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

- Gartner (2006)
Context: Service-based Systems (SBSs)
Poor but recurring design practices in SBSs cause more effort and design quality.
Service antipatterns in SBSs hinder design quality and
Service antipatterns in SBSs hinder design quality.
Service antipatterns in SBSs hinder design quality and code churns.

Service Component Architecture
SOAP Web services
RESTful APIs

SBSs Technologies

Francis PALMA (ÉPM, UQÀM)
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

J2EE AntiPatterns

SOA Design Patterns

Service Design Patterns

SOA with REST

SOA Patterns
Tiny Service Antipattern in SOAP Web Services

- Few Operations
- Low Cohesion
- High Coupling
God Component Antipattern in SCA

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

import org.objectweb.fratcal.api.Component;
import org.objectweb.fratcal.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

Francis PALMA (ÉPM, UQÂM)

Detection of Service Antipatterns

August 19, 2015 8 / 60

DropBox Server Response 1:

Header:
{
  x-frame-options=[SAMEORIGIN],
  x-dropbox-request-id=[b9a25269beb2c75fa77d7e21e1638bb9d],
  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-Requestld=[c64da98881e565a90a5dd9aece9f049]
}

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
    }
  ]
}

No links to follow...

DropBox Server Response 2:

Header:
{
  x-frame-options=[SAMEORIGIN],
  x-dropbox-request-id=[cd12e1e84432746448542b11b530071],
  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-Requestld=[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...

No links to follow...
Challenges and Problems

Service antipatterns in SBSs hinder design quality and changed lines of code and code churns.

- Service Component Architecture
- SOAP Web services
- RESTful APIs
Challenges and Problems

Service antipatterns in SBSs hinder design quality and

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

Changed lines of code and code churns
Service antipatterns in SBSs

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

Service Component Architecture

SOAP Web services

RESTful APIs

Changed lines of code and code churns

design quality and hinder

Web Service Interface

- Types
  - Messages
  - Port Types
    - Operations
      - Input
      - Output
    - Bindings
      - Service
        - Port
- XML-based
- SOAP-based
- WS*-standards
- WSDL specifications (remote)
- BPEL composition

Francis PALMA (ÉPM, UQÀM)
Detection of Service Antipatterns
August 19, 2015
Challenges and Problems

**Service antipatterns** in SBSs

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

- **SOAP Web services**
  - SOAP, REST, JMS, RMI
  - SC DL specifications (local)
  - XML or SDO (Service Data Object)

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

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

**Properties**

**Changed lines of code and code churns**

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

Francis PALMA (ÉPM, UQÀM)
Challenges and Problems

Service antipatterns in SBSs

Design quality and hinder

Challenge 1

No unified abstraction

Francis PALMA (ÉPM, UQÁM)
Detection of Service Antipatterns
August 19, 2015 9 / 60
Challenges and Problems

Challenge 2
- Only textual descriptions
- No specifications

Service antipatterns in SBSs

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

Service Component Architecture
SOAP Web services
RESTful APIs

Changed lines of code and code churns

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

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

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

Challenge 1
- Only textual descriptions
- No specifications

Francis PALMA (ÉPM, UQÀM)
Detection of Service Antipatterns
August 19, 2015 9 / 60
Challenges and Problems

