Specification and Detection of SOA Antipatterns

Francis Palma
Ptidej Team, DGIGL, École Polytechnique de Montréal, Canada
Email: francis.palma@polymtl.ca

Service-oriented architecture (SOA) provides a collection of principles and methodologies for designing and developing service-based systems (SBSs). SBSs are composed of loosely-coupled, platform independent, and reusable functional units, *i.e.*, *services*. Alternative technologies to implement SBSs are REST-style (REpresentational State Transfer), Service Component Architecture (SCA), SOAP-based Web service, and so on. However, SBSs cannot overcome some common software engineering challenges, *e.g.*, evolution, to fit new user requirements or changes in execution contexts. All these changes may degrade the quality of design and quality of service of SBSs and may cause the presence of common bad practiced solutions—*antipatterns*.

Identified Research Problems:
1. *Impact of SOA antipatterns is not yet verified in SBSs;*
2. *No specification of SOA antipatterns;*
3. *No dedicated approach and framework for their detection;*

Research Questions: *RQ1.* Do service-oriented antipatterns differ from the object-oriented antipatterns? *RQ2.* Do the service-oriented antipatterns vary in diverse SBSs developed using different SOA technologies? *RQ3.* Can we efficiently specify and detect SOA antipatterns regardless of SOA technologies? *RQ4.* How do the SOA antipatterns impact the maintenance and evolution of SBSs?

Hypotheses: *H1. Generality:* Our domain specific language (DSL) allows the specification of different SOA antipatterns, from simple to more complex ones. *H2. Accuracy:* Antipatterns detection algorithms have a precision of 75%, and a recall of 100%, *i.e.*, more than three-quarters of detected antipatterns are true positives and we do not miss any existing antipatterns. *H3. Extensibility:* Our DSL and SOFA framework are extensible for adding new metrics and antipatterns. *H4. Performance:* The required time for the antipatterns detection is considerably low, *i.e.*, in the order of seconds.

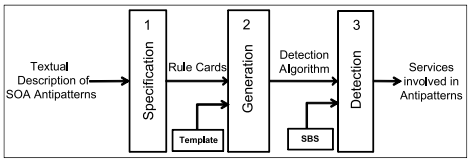

Figure 1. Proposed SODA approach.

AntipatternName	ServicesInvolved	Metrics

Tiny Service	`MediatorDelegate`	`NOR=4`
		`CPL=0.440`
		`NMD=1`

Multi Service	`IMediator`	`COH=0.027`
		`NMD=13`
		`RT=132ms`

Subjects: (1) Eight REST antipatterns: breaking self-descriptiveness, forgetting hypermedia, ignoring caching, ignoring status code, and so on; (2) Ten Web service antipatterns: ambiguous name, chatty web service, fine grained web service, god object web service, redundant port-types, and so on; (3) SCA antipatterns: sand pile, the knot, nobody home, bottleneck service, service chain, bloated service, and so on.

```
1 RULE_CARD: MultiService {
2  RULE: MultiService {INTER MultiMethod HighResponse LowA LowCohesion};
3  RULE: MultiMethod {NMD VERY_HIGH};
4  RULE: HighResponse {RT VERY_HIGH};
5  RULE: LowA {A LOW};
6  RULE: LowCohesion {COH LOW};
7 };
```

```
1 RULE_CARD: TinyService {
2  RULE: TinyService {INTER FewMethod HighCoupling};
3  RULE: FewMethod {NMD VERY_LOW};
4  RULE: HighCoupling {CPL HIGH};
5 };
```

Figure 2. Rule cards for two SCA antipatterns.

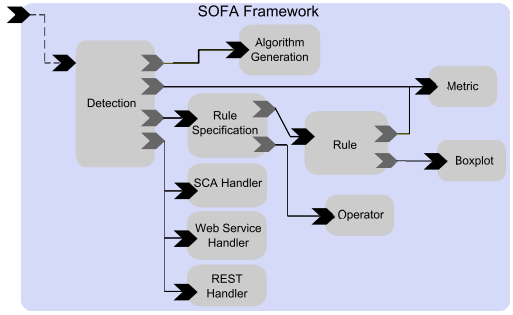

Figure 3. The SOFA framework.

Antipatterns	Involved Web Services	Metrics	Boxplot Values Min|Median|Max
Ambiguous Name	`AIP3_PV_Impact-Callback`	ALS 0.675 RGTS 0.85 NVMS 26 NVOS 7	0.027|0.463|0.675 0.0|0.0|0.85 4|6|54 1|3|20
Fine Grained Web Service	`SrtmWsPortType`	NOD 2 COH 0.0	2|5.5|27 0.0|0.216|0.443
	`Hydro1KWsPortType`	NOD 2 COH 0.0	*same as above*
	`ShadowWsPortType`	NOD 2 COH 0.0	*same as above*

Objects: (1) RESTful APIs: Facebook, BestBuy, Twitter, DropBox, YouTube, and so on; (2) SOAP-based Web services: 122 weather- and finance-related Web services from programmableweb.com; (3) SCA Systems: Two SCA systems *Home-Automation* and *FraSCAti*.

Results: (1) ICSOC 2012: Detection of ten SCA antipatterns; (2) IJCIS 2013: three more antipatterns from ICSOC 2012 and extensive validation with FraSCAti; (3) ECSA 2014: Detection of ten Web service antipatterns; and (4) ICSOC 2014: Detection of eight REST antipatterns.