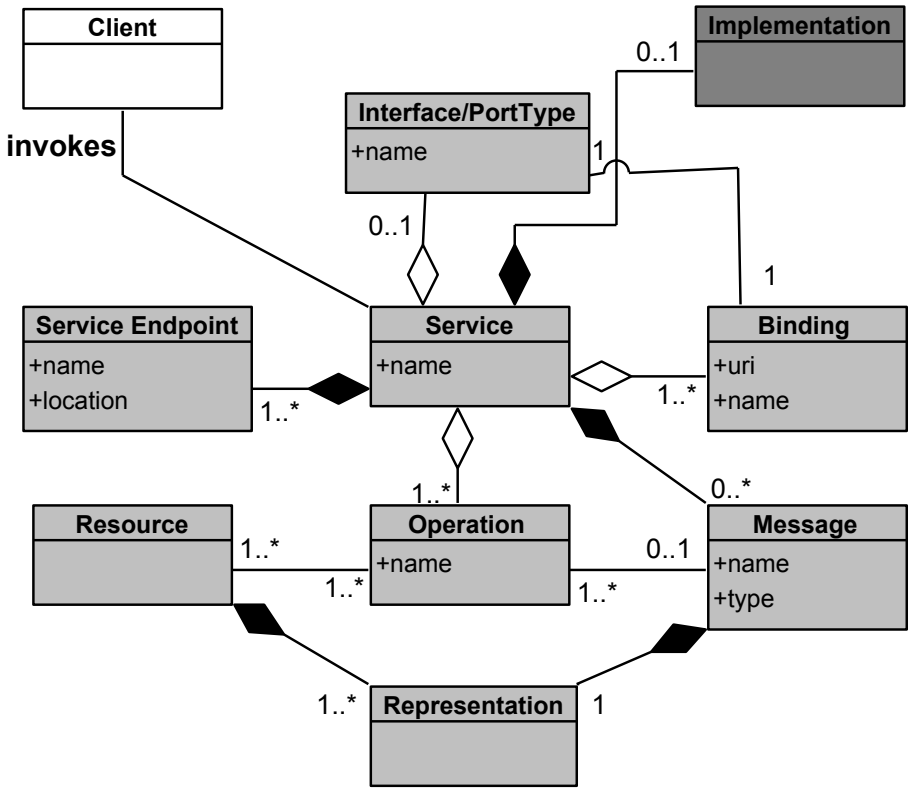


A unified abstraction combining different SBSs technologies

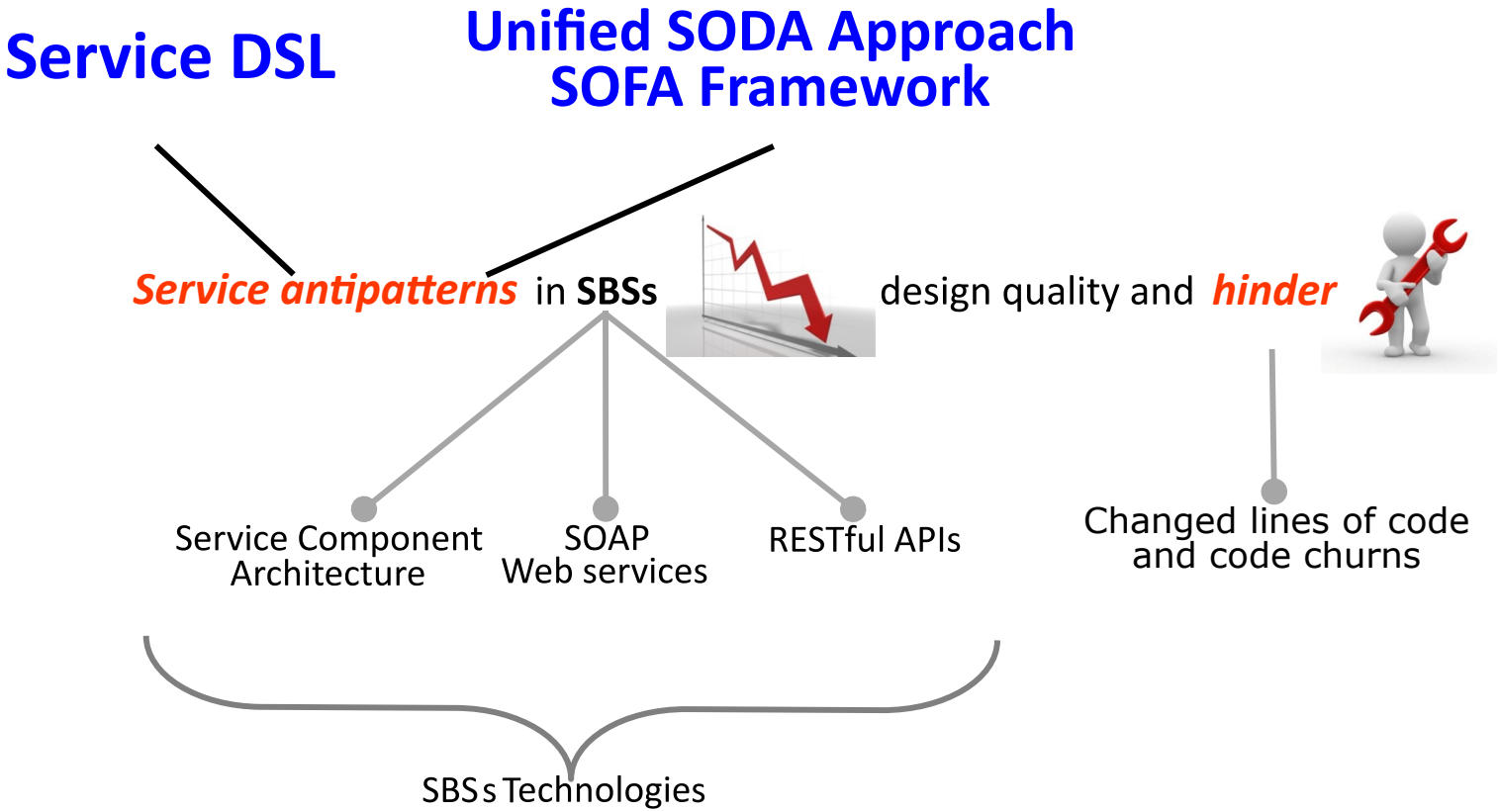
Unified Abstraction



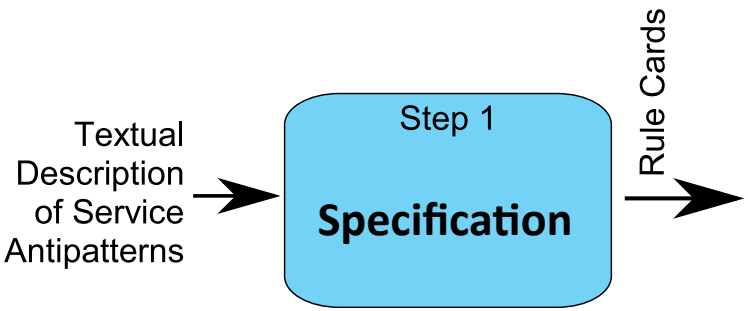
Meta-abstraction



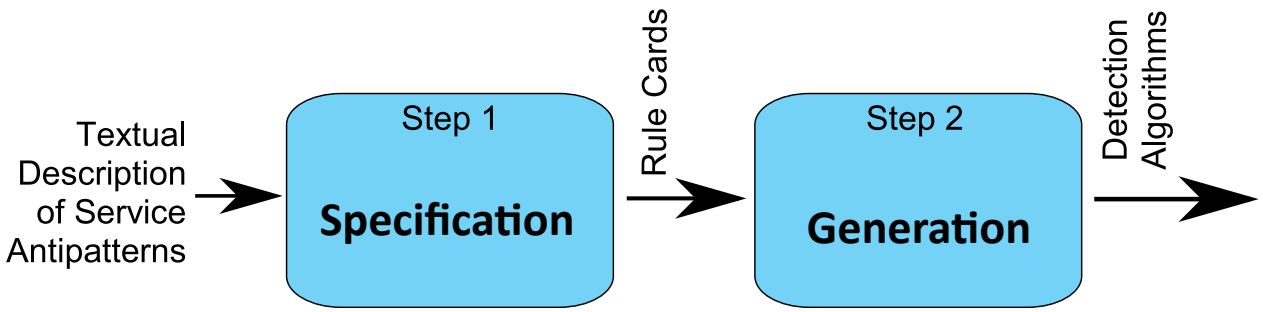
Contributions 2 and 3



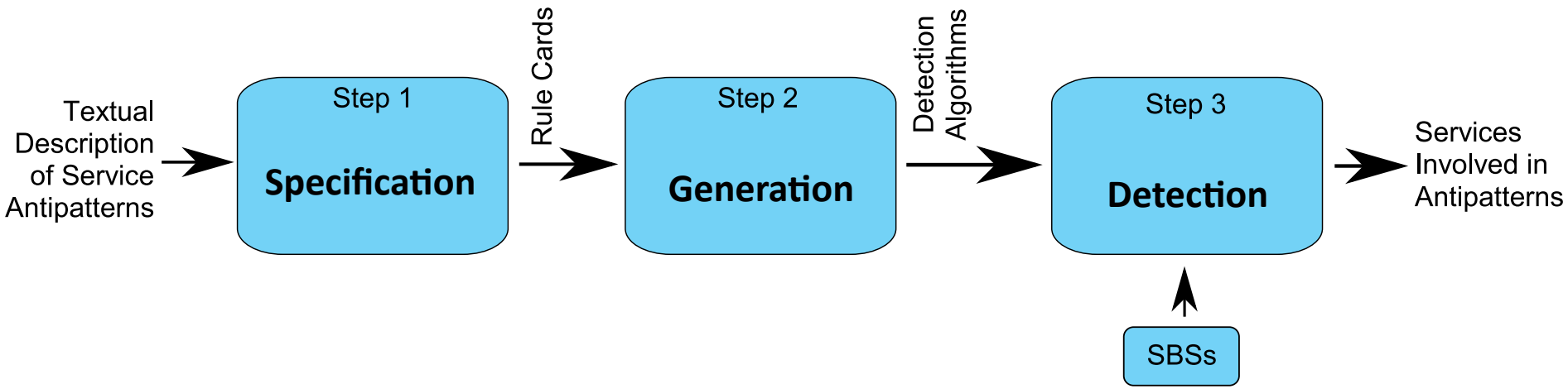
SODA (Service Oriented Detection for Antipatterns)



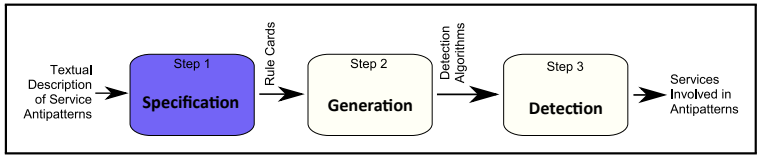
SODA (Service Oriented Detection for Antipatterns)



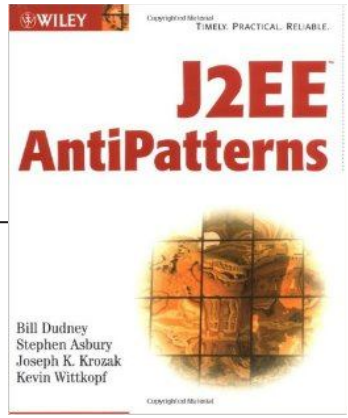
SODA (Service Oriented Detection for Antipatterns)