**Challenge 1**

- Only textual descriptions
- No specifications

**Challenge 2**

- Service antipatterns in SBSs

**Challenge 3**

- No dedicated unified approach
- No unified framework

**Web Service Interface**

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

**REST Resource**

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

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

**Service Architecture**

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

**References**

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

**Changed lines of code and code churns**

**Design quality and hinder**
Challenges and Problems

**Challenge 1**
- Only textual descriptions
- No specifications

**Challenge 2**
- No dedicated unified approach
- No unified framework

**Challenge 3**
- No empirical evidence

**Challenge 4**
- No empirical evidence

---

**Service antipatterns in SBSs**

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

**SOAP Web services**

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

**Web Service Interface**
- Types
- Messages
- Port Types
- Operations
- Input
- Output
- Bindings
- Service
- Port

**REST Resource**
https://api.twitter.com/1.1/followers/ids.json

**Changed lines of code and code churns**

---

**Challenge 1**
- Only textual descriptions
- No specifications

**Challenge 2**
- No dedicated unified approach
- No unified framework

**Challenge 3**
- No empirical evidence

**Challenge 4**
- No empirical evidence

---

**Francis PALMA (ÉPM, UQÀM)**
Detection of Service Antipatterns
August 19, 2015 9 / 60
<table>
<thead>
<tr>
<th>Component-based Systems (CBS)</th>
<th>Object-Oriented Systems (OO)</th>
<th>Service Component Architecture (SCA)</th>
<th>SOAP Web services</th>
<th>RESTful APIs</th>
<th>Unified</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Khomh et al. (2011)</td>
<td></td>
<td>Mateos et al. (2011)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Stoianov et Sora (2010)</td>
<td></td>
<td>Rodriguez et al. (2010b)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Service antipatterns in SBSs

Service Component Architecture

SOAP Web services

RESTful APIs

Changed lines of code and code churns

Service antipatterns in SBSs hinder design quality and

No unified abstraction

Challenge 1

Changed lines of code and code churns

SOAP , REST, JMS, RMI

SCDL specifications (local)

XML or SDO (Service Data Object)

One additional layer of components

References

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

Web Service Interface

Rest Resource

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

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

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

Francis PALMA (ÉPM, UQÀM) Detection of Service Antipatterns August 19, 2015 11 / 60
Service antipatterns in SBSs

Service Component Architecture

SOAP Web services

RESTful APIs

Changed lines of code and code churns

design quality and hinder

A unified abstraction combining different SBSs technologies

Challenge 1

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

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

REST Resource

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

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

Francis PALMA (ÉPM, UQÀM)

Detection of Service Antipatterns

August 19, 2015
Thesis Contributions

Challenge 2
- Only textual descriptions
- No specifications

Service antipatterns in SBSs
- Service Component Architecture
- SOAP Web services
- RESTful APIs

Changed lines of code and code churns
design quality and hinder
Thesis Contributions

Challenge 2: Service DSL
- Only textual descriptions
- No specifications

Service antipatterns in SBSs
- Service Component Architecture
- SOAP Web services
- RESTful APIs

Changed lines of code and code churns
- Service antipatterns hinder design quality and hinder
Service antipatterns in SBSs hinder design quality and changed lines of code and code churns

Challenge 3
- No dedicated unified approach
- No unified framework
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
Challenges:

- No 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

Francis PALMA (ÉPM, UQÀM)
Service antipatterns in SBSs

- Service Component Architecture
- SOAP Web services
- RESTful APIs

Changed lines of code and code churns

Empirical Evidence

- No empirical evidence

Challenge 4
Thesis Contributions

**Service DSL**
- Only textual descriptions
- No specifications

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

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

**Validation of SODA**
- XML-based
- SOAP-based
- WS*-standards

**Changed lines of code and code churns**

**Empirical Evidence**
- No empirical evidence

**A unified abstraction combining different SBSs technologies**

**Challenge 1**
- Only textual descriptions
- No specifications

**Challenge 2**
- No dedicated unified approach
- No unified framework

**Challenge 3**
- No dedicated unified approach
- No unified framework

**Challenge 4**
- No empirical evidence
Service antipatterns in SBSs hinder design quality and change lines of code and code churns.

A unified abstraction combining different SBSs technologies.

Francis PALMA (ÉPM, UQ ÀM)
Unified Abstraction
Meta-abstraction

[Diagram of a class diagram with relationships between Client, Implementation, Interface/PortType, Service, Binding, Resource, Operation, and Message.]
Contributions 2 and 3

Service DSL

Unified SODA Approach
SOFA Framework

Service antipatterns in SBSs

- Service Component Architecture
- SOAP Web services
- RESTful APIs

Changed lines of code and code churns

ECSA '14
Service DSL Unified SODA Approach
SOFA Framework

Francis PALMA (ÉPM, UQÀM)
Detection of Service Antipatterns
August 19, 2015 15 / 60
SODA (Service Oriented Detection for Antipatterns)

Step 1

Textual Description of Service Antipatterns

Specification

Rule Cards
SODA (Service Oriented Detection for Antipatterns)

Specifying Generation

Francis PALMA (ÉPM, UQÂM)
Detection of Service Antipatterns
August 19, 2015 16 / 60
SODA (Service Oriented Detection for Antipatterns)

Textual Description of Service Antipatterns → Step 1: Specification

Rule Cards → Step 2: Generation

Detection Algorithms → Step 3: Detection

Services Involved in Antipatterns

SBSs
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

Service Antipatterns

- LowPerformance
- LowAvailability
- HighResponse
- HighCoupling
- FewMethod
- Bottleneck Service

LowCohesion

- COH VERY_LOW
- COH HIGH

- COMPOS from ONE

- ParentService
- ChildService
- ContainedService

HighDataAccesor

- ANPT HIGH
- ANAM VERY_HIGH

- ASSOC from MANY
- NRO >1

TheKnot

- INTER

- A LOW

- RT VERY_HIGH

- NMD VERY_LOW
- CPL HIGH

MultiService

- INTER

- MultiMethod

- NMD VERY_HIGH
- NMI VERY_HIGH

DataService

- COH HIGH

- HighCohesion

- PrimitiveParameter

- HighDataAccesor

- SmallParameter

- ANP

- ANAM

- NRO LOW

ChattyService

- TotalInvocation

- INTER

- SandPile

- INTER

- LowResponse

- HighAvailability

- HighPerformance

TinyService

- INTER

- Bottleneck Service

- VERY LOW

- VERY HIGH

Francis PALMA (ÉPM, UQÀM) Detection of Service Antipatterns August 19, 2015 18 / 60
BNF Grammar of Rule Cards for SODA

rule_card ::= RULE_CARD:rule_cardName \((rule)^+\);
rule ::= RULE:ruleName \{content_rule\};
content_rule ::= metric | relationship | operator ruleType \((ruleType)^+\)
| RULE_CARD: rule_cardName
ruleType ::= ruleName | rule_cardName
operator ::= INTER | UNION | DIFF | INCL | NEG
metric ::= id_metric ordi_value
| id_metric comparator num_value
id_metric ::= ALS | ANAM/ANAO | ANIM | ANP | ANPT | ARIM | ARIO | ARIP
| COH | CPL | NCO | NI | NIR | NMD/NOD | NOPT
| NOR | NPT | NSE | NUM | NVMS | NVOS | RGTS | TNP
| A | NMI | NTMI | RT
ordi_value ::= VERY HIGH | HIGH | MEDIUM | LOW | VERY LOW
comparator ::= < | ≤ | = | ≥ | >
relationship ::= relationType FROM ruleName cardinality TO ruleName cardinality
relationType ::= ASSOC | COMPOS
cardinality ::= ONE | MANY | ONE_OR_MANY | num_value NUMBER_OR_MANY
rule_cardName, ruleName, ruleClass ∈ string
num_value ∈ double
List of 27 Static and Dynamic Metrics

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

\begin{verbatim}
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’ \not\in ResponseHeader };
11  RULE: NoBodyLink { TLB = 0 };
12 }
\end{verbatim}

*HM = HTTP Method; HL = Hyperlinks; TLB = Total Number of Links in Body;
Detection Algorithms Generation Steps

Step 1: Create Meta-model (.ecore)

Step 2: Write Rules (.rc)

Step 3: Parse and Validate Rules

Step 4: Generation (Detection Algorithms .java)

Textual Description of Service Antipatterns → Specification → Generation → Detection → Services Involved in Antipatterns
SOFA (Service Oriented Framework for Antipatterns)