Domain Analysis for Antipatterns Specifications



Tiny Service



Also Known As: Refactor Mercilessly

Most Frequent Scale: System

Refactorings: Interface Consolidation

Refactored Solution Type: Software

Root Causes: Ignorance

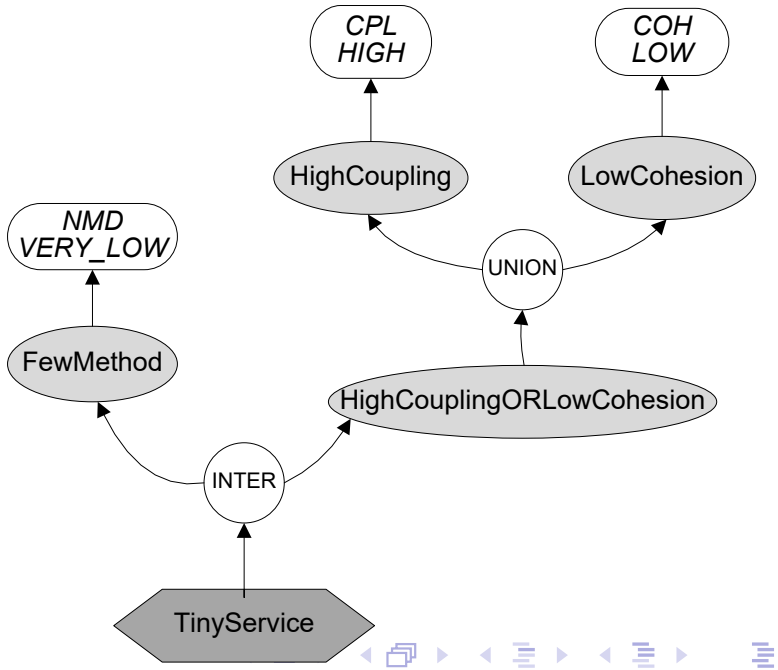
Unbalanced Forces: Complexity and resources

Anecdotal Evidence: "I read somewhere that each use case should map to a separate service."

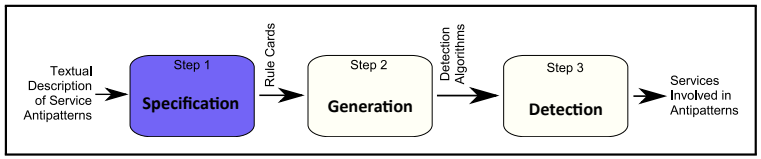
Symptoms and Consequences

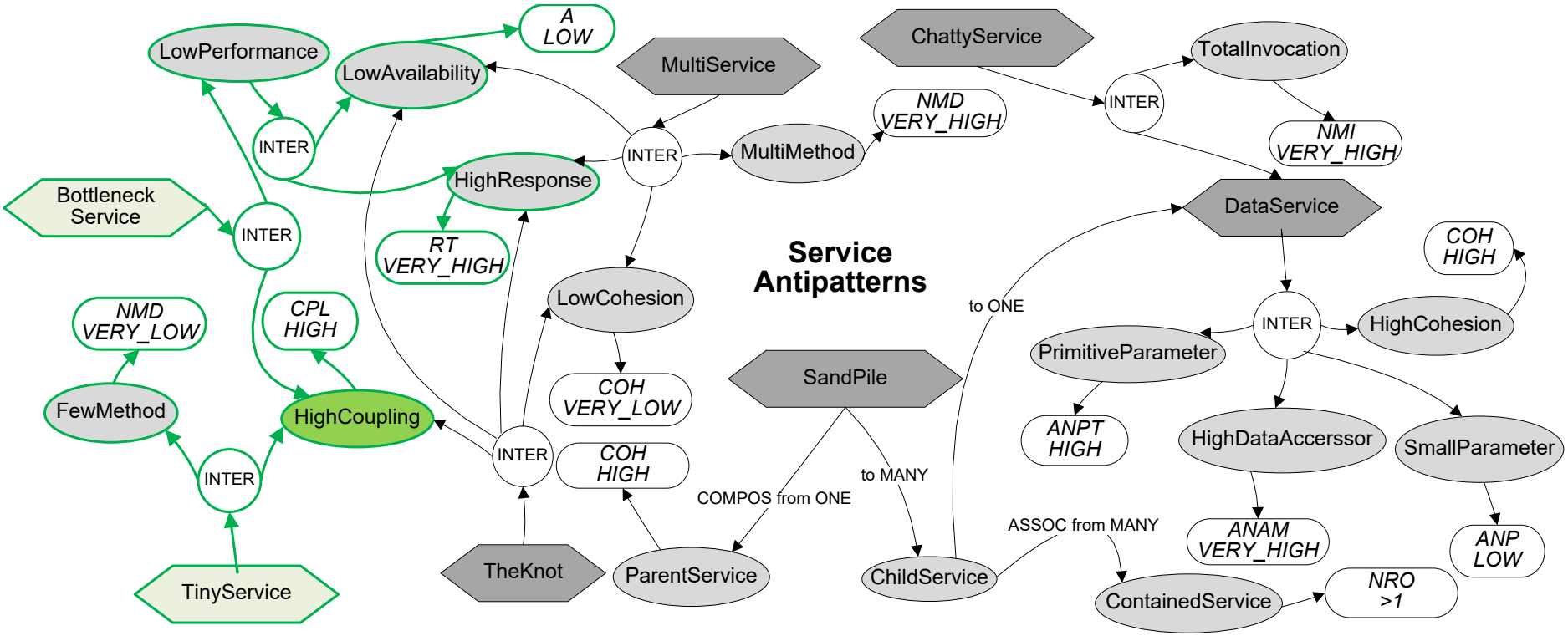
The most significant issue with the Tiny Service AntiPattern is that multiple services are required to support one, core business abstraction, and without tying all the processes together into one service, developers need to know all the different services to use, and how they should be coordinated and sequenced to support one overall business process workflow. There are a number of specific symptoms and consequences as follows:

- **A service interface has few methods (possibly only one).** When a service implements a small number of methods, this may indicate an incomplete service.
- **Multiple services support methods against the same core abstraction.** If there are methods in different services that perform different functions on the same abstraction (such as createOrder in one service and approveOrder in another), then this is a strong indicator that those services are incomplete and should be combined.

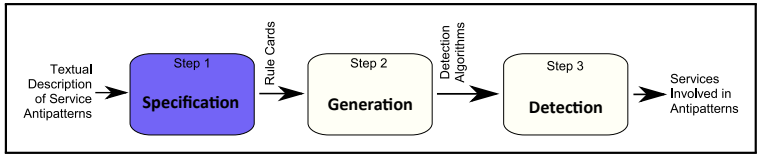


Domain Analysis for Antipatterns Specifications





BNF Grammar of Rule Cards for SODA



1 *rule_card* ::= RULE_CARD: *rule_cardName* { $(rule)^+$ };
2 *rule* ::= RULE: *ruleName* { *content_rule* };

3 *content_rule* ::= *metric* | *relationship* | *operator ruleType* $(ruleType)^+$
4 | RULE_CARD: *rule_cardName*

5 *ruleType* ::= *ruleName* | *rule_cardName*

6 *operator* ::= INTER | UNION | DIFF | INCL | NEG

7 *metric* ::= *id_metric ordi_value*
8 | *id_metric comparator num_value*

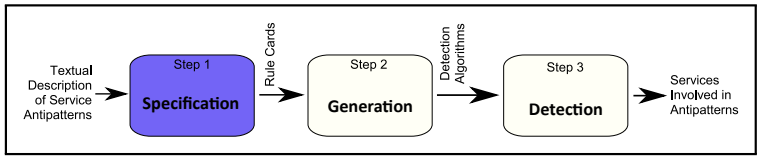
9 *id_metric* ::= ALS | ANAM/ANAO | ANIM | ANP | ANPT | ARIM | ARIO | ARIP
10 | COH | CPL | NCO | NI | NIR | NMD/NOD | NOPT
11 | NOR | NPT | NSE | NUM | NVMS | NVOS | RGTS | TNP
12 | A | NMI | NTMI | RT

13 *ordi_value* ::= VERY HIGH | HIGH | MEDIUM | LOW | VERY LOW
14 *comparator* ::= $< \mid \leq \mid = \mid \geq \mid >$

15 *relationship* ::= *relationType* FROM *ruleName cardinality* TO *ruleName cardinality*
16 *relationType* ::= ASSOC | COMPOS
17 *cardinality* ::= ONE | MANY | ONE_OR_MANY | *num_value* NUMBER_OR_MANY

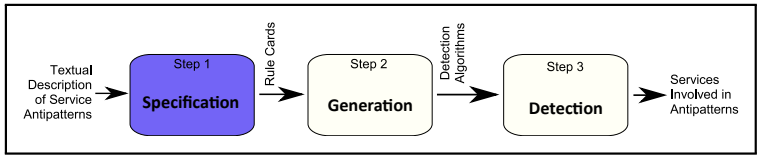
18 *rule_cardName*, *ruleName*, *ruleClass* $\in$ string
19 *num_value* $\in$ double

List of 27 Static and Dynamic Metrics



Metrics	Full Names	Type
A	Availability of a Service	dynamic
NMI	Number of Method Invocations	dynamic
NTMI	Number of Transitive Methods Invoked	dynamic
RT	Response Time of a Service	dynamic
ALS	Average Length of Signatures	static
ANP	Average Number of Parameters in Operations	static
ANPT	Average Number of Primitive Type Parameters	static
ANIO	Average Number of Identical Operations	static
ANAO	Average Number of Accessor Operations	static
ARIP	Average Ratio of Identical Port-Types	static
ARIO	Average Ratio of Identical Operations	static
ARIM	Average Ratio of Identical Messages	static
COH	Service Cohesion	static
CPL	Service Coupling	static
NCO	Number of CRUD Operations	static
NOD	Number of Operations Declared	static
NOPT	Number of Operations in Port-Types	static
NI	Number of Interfaces	static
NIR	Number of Incoming References	static
NOR	Number of Outgoing References	static
NPT	Number of Port-Types	static
NSE	Number of Services Encapsulated	static
NUM	Number of Utility Methods	static
NVMS	Number of Verbs in Message Signatures	static
NVOS	Number of Verbs in Operation Signatures	static
RGTS	Ratio of General Terms in Signatures	static
TNP	Total Number of Parameters	static

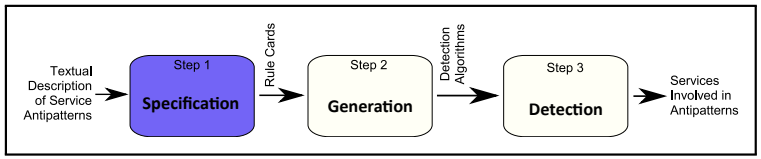
Rule for **Tiny Service** Antipattern




```
1 RULE_CARD: TinyService {
2    RULE: TinyService {INTER FewOperation HighCouplingORLowCohesion};
3    RULE: FewOperation {NOD VERY_LOW};
4    RULE: HighCouplingORLowCohesion {UNION HighCoupling LowCohesion};
5    RULE: HighCoupling {CPL HIGH};
6    RULE: LowCohesion {COH LOW};
7 };
```

*NOD = Number of Operations Defined; CPL = Coupling; COH = Cohesion;

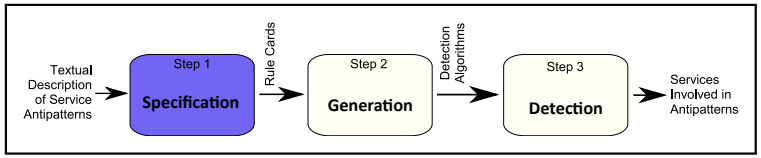
Rule for **God Component** Antipattern




```
1 RULE_CARD: GodComponent {
2    RULE: GodComponent {INTER HighEncapsulation MultiMethod HighParameter};
3    RULE: HighEncapsulation {NOSE HIGH};
4    RULE: MultiMethod {NMD VERY_HIGH};
5    RULE: HighParameter {TNP VERY_HIGH};
6 };
```