Detection of Service Antipatterns
August 19, 2015 25 / 60
Validation of SODA Approach

Service antipatterns in SBSs
- Service Component Architecture
- SOAP Web services
- RESTful APIs

Design quality and hinder

Changed lines of code and code churns

SBSs Technologies
<table>
<thead>
<tr>
<th>Research Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can we <strong>efficiently specify</strong> and <strong>detect</strong> service <strong>antipatterns</strong> in different development <strong>technologies</strong> and architectural styles of service-based systems?</td>
</tr>
</tbody>
</table>

**Four** experimental assumptions:
- A1. Generality
- A2. Accuracy
- A3. Extensibility
- A4. Performance
Subject: 31 Service Antipatterns

SCA
- Multi Service
- Tiny Service
- Data Service
- Chatty service
- Service Chain
- Stovepipe Service
- Duplicated Service
- Bottleneck Service

SOAP
- Low Cohesive Operations
- May be It's Not RPC
- Redundant Port-Types

REST
- Ambiguous Name
- Nobody Home
- Bloated Service

CRUDy Interface
- CRUDy URI

- Forgetting Hypermedia
- Ignoring MIME Types
- Breaking Self-descriptiveness
- Ignoring Caching
- Ignoring Status Code
- Misusing Cookies
- Tunelling Through GET
- Tunelling Through POST
- Amorphous URI
- Contextless Resource Names
- Non-heirarchical Nodes
- Pluralised Nodes

Francis PALMA (ÉPM, UQÀM)
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
3. **Apply** detection algorithms automatically
   - On Home-Automation, FraSCAti, and Web services
4. **Manually validate** detection results
5. **Use precision, recall, and F$_1$-measure** as detection accuracy measure
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

Francis PALMA (ÉPM, UQÀM)
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 F₁-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 for
- Tiny Service
- God Component
- Forgetting Hypermedia
## Detection of Tiny Service Antipattern

<table>
<thead>
<tr>
<th>Service Antipatterns</th>
<th>Applicable SBS Technology</th>
<th>Identified Service(s)</th>
<th>Metrics/Occurrences</th>
<th>Detection Time</th>
<th>Precision</th>
<th>Recall</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny Service</td>
<td>SCA (Home-Automation)</td>
<td>MediatorDelegate</td>
<td>NMD=1; CPL=0.44; NOR=4;</td>
<td>0.194s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCA (FraSCAti)</td>
<td>sca-parser</td>
<td>NMD=1; CPL=0.56;</td>
<td>0.067s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web services</td>
<td>SrtmWsPortType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hydro1KWsPortType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ShadowWsPortType</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XigniteTranscripts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BGCanorUSTreasures</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Metrics/Occurrences**: NMD=1; CPL=0.44; NOR=4; NMD=1; CPL=0.56; NOD=2; COH=0.0; NOD=4; COH=0.125; NOD=3; COH=0.083;
- **Detection Time**: 0.194s, 0.067s, 0.945s
- **Precision**: [6/7] 85.71%
- **Recall**: [6/6] 100%
- **F<sub>1</sub>**: 92.31%
Example of Tiny Service Antipattern

```xml
<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

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

```java
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

<table>
<thead>
<tr>
<th>REST APIs</th>
<th>(12) BestBuy</th>
<th>(15) Dropbox</th>
<th>(29) Facebook</th>
<th>(10) Twitter</th>
<th>(9) YouTube</th>
<th>(115) Total</th>
<th>precision-recall</th>
<th>Average Precision-Recall</th>
<th>Detection Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Forgetting Hypermedia</strong></td>
<td>0/0</td>
<td>9/10</td>
<td>8/8</td>
<td>4/4</td>
<td>2/3</td>
<td>36/38</td>
<td>94.58%</td>
<td>100%</td>
<td>19.54s</td>
</tr>
<tr>
<td></td>
<td>0/0</td>
<td>9/9</td>
<td>8/8</td>
<td>4/4</td>
<td>2/2</td>
<td>36/36</td>
<td>pr</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