*NOSE = Number of Services Encapsulated; NMD = Number of Methods Defined; TNP = Total Number of Parameters;

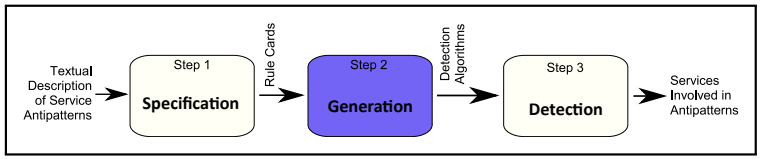
Rule for **Forgetting Hypermedia** Antipattern

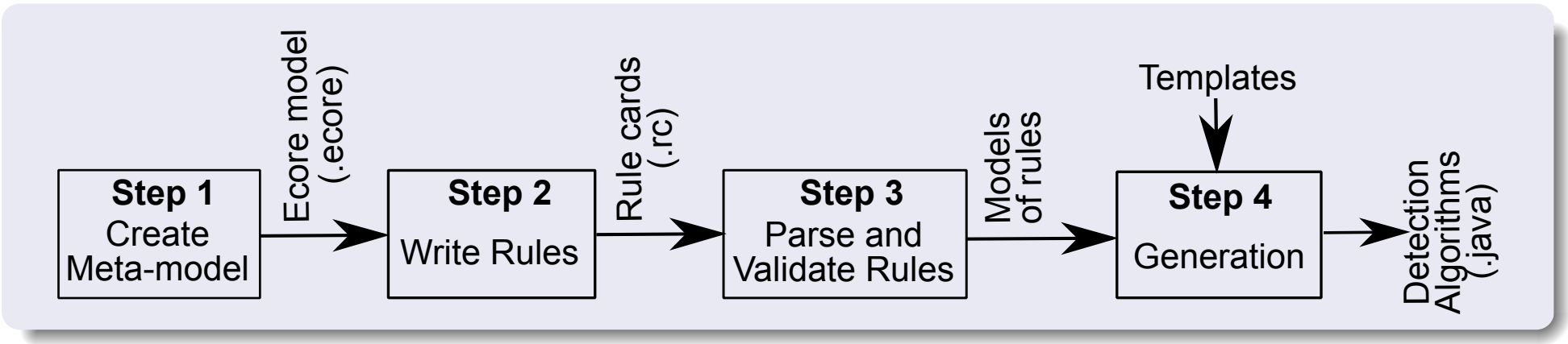



```
1 RULE_CARD: ForgetHyperMedia {
2     RULE: ForgetHyperMedia { UNION GetRequestLink PostRequestLink };
3     RULE: GetRequestLink { INTER HttpMethodGet NoLinkGet };
4     RULE: HttpMethodGet { HM = 'GET' };
5     RULE: NoLinkGet { UNION NoBodyLink NoHeaderLink };
6     RULE: NoHeaderLink { HL = NULL };

7     RULE: PostRequestLink { INTER HttpMethodPost NoLinkPost };
8     RULE: NoLinkPost { INTER NoBodyLink NoLocationHeader };
9     RULE: HttpMethodPost { HM = 'POST' };
10    RULE: NoLocationHeader { 'Location' ∉ ResponseHeader };

11    RULE: NoBodyLink { TLB = 0 };
12 };
```

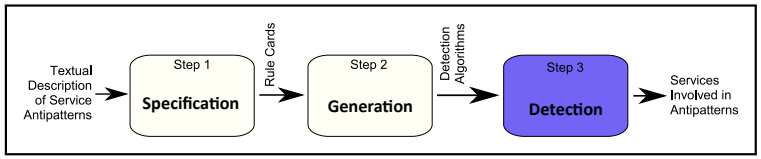
*HM = HTTP Method; HL = Hyperlinks; TLB = Total Number of Links in Body;

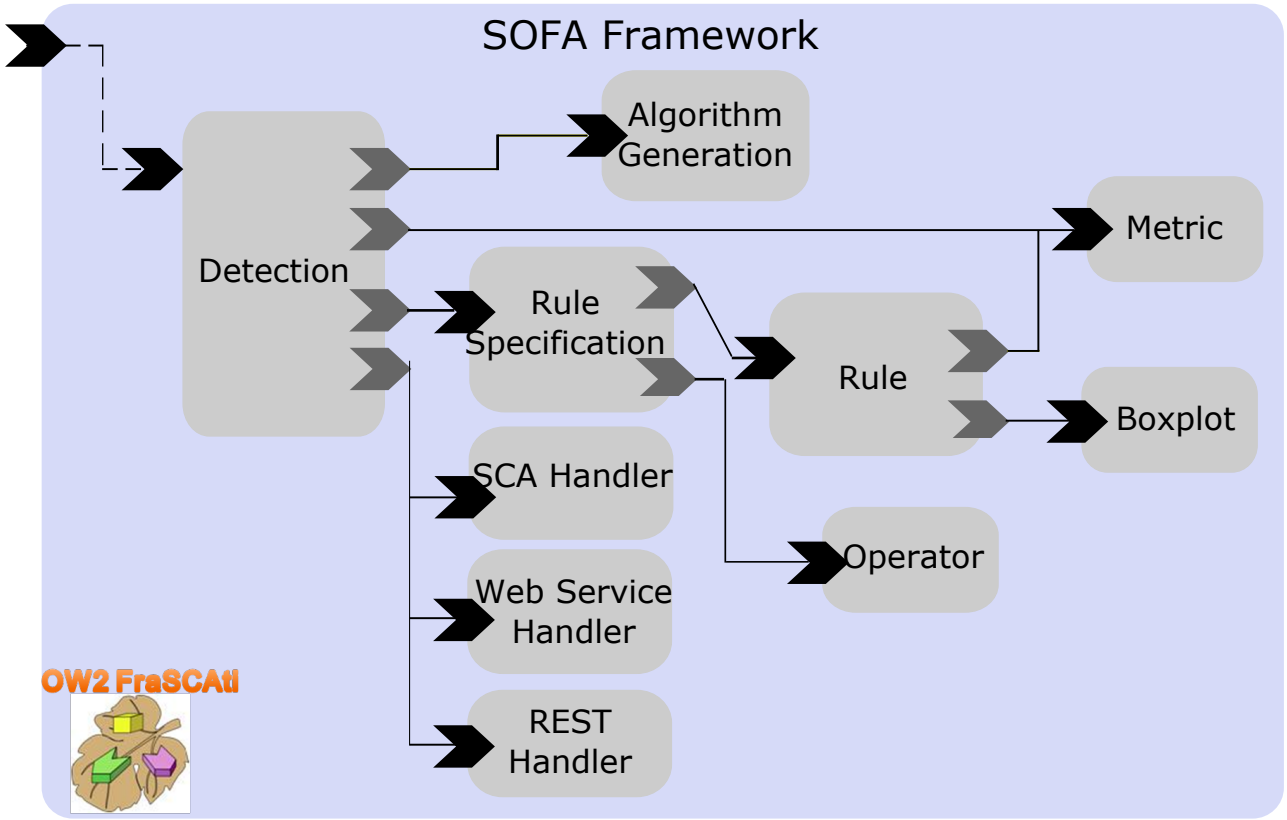
Detection Algorithms Generation Steps



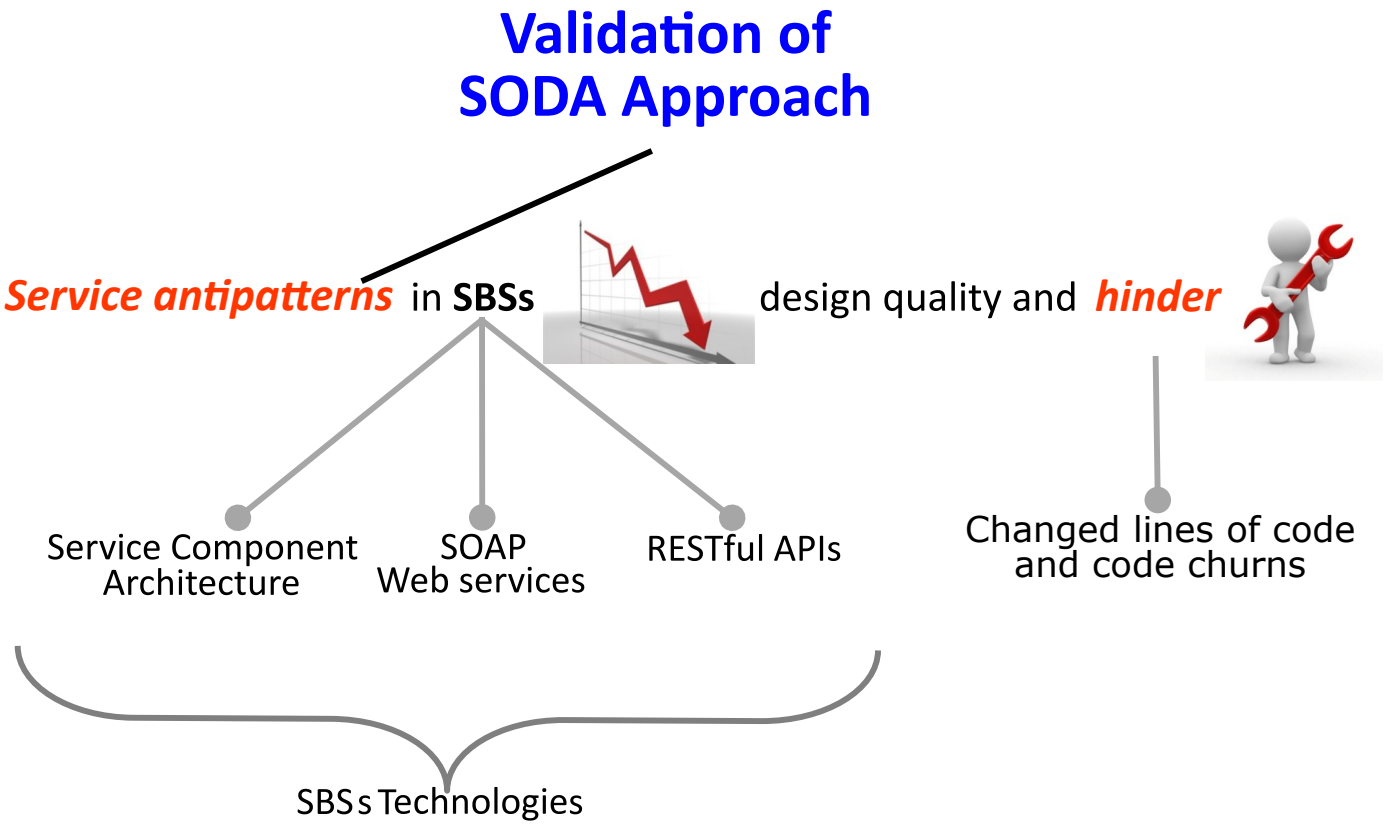


SOFA (Service Oriented Framework for Antipatterns)





Contribution 3



Validation

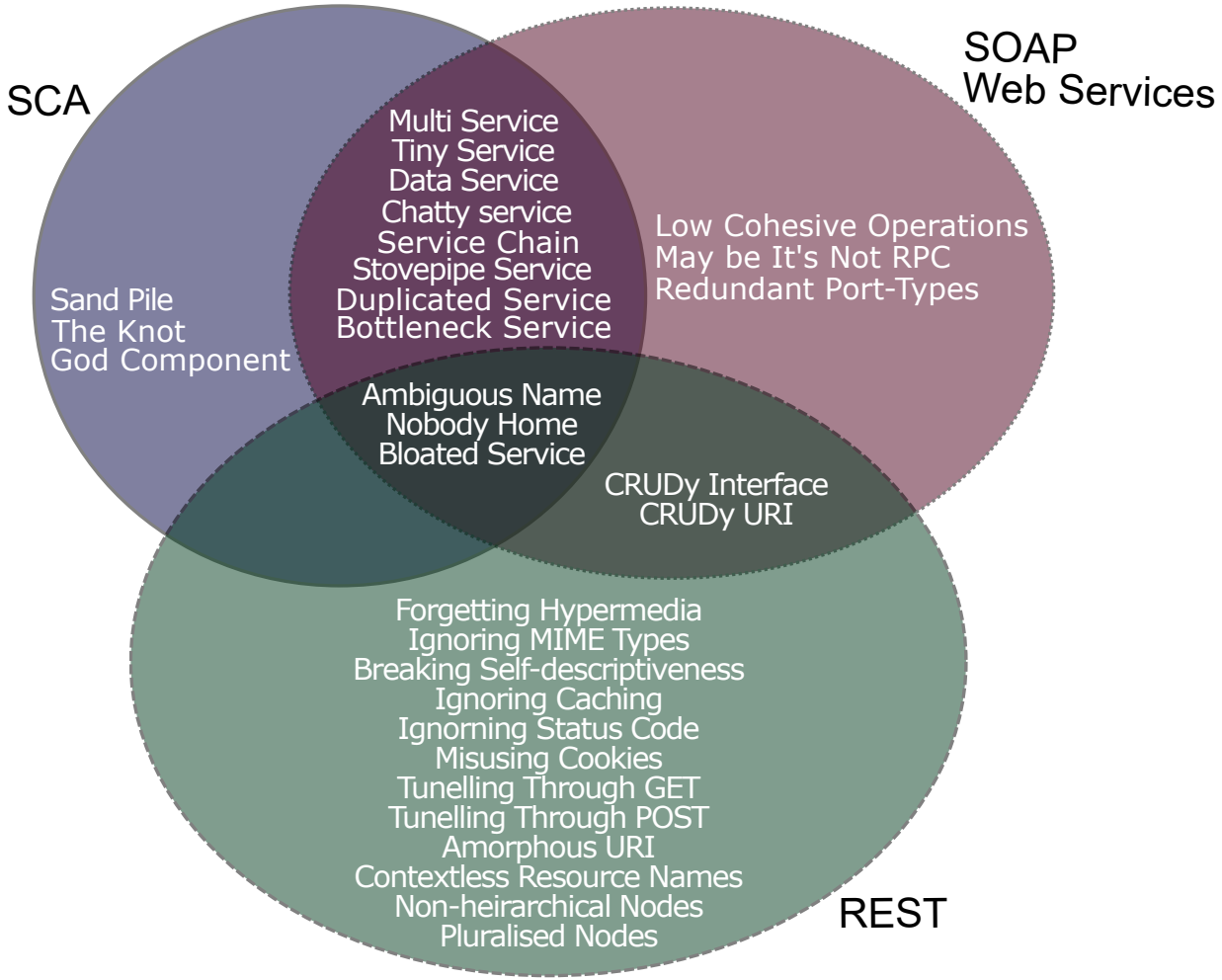
Research Question

Can we efficiently specify and detect service antipatterns in different development technologies and architectural styles of service-based systems?

Four experimental assumptions:

- A1. Generality
- A2. Accuracy
- A3. Extensibility
- A4. Performance

Subject: 31 Service Antipatterns



Objects: Service-based Systems

- Service Component Architecture
  - Home-Automation (13 SCA components)
  - FraSCAti OW2 (91 SCA components)
- SOAP Web services
  - 13 Weather SOAP Web services
  - 109 Finance SOAP Web services
- RESTful APIs

Validation Process

Five steps

1. Specify service antipatterns

Validation Process

Five steps

1. Specify service antipatterns
2. Generate detection algorithms
   - Implement REST clients and invoke 309 methods

Validation Process

Five steps

1. Specify service antipatterns
2. Generate detection algorithms
   - Implement REST clients and invoke 309 methods
3. Apply detection algorithms automatically
   - On Home-Automation, FraSCAti, and Web services
   - On REST requests-responses and on REST request URIs

Validation Process

Five steps

1. Specify service antipatterns
2. Generate detection algorithms
   - Implement REST clients and invoke 309 methods
3. Apply detection algorithms automatically
   - On Home-Automation, FraSCAti, and Web services
   - On REST requests-responses and on REST request URIs
4. Manually validate detection results

Validation Process

Five steps

1. Specify service antipatterns
2. Generate detection algorithms
   - Implement REST clients and invoke 309 methods
3. Apply detection algorithms automatically
   - On Home-Automation, FraSCAti, and Web services
   - On REST requests-responses and on REST request URIs
4. Manually validate detection results
5. Use precision, recall, and F_1-measure as detection accuracy measure

Validation of Detection Results

- Service Component Architecture (SCA)
  - Home-Automation: **7** execution scenario; **3** undergraduate students
  - FraSCAti: **5** execution scenario; **Core** development team
- SOAP Web Services
  - **2** graduate students
- RESTful APIs
  - **309** REST requests-responses
  - **3** professionals and **1** graduate student

Detection Results

Detection results for

- Tiny Service
- God Component
- Forgetting Hypermedia

Detection of **Tiny Service** Antipattern

Service Antipatterns	Applicable SBS Technology	Identified Service(s)	Metrics/Occurrences	Detection Time	Precision	Recall	F_1
Tiny Service	SCA *(Home-Automation)*	MediatorDelegate	NMD=1;CPL=0.44;NOR=4;	0.194s	[6/7] 85.71%	[6/6] 100%	92.31%
	SCA *(FraSCAti)*	sca-parser	NMD=1;CPL=0.56;	0.067s			
	Web services	SrtmWsPortType	NOD=2;COH=0.0;	0.945s			
		Hydro1KWsPortType					
		ShadowWsPortType					
		XigniteTranscripts	NOD=4;COH=0.125;				
		BGCantorUSTreasuries	NOD=3;COH=0.083;				

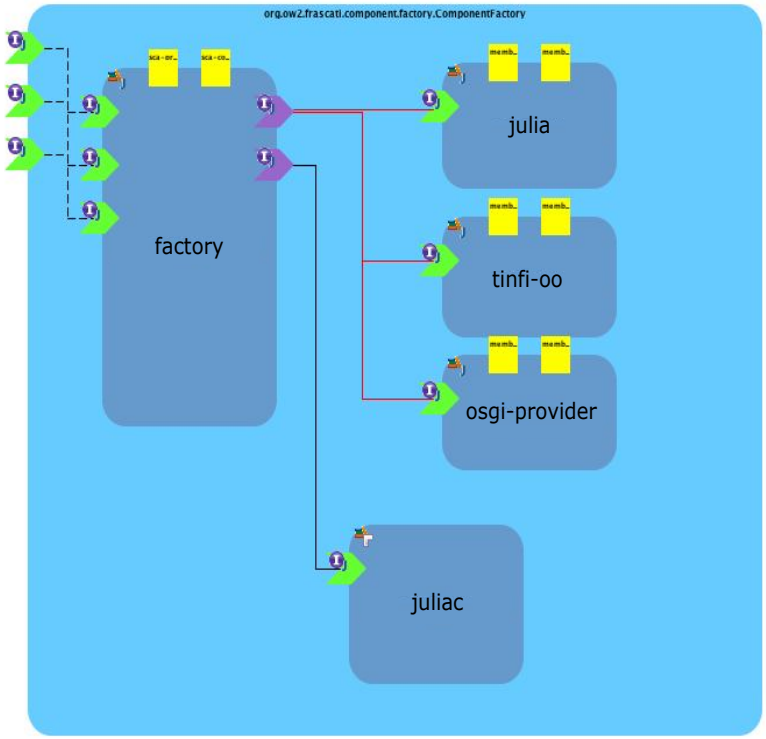
Example of **Tiny Service** Antipattern

```
<portType name='Hydro1KWsPortType'>
 <operation name='GetCapabilities' parameterOrder='GetCapabilitiesRequest'>
  <input message='tns:Hydro1kWsPortType_GetCapabilities'></input>
  <output message='tns:Hydro1kWsPortType_GetCapabilitiesResponse'></output>
 </operation>
 <operation name='GetMap' parameterOrder='GetMapRequest'>
  <input message='tns:Hydro1kWsPortType_GetMap'></input>
  <output message='tns:Hydro1kWsPortType_GetMapResponse'></output>
 </operation>
</portType>
```

Detection of **God Component** Antipattern

Service Antipatterns	Applicable SBS Technology	Identified Service(s)	Metrics/Occurrences	Detection Time	Precision	Recall	F_1
God Component	SCA *(FraSCAti)*	FraSCAti	NOSE=6;NMD=12;TNP=12	0.069s	[2/2] 100%	[2/2] 100%	100%
		component-factory	NOSE=5;NMD=7;TNP=12				
Sand Pile	SCA *(Home-Automation)*	HomeAutomation	NCS=13;ANP=1;ANPT=1;ANAM=100%;COH=0.17	0.184s	[1/1] 100%	[1/1] 100%	100%
The Knot	SCA *(Home-Automation)*	IMediator	COH=0.027;NIR=7;NOR=7;CPL=1.0;RT=57ms	0.412s	[2/3] 66.67%	[2/2] 100%	80%
		PatientDAO	COH=0.027;NIR=7;NOR=7;CPL=1.0;RT=57ms				
	SCA *(FraSCAti)*	sca-parser	CPL=0.84;COH=0.08;RT=44ms	0.07s			

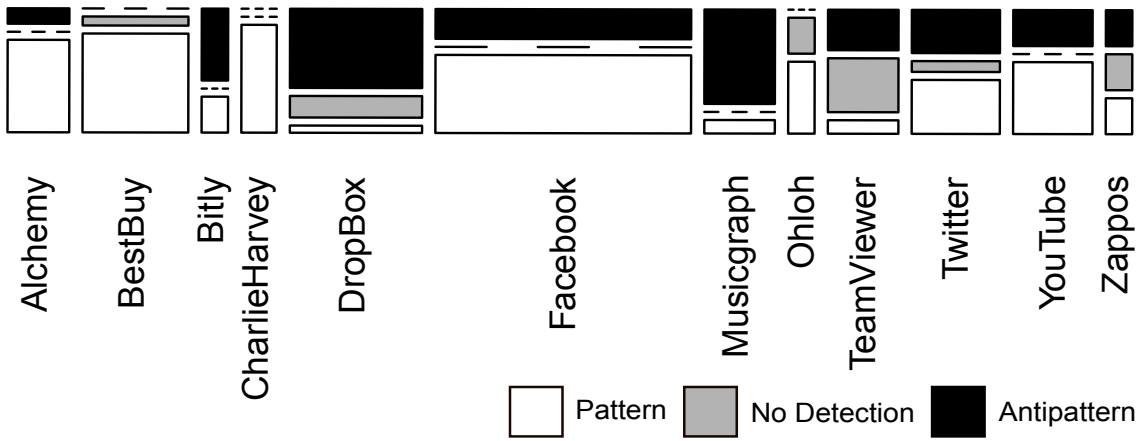
Example of **God Component** Antipattern




```
package org.ow2.frascati.component.factory.api;

import org.objectweb.fractal.api.Component;
import org.objectweb.fractal.api.type.ComponentType;

import org.osoa.sca.annotations.Service;

@Service
public interface ComponentFactory
{

   void generateMembrane(ComponentFactoryContext context, ComponentType componentType, String
 membraneDesc,
       String contentClass) throws FactoryException;


  void generateScaPrimitiveMembrane(ComponentFactoryContext context, ComponentType
 componentType, String classname)
       throws FactoryException;

  void generateScaCompositeMembrane(ComponentFactoryContext context, ComponentType
 componentType)
       throws FactoryException;

  Component createComponent(ComponentFactoryContext context, ComponentType componentType,
String membraneDesc,
      Object contentClass) throws FactoryException;

  Component createScaPrimitiveComponent(ComponentFactoryContext context, ComponentType
componentType, String classname)
      throws FactoryException;

  Component createScaCompositeComponent(ComponentFactoryContext context, ComponentType
componentType)
      throws FactoryException;

  Component createScaContainer(ComponentFactoryContext context)
      throws FactoryException;

}
```