### Diagram

- **Pattern**
- **No Detection**
- **Antipattern**

### REST Antipatterns

- Alchemy
- BestBuy
- Bitly
- CharlieHarvey
- Dropbox
- Facebook
- Musicgraph
- Ohloh
- TeamViewer
- Twitter
- YouTube
- Zappos

**Detection Time**: 19.54s
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_1gftd5edos6qykgn6t46riezp_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=["ePRUFyBkeQ2ncpP9OLHK80fDw4/KD3Jx-OynnFsIixH9dRdz0sgl-k"],  
    content-length=[324],  
    server=[GSE],  
    alternate-protocol=[443:quic],  
    date=[Fri, 16 May 2014 15:23:34 GMT]
}
```

Body:
```
{  
    "kind": "youtube#videoListResponse",  
    "etag": "\"ePRUFyBkeQ2ncpP9OLHK80fDw4/KD3Jx-OynnFsIixH9dRdz0sgl-k\"",  
    "pageInfo": {  
        "totalResults": 1,  
        "resultsPerPage": 1 },  
    "items": [  
        {  
            "kind": "youtube#video",  
            "etag": "\"ePRUFyBkeQ2ncpP9OLHK80fDw4/9hUt36nrZXNpfqDhYP_bu7W-d3U\"",  
            "id": "SRQtW-sjDGw"
        }
    ]
}
```
Verifying Four Assumptions

Assumptions

A1: Generality
A2: Accuracy
A3: Extensibility
A4: Performance
## A2: Accuracy of Detection Algorithms

<table>
<thead>
<tr>
<th>Antipatterns Groups</th>
<th>Average Precision</th>
<th>Average Recall</th>
<th>Average F-1 measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCA $\cap$ REST $\cap$ Web services</td>
<td>93.33%</td>
<td>100%</td>
<td>96.3%</td>
</tr>
<tr>
<td>SCA $\cap$ Web services</td>
<td>82.59%</td>
<td>95.83%</td>
<td>86.34%</td>
</tr>
<tr>
<td>REST $\cap$ Web services</td>
<td>75%</td>
<td>100%</td>
<td>83.33%</td>
</tr>
<tr>
<td>SCA</td>
<td>88.89%</td>
<td>100%</td>
<td>93.3%</td>
</tr>
<tr>
<td>Web services</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>REST Req/Res Antipatterns</td>
<td>82.81%</td>
<td>90.4%</td>
<td>86.44%</td>
</tr>
<tr>
<td>REST Req/Res Patterns</td>
<td>100%</td>
<td>99.76%</td>
<td>99.88%</td>
</tr>
<tr>
<td>REST Linguistic Antipatterns</td>
<td>81.4%</td>
<td>78%</td>
<td>79.66%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>88%</strong></td>
<td><strong>95.5%</strong></td>
<td><strong>90.66%</strong></td>
</tr>
</tbody>
</table>
A3: Extensibility of SOFA Framework

Detection Algorithm Generation

SOFA Framework

Operator

Rule Specification

Rule

Metric

Boxplot

SCA Handler

Web Service Handler

REST Handler

OWL FraSCAti
## A4: Performance of Detection Algorithms

<table>
<thead>
<tr>
<th>Antipatterns Groups</th>
<th>Average Detection Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCA ∩ REST ∩ Web services</td>
<td>0.511s</td>
</tr>
<tr>
<td>SCA ∩ Web services</td>
<td>9.09s</td>
</tr>
<tr>
<td>REST ∩ Web services</td>
<td>18.98s</td>
</tr>
<tr>
<td>SCA</td>
<td>0.184s</td>
</tr>
<tr>
<td>Web services</td>
<td>144.72s</td>
</tr>
<tr>
<td>REST Syntactic Antipatterns</td>
<td>22.06s</td>
</tr>
<tr>
<td>REST Syntactic Patterns</td>
<td>20.44s</td>
</tr>
<tr>
<td>REST Linguistic Antipatterns</td>
<td>0.70s</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>27.09s</strong></td>
</tr>
</tbody>
</table>
Verifying Four Assumptions

- A1: Generality
- A2: Accuracy
- A3: Extensibility
- A4: Performance
**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.
Service antipatterns in SBSs hinder design quality and change

- Service Component Architecture
- SOAP Web services
- RESTful APIs

Empirical Evidence

Changed lines of code and code churns
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?

- **RQ2.1** - What is the relation between service antipatterns and change-proneness?
- **RQ2.2** - What is the relation between particular kinds of service antipatterns and change-proneness?
## Study Subjects: Service Antipatterns

<table>
<thead>
<tr>
<th>Antipatterns</th>
<th>Names</th>
<th>Detected Instances</th>