Detection of **Forgetting Hypermedia** Antipattern

REST APIs	(12)BestBuy	(15)DropBox	(29)Facebook	(10)Twitter	(9)YouTube	(115) Total	precision-recall	Average Precision-Recall	Detection Time
REST Antipattern									
Forgetting Hypermedia	0/0	9/10	8/8	4/4	2/3	36/38	p	94.58%	19.54s
	0/0	9/9	8/8	4/4	2/2	36/36	r	100%	



Example of **Forgetting Hypermedia** Antipattern

Method name: ***youtube_videos_list***
Path: ***/videos***

Request:
Header:

{
cache-control=[no-cache],
content-type=[application/xml],
connection=[keep-alive],
host=[www.googleapis.com],
accept=[application/xml],
get /youtube/v3/videos?id=srqtw-sjdgw&part=id&access_token=ya29.gabeu_oiqsat4boaaac2wrtgh5ba_1gftd5edos6qykgm6t46rieezp_znyg4g
http/1.1=[null],
user-agent=[Apache CXF 2.4.0],
pragma=[no-cache]
}

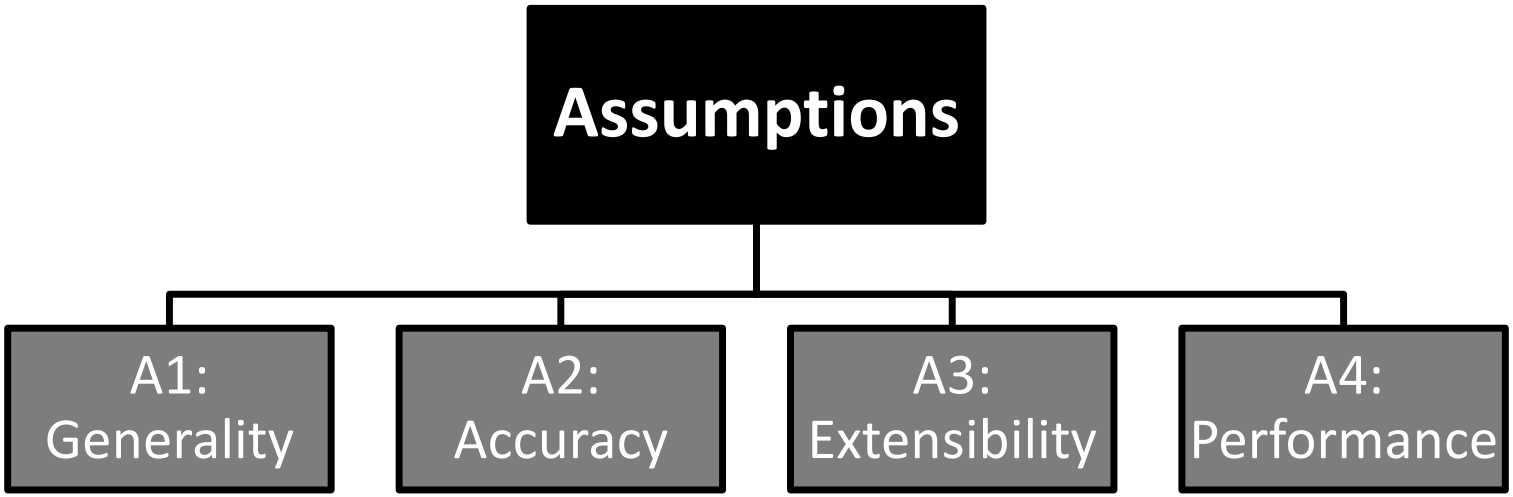
Response:
Status Code: 200
Header:

{
cache-control=[private, max-age=300, must-revalidate, no-transform],
content-type=[application/json; charset=UTF-8],
x-frame-options=[SAMEORIGIN],
x-content-type-options=[nosniff],
x-xss-protection=[1; mode=block],
expires=[Fri, 16 May 2014 15:23:34 GMT],
etag=["ePFRUfYBkeQ2ncpP9OLHKB0fDw4/KD3Jx-OynnFf5lxH9dRdz0sgL-k"],
content-length=[324],
server=[GSE],
alternate-protocol=[443:quic],
date=[Fri, 16 May 2014 15:23:34 GMT]}

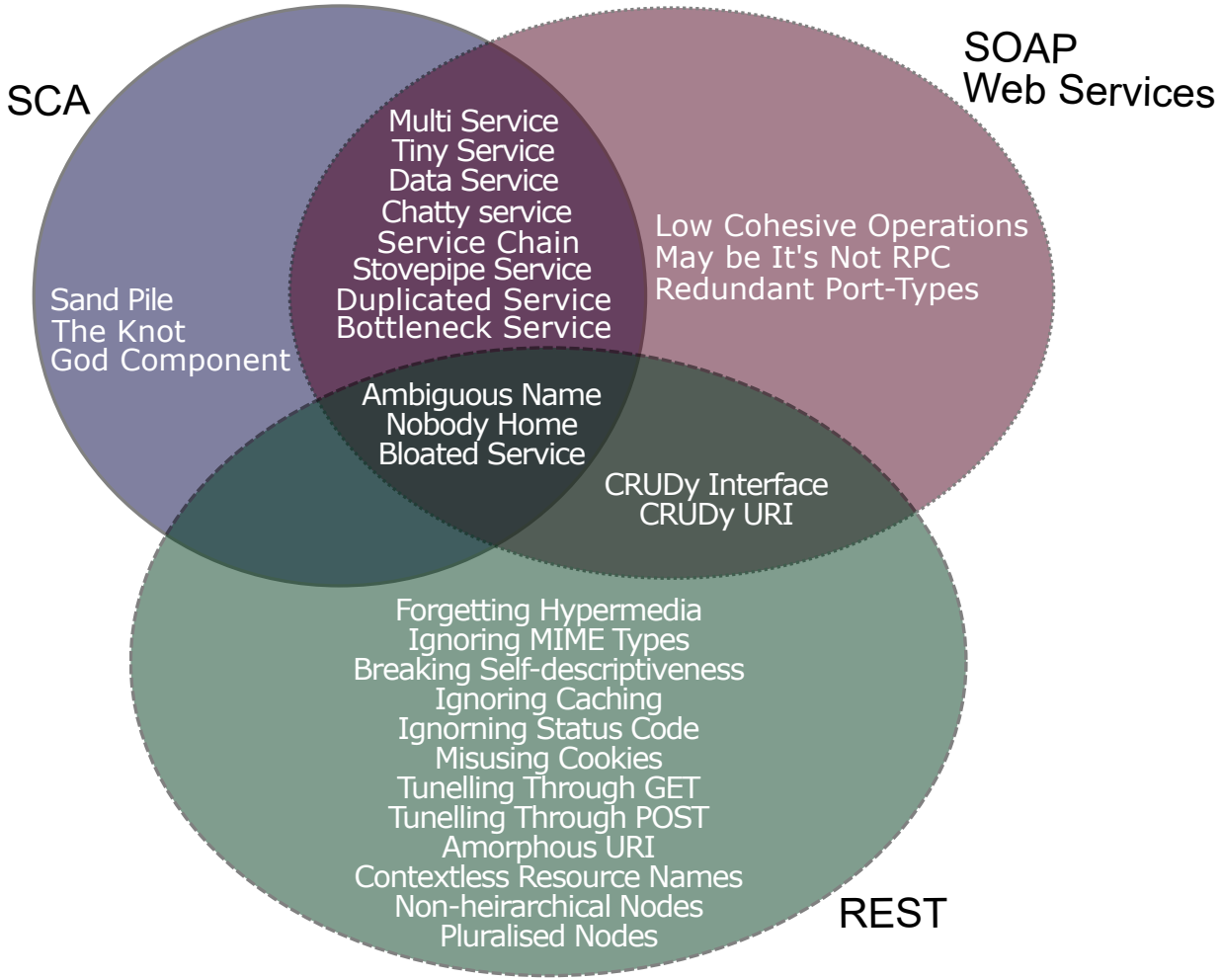
Body:

{
"kind": "youtube#videoListResponse",
"etag": "\"ePFRUfYBkeQ2ncpP9OLHKB0fDw4/KD3Jx-OynnFf5lxH9dRdz0sgL-k\"",
"pageInfo": {
"totalResults": 1,
"resultsPerPage": 1 },
"items": [{
"kind": "youtube#video",
"etag": "\"ePFRUfYBkeQ2ncpP9OLHKB0fDw4/9hUt36nrZXNpfqDhYP_bu7W-d3U\"",
"id": "SRQtW-sjDGw"
}]
}

Verifying Four Assumptions



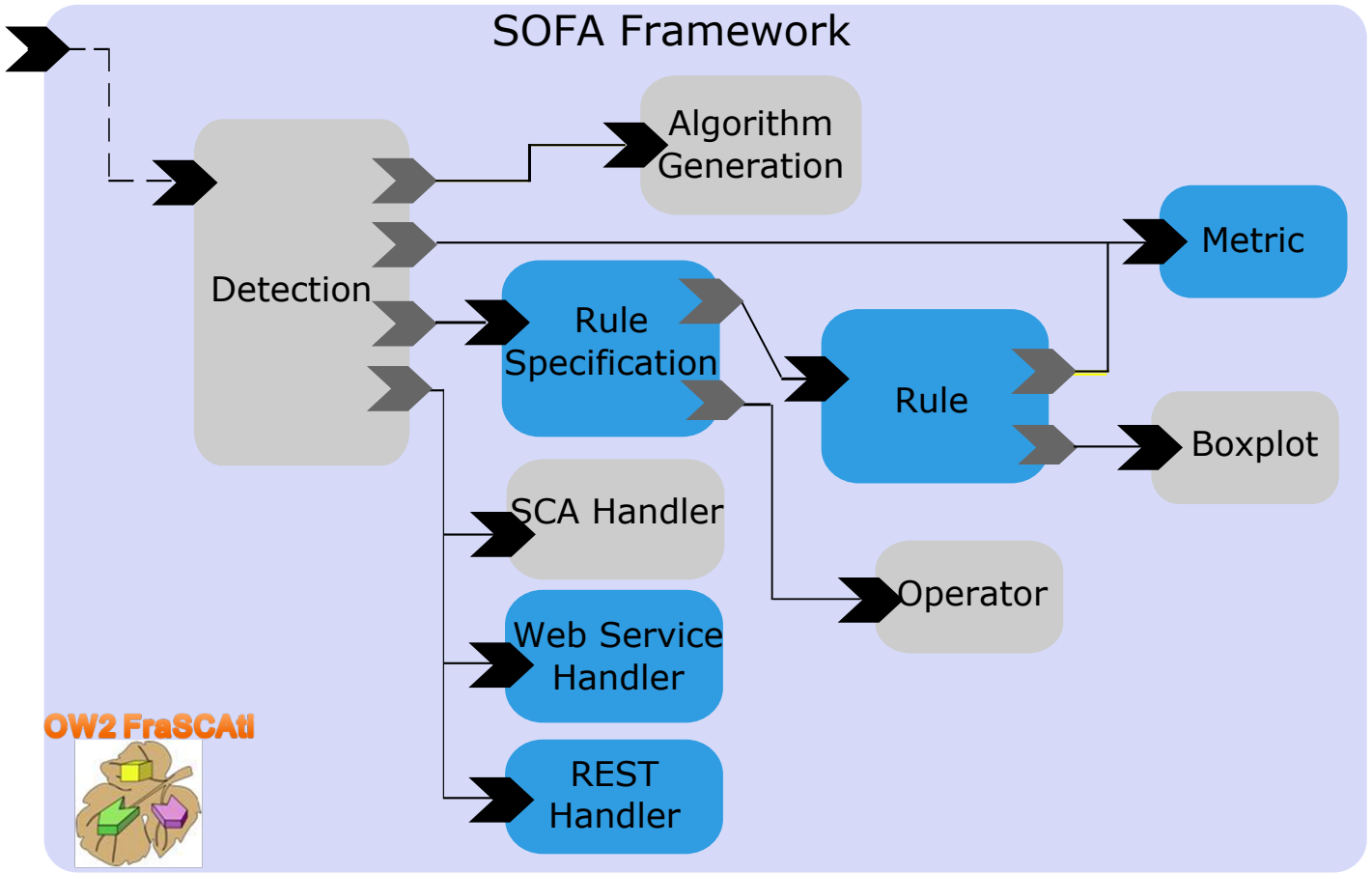
A1: Generality of DSL



A2: Accuracy of Detection Algorithms

Antipatterns Groups	Average Precision	Average Recall	Average F-1 measure
SCA ∩ REST ∩ Web services	93.33%	100%	96.3%
SCA ∩ Web services	82.59%	95.83%	86.34%
REST ∩ Web services	75%	100%	83.33%
SCA	88.89%	100%	93.3%
Web services	100%	100%	100%
REST Req/Res Antipatterns	82.81%	90.4%	86.44%
REST Req/Res Patterns	100%	99.76%	99.88%
REST Linguistic Antipatterns	81.4%	78%	79.66%
Average	**88%**	**95.5%**	**90.66%**

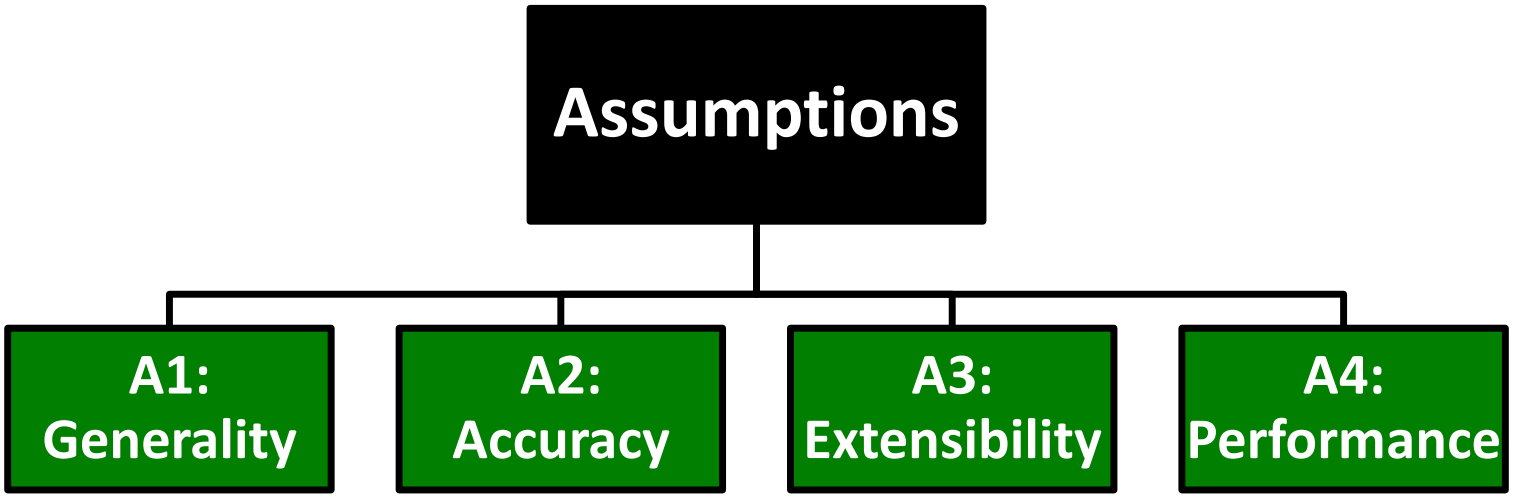
A3: Extensibility of SOFA Framework



A4: Performance of Detection Algorithms

Antipatterns Groups	Average Detection Times
SCA ∩ REST ∩ Web services	0.511s
SCA ∩ Web services	9.09s
REST ∩ Web services	18.98s
SCA	0.184s
Web services	144.72s
REST Syntactic Antipatterns	22.06s
REST Syntactic Patterns	20.44s
REST Linguistic Antipatterns	0.70s
Average	**27.09s**

Verifying Four Assumptions



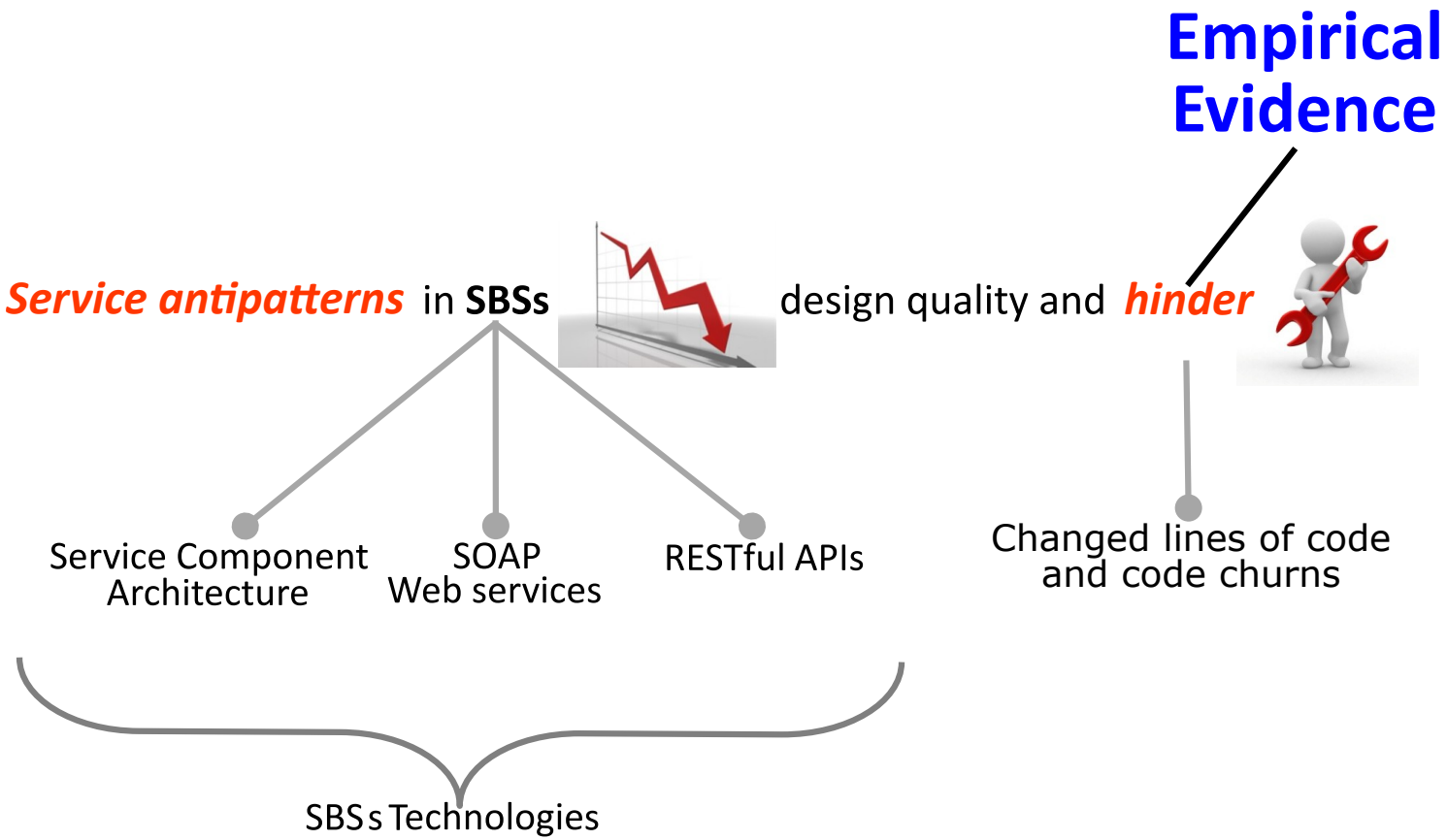
Answer to the Research Question

Research Question

Can we efficiently specify and detect service antipatterns in different development technologies and architectural styles of service-based systems?

Answer: With our proposed unified SODA approach that encompasses the unified abstraction and the service DSL, we can effectively specify and detect service antipatterns in different SBSs technologies in terms of accuracy and performance.

Contribution 4



Impact of Service Antipatterns on SBSs

Research Question

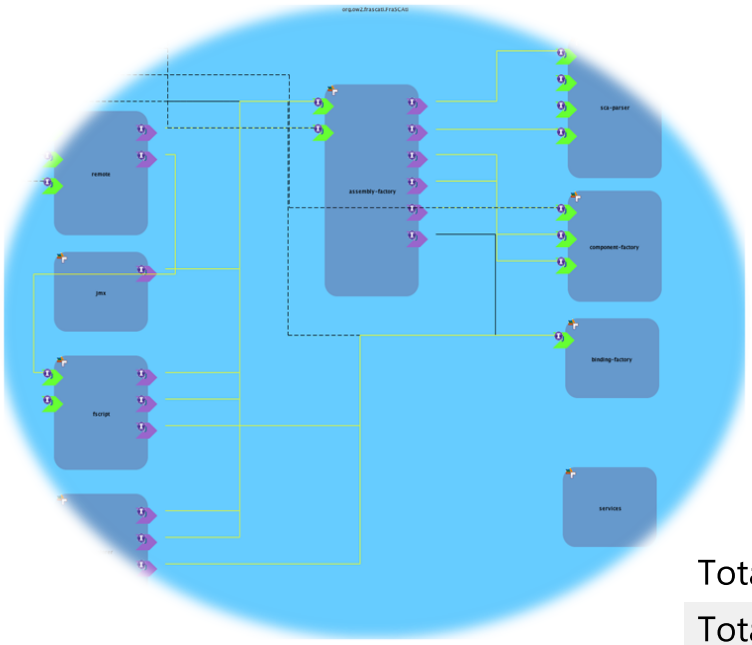
What are the impact of service antipatterns and patterns on the maintenance and evolution of service-based systems?

- **RQ**$_{2.1}$ - What is the relation between service antipatterns and change-proneness?
- **RQ**$_{2.2}$ - What is the relation between particular kinds of service antipatterns and change-proneness?