<th>Involved Java Source Files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bloated Service</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Bottleneck Service</td>
<td>2</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>God Component</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Multi Service</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Nobody Home</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Service Chain</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>The Knot</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Tiny Service</td>
<td>1</td>
<td>24</td>
</tr>
</tbody>
</table>
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

Francis PALMA (ÉPM, UQÀM)  Detection of Service Antipatterns  August 19, 2015  49 / 60
Study Approach

FraSCAti version control system

invoke and execute FraSCAti services

Service Antipatterns Detection

Server-side artefacts analysis

Client-side artefacts analysis

1 Import Mailing Data

2 Source Changes Extractor

Code Churns Extractor

code churns

number of changes per service

3 Service Antipatterns Detection

FraSCAti services involved in antipatterns

4 Data Preparation

Francis PALMA (ÉPM, UQÀM)
Study Variables

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

**Dependent variables**
- Change-proneness of services’ source files
  - Total number of changes as $c_i$
  - Total number of code churns as $d_i$
RQ₂.1: Antipatterns are More Change-prone

<table>
<thead>
<tr>
<th>Treatment Groups</th>
<th>Treatment Types</th>
<th>p-value</th>
<th>Cliff’s δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>antipatterns ~ non-antipatterns</td>
<td>total number of code churns</td>
<td>0.015</td>
<td>0.515 (large)</td>
</tr>
<tr>
<td>antipatterns ~ non-antipatterns</td>
<td>total number of changes</td>
<td>0.011</td>
<td>0.496 (large)</td>
</tr>
</tbody>
</table>
RQ2.2: Antipatterns are Not Equally Change-prone

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

<table>
<thead>
<tr>
<th>Treatment Groups</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>total number of code churns ~ antipattern</td>
<td>0.0002</td>
</tr>
<tr>
<td>total number of changes ~ antipattern</td>
<td>0.01003</td>
</tr>
</tbody>
</table>
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 1
- Only textual descriptions
- No specifications

Service antipatterns in SBSs

Challenge 2
- Only textual descriptions
- No specifications

Challenge 3
- No dedicated unified approach
- No unified framework

Challenge 4
- No empirical evidence

No unified abstraction

Service Component Architecture

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

SOAP Web services

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

RESTful APIs

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

Changed lines of code and code churns

Francis PALMA (ÉPM, UQÀM)
Conclusion: Our Contributions

- Service DSL
  - Only textual descriptions
  - No specifications

- Challenge 2
  - Service antipatterns in SBSs

- Challenge 3
  - Unified SODA Approach
  - SOFA Framework
  - Validation of SODA

- Challenge 4
  - Empirical Evidence
  - No empirical evidence

A unified abstraction combining different SBSs technologies

Service Component Architecture
SOAP Web services
RESTful APIs

Service antipatterns hinder design quality and changed lines of code and code churns

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

Francis PALMA (ÉPM, UQÀM)
Detection of Service Antipatterns
August 19, 2015 56 / 60
Research Question 1

Can we efficiently specify and detect service antipatterns in different development technologies and architectural styles of service-based systems?
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
Proposing a unified approach to specify and detect service antipatterns by assessing the quality of design and quality of service of SBSs.