Study Subjects: Service Antipatterns

	Names	Detected Instances	Involved Java Source Files
Antipatterns	Bloated Service	3	25
	Bottleneck Service	2	24
	God Component	2	4
	Multi Service	1	5
	Nobody Home	4	12
	Service Chain	3	10
	The Knot	1	24
	Tiny Service	1	24

Study Object: FraSCAti OW2

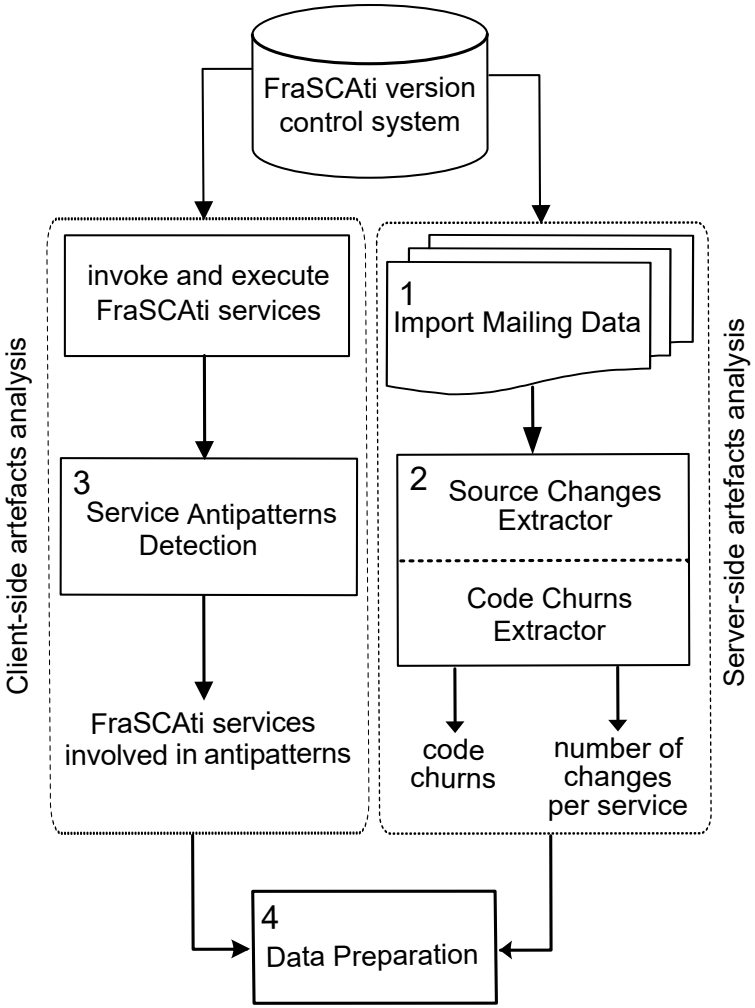


http://frascati.ow2.org/

Total Services	130 (62 analysed)
Total Size	170 KLOC
Total Changed Files	15,863*
Total Java Source Files	9,020*
Total Changes	71,151*
Total Code Churns	62,676,363*

*Entire commit history

Study Approach



Study Variables

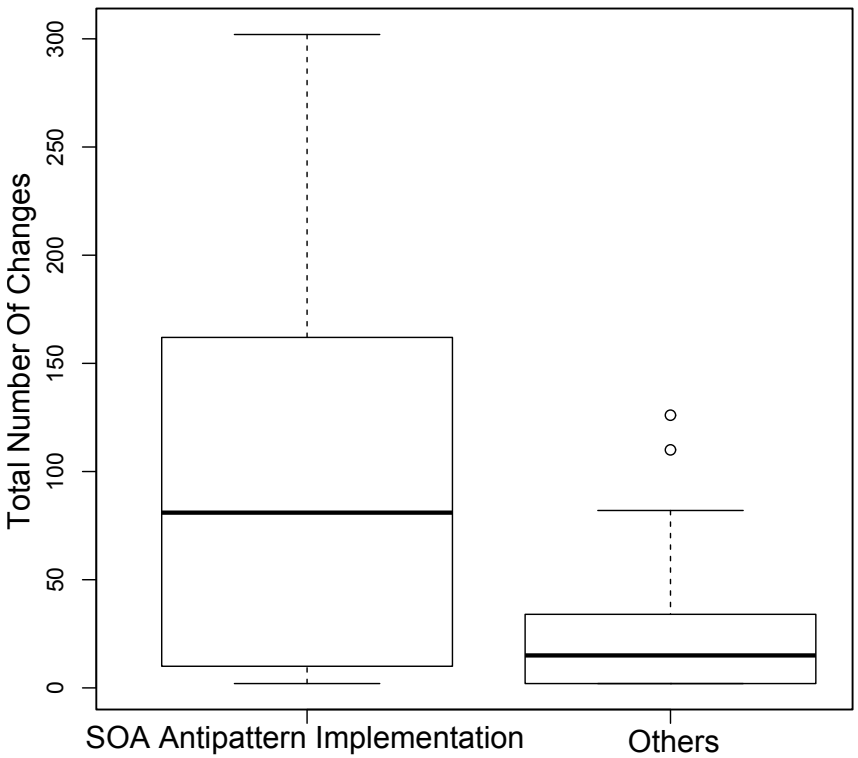
Independent variables

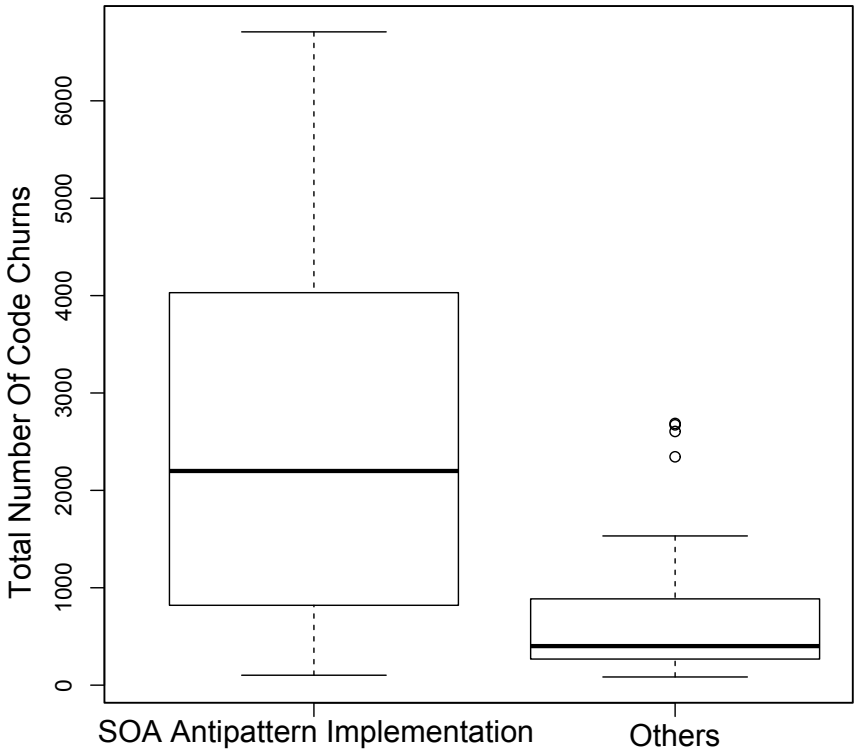
- Eight service antipatterns
  - $\mathbf{f}_i^1$: file **i** involved in the implementation of any antipattern (RQ1)
  - $\mathbf{f}_{i,j}^2$: file **i** involved in the implementation of antipattern **j** (RQ2)

Dependent variables

- Change-proneness of services' source files
  - Total number of changes as $\mathbf{c}_i$
  - Total number of code churns as $\mathbf{d}_i$

$RQ_{2.1}$: Antipatterns are More Change-prone

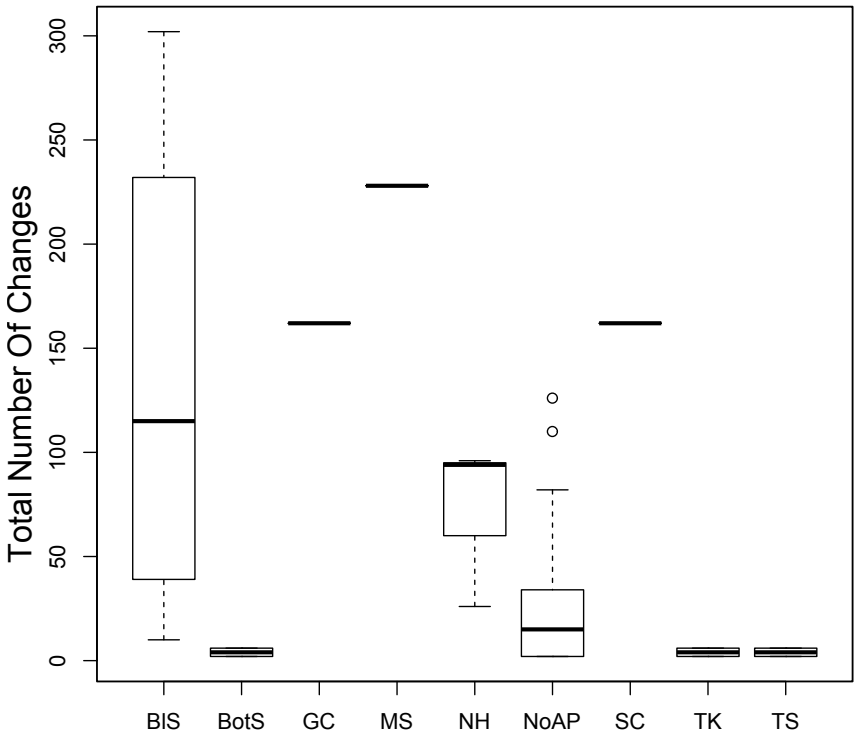


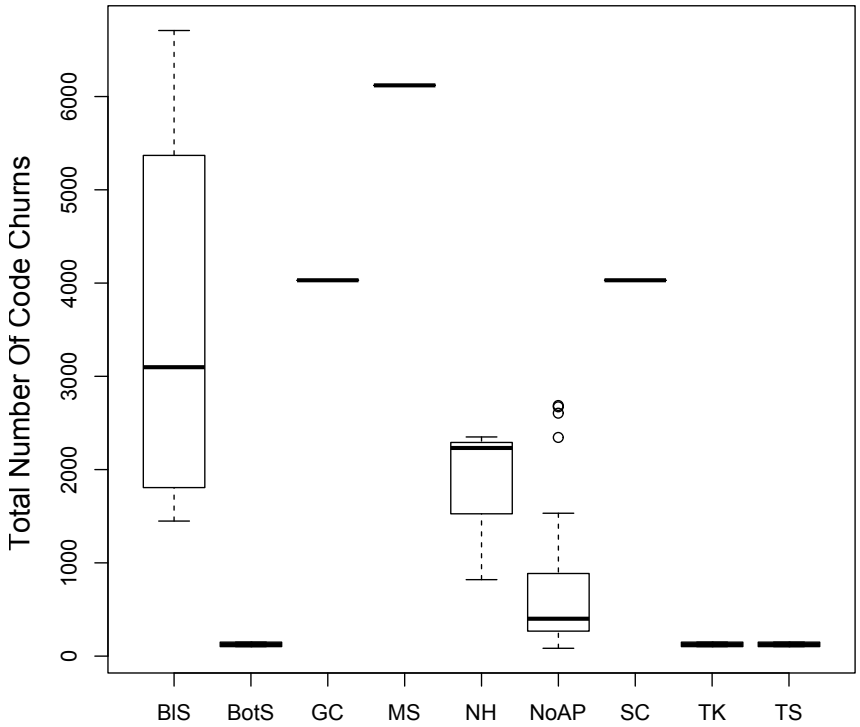


Treatment Groups	Treatment Types	p-value	Cliff's δ
antipatterns $\sim$ non-antipatterns	total number of code churns	**0.015**	**0.515 (large)**
antipatterns $\sim$ non-antipatterns	total number of changes	**0.011**	**0.496 (large)**

$RQ_{2.2}$: Antipatterns are Not Equally Change-prone

God Component, **Multi Service**, and **Service Chain** antipatterns are more change-prone.





Treatment Groups	p-value
total number of code churns $\sim$ antipattern	**0.0002**
total number of changes $\sim$ antipattern	**0.01003**

Answer to the Research Question

Research Question

What are the impact of service antipatterns and patterns on the maintenance and evolution of Service-based systems?

Answer: The services involved in antipatterns, in terms of their implementations, are more change-prone than the services that are not involved in any antipattern.

Conclusion: Our Challenges

Challenge 2

- Only textual descriptions
- No specifications

Challenge 3

- No dedicated unified approach
- No unified framework

Challenge 4

- No empirical evidence

Service antipatterns in **SBSs**  design quality and ***hinder*** 

Service Component Architecture

SOAP Web services

RESTful APIs

Changed lines of code and code churns

Challenge 1

Conclusion: Our Contributions

Challenge 2

Service DSL

Challenge 3

Unified SODA Approach
SOFA Framework
Validation of SODA

Challenge 4

Empirical Evidence

Service antipatterns in **SBSs**  design quality and ***hinder*** 

Changed lines of code and code churns

Service Component Architecture

SOAP Web services

RESTful APIs

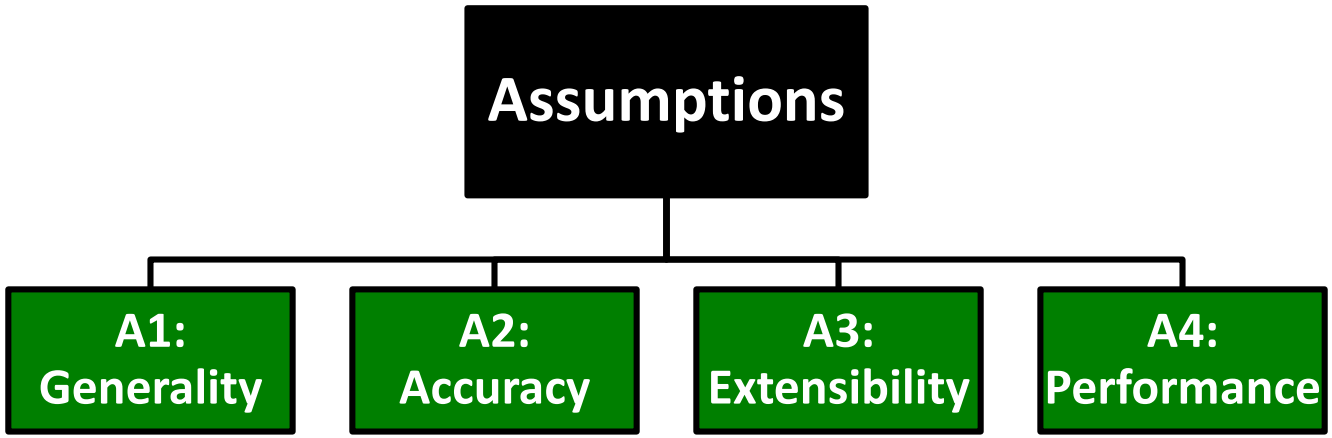
A unified abstraction combining different SBSs technologies

Challenge 1

Conclusion

Research Question 1

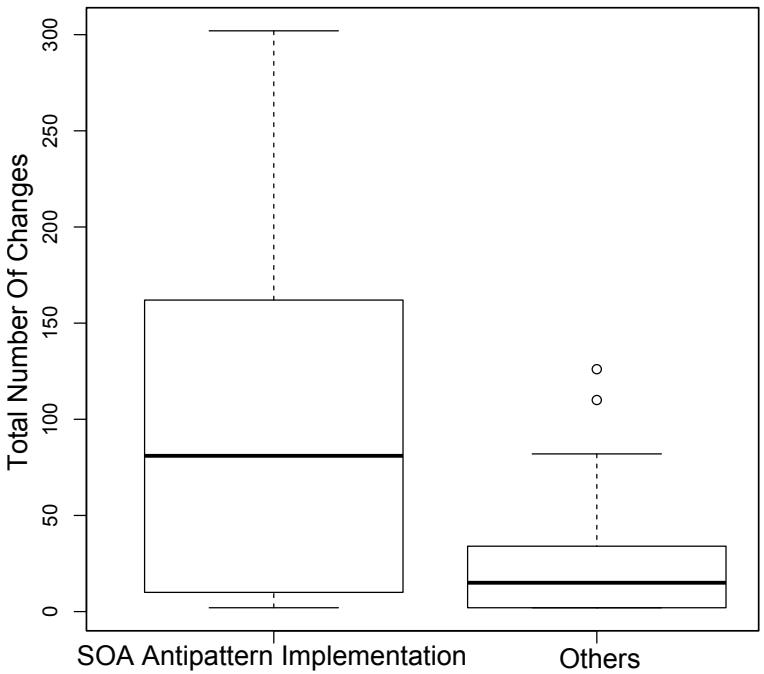
Can we efficiently specify and detect service antipatterns in different development technologies and architectural styles of service-based systems?

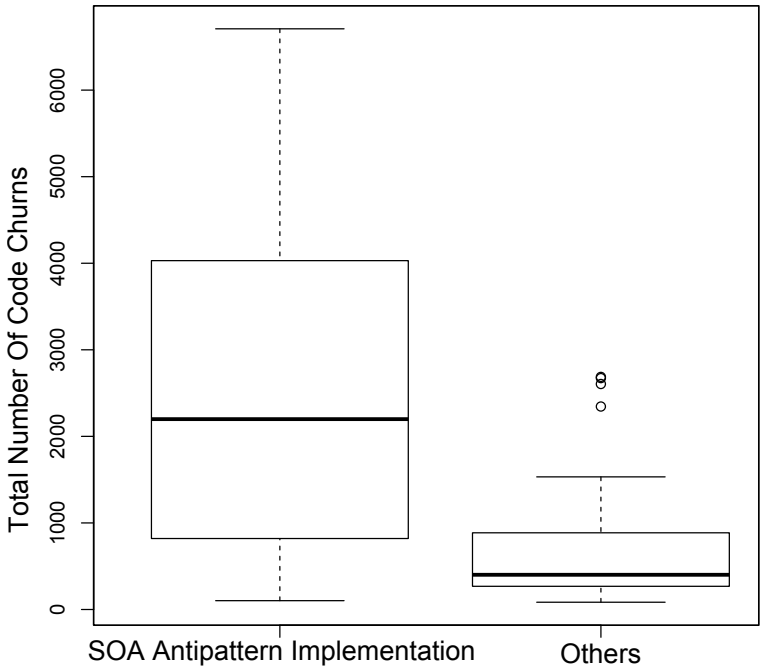


Conclusion

Research Question 2

What are the impact of service antipatterns and patterns on the maintenance and evolution of service-based systems?





Perspectives

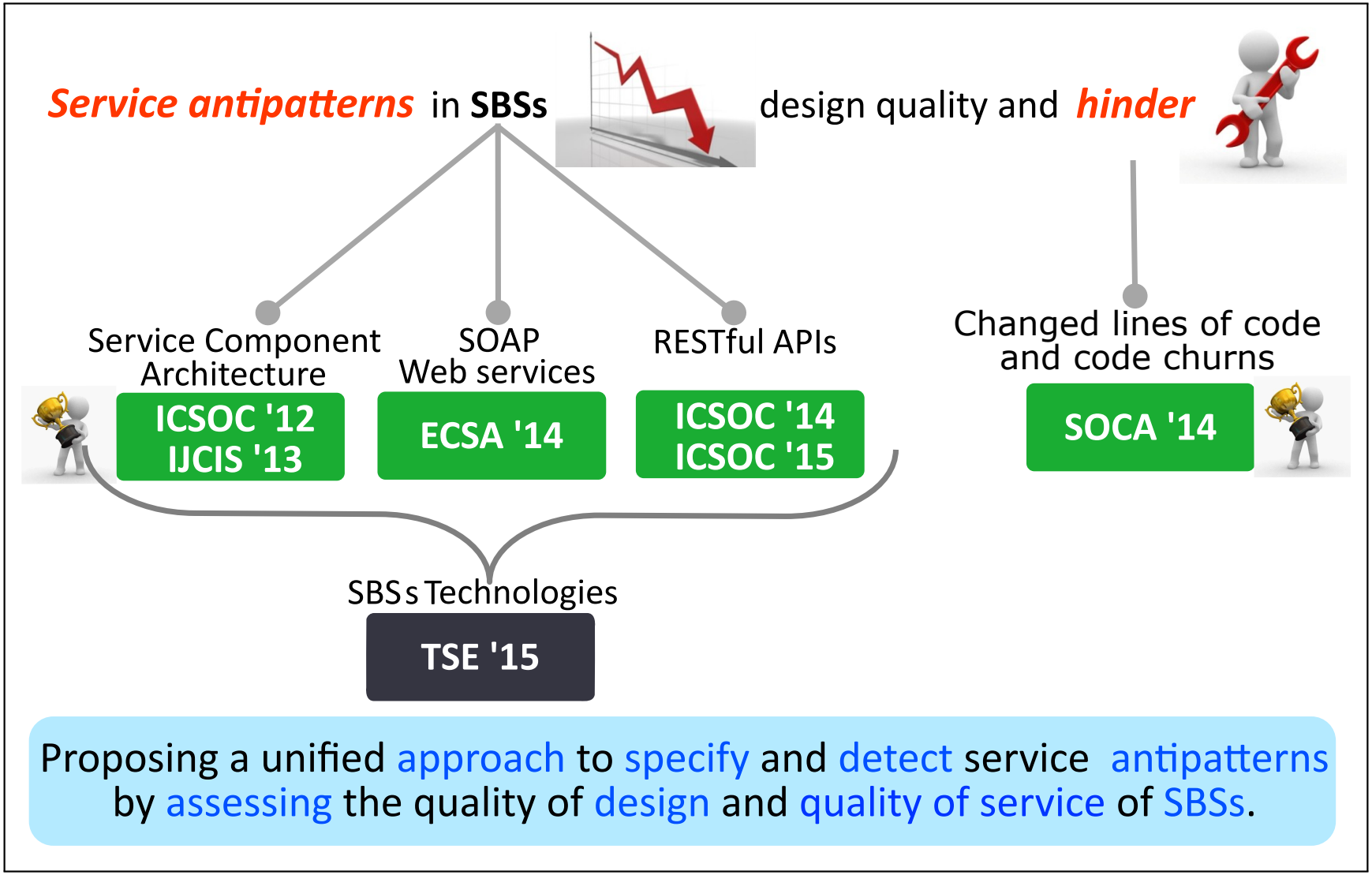
Short-term perspectives

- Replicate SODA: more RESTful APIs, more (anti)patterns
- Communicate the detection results with real developers
- Impact study on Web services and RESTful APIs
- Study the evolution of service antipatterns in SBSs

Long-term perspectives

- More experiments and analyse results with industrial partners
- Propose a corrective approach
- Impact of refactored service antipatterns

Thank You!


Service antipatterns in SBSs
design quality and hinder
Service Component Architecture
SOAP Web services
RESTful APIs
Changed lines of code and code churns
ICSOC '12
IJCIS '13
ECSA '14
ICSOC '14
ICSOC '15
SOCA '14
SBSs Technologies
TSE '15
Proposing a unified approach to specify and detect service antipatterns by assessing the quality of design and quality of service of SBSs.