BPEL4PEOPLE Anti-Patterns: Discovering Authorization Constraint Anti-Patterns in Web Services

Henrique J. A. Holanda[1], **Carla K. de M. Marques**[2], **Francisca Aparecida P. Pinto**[3], **Yann-Gaël Guéhéneuc**[4]

[1] Department of Computer, State University of Rio Grande do Norte
Mossoró, RN, Brazil, 59.610-210

{henriqueholanda@uern.com}

[2]Federal Institute of Rio Grande do Norte
Mossoró, RN, Brazil,

{carla.marques@ifrn.edu.br }

[3]Department of Computer, Federal Rural University of the Semi-Arid
and State University of Rio Grande do Norte
Mossoró, RN, Brazil, 59625-900

{aparecidapradop@gmail.com}

[4]Department of Computer Engineering, University of Montreal,
École Polytechnique de Montréal

{yann-gael.gueheneuc@polymtl.ca}

Abstract. *Despite the abundance of analysis techniques to discover anti-patterns in BPEL, there is hardly any support for authorization constraint errors in web services orchestrated by BPEL4People. Most techniques simply abstract from people (human user interactions), while people dependencies can be the source of all kinds of errors. This paper focuses on the discovery authorization constraint anti-patterns in web services orchestrated by BPEL4People. We present an analysis approach that is expressed in terms of rule card, the well-known, stable, adaptable, and effective model-checking techniques can be used to discover authorization constraint errors. Moreover, our approach enables a seamless integration of control-flow and authorization constraint verification.*

1. Introduction

A BPEL4People is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models. BPEL4People systems are driven by explicit process models, i.e., based on a process model, a system is configured that supports the modeled process where exist user interactions. In this paper, we primarily focus on the analysis of the models used to configure web services orchestrated by BPEL4People. However, our approach is also applicable to other software system that manages and executes operational processes involving people. In the last 15 years, many analysis techniques have been developed to analyze web services orchestrated by BPEL4People. Most analysis techniques focus

on verification, i.e., the discovery of design errors. Although many process representations have been used or proposed, most researchers are using Petri Nets as a basic model [H.J.A Holanda and Serra 2010]. The flow-oriented nature of BPEL4People processes makes the Petri Net formalism a natural candidate for the modeling and analysis of work flows.

Unfortunately, lion's share of attention has been devoted to control-flow while ignoring other perspectives such as data-flow and resource allocation. Analysis techniques typically check for errors such as deadlocks, live locks, etc. while abstracting from data and uses. Existing approaches typically suffer from the following two problems: i) they look at only one perspective in isolation (e.g., only control-flow); and ii) the types of errors they capture are usually not configurable and mainly driven by the verification algorithms themselves rather than by user requirements [Dumas et al. 2005].

To address some of the limitations of existing approaches, we propose a new analysis framework based on uses of "anti-patterns" expressed in terms of rule card. Assuming a rule card representation, we define anti-patterns related to the BPEL4People. The term "anti-patterns" was coined in 1995 by Andrew Koenig. He stated that "An anti-pattern is just like pattern, except that instead of solution it gives something that looks superficially like a solution but isn't one" [Moha et al. 2012]. The goal of anti-patterns is to formally describe repeated mistakes such that they can be recognized and repaired. In this paper, we use rule card to formalize our anti-patterns. This formalization can be used to discover the occurrence of such anti-patterns in BPEL4People. Although not elaborated on in this paper, the same techniques can be used to define correctness notions related to the control-flow and check these in an integral way.

An example of an anti-pattern is Dangling Inputs Group of Users (DIGU). This anti-pattern describes the situation where some group of users' needs to be attributed for one task, but either it has never been created. The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 introduces BPEL and BPEL4People. Section 3.2 introduces anti-patter. Section 5 presents the specification of authorization Constrain anti-pattern in web services orchestrated with BPEL4People. Section 4 presents our proposed approach. Section 5.1 presents the experiments and results. Section 6 concludes the paper and gives suggestions of future works.

2. Related Work

In the past years, several catalogs to specify and detect anti-patterns of SOA [Moha et al. 2012], [Palma et al. 2014b], Rest [Palma et al. 2014a] and BPEL [Palma et al. 2013] services and languages have been proposed. In [Smith and Williams 2003] the authors detected and evaluated more new software antipatterns and in [Sinnig et al. 2005] the authors detected and evaluated patterns in model-based engineering. We can highlight the drawbacks of the current literature as follows:

- Anti-patterns and approaches to detect them were considered only for SOA [Palma et al. 2014b] , [Palma et al. 2014a] and [Palma et al. 2013] models;
- Extending BPEL Engines with BPEL4People [Holmes et al. 2008];
- A model checking approach to verify BPEL4WS workflows [Bianculli et al. 2007];

- Web Services Human Task (WS-HumanTask) [Ings et al. 2012];
- Access control and authorization constraints for WS-BPEL [Bertino et al. 2006];
- Verifying BPEL workflows under authorization constraints [Zha 2006];

Finally, there is no detection approach for detecting authorization constraint anti-patterns in web services orchestrated by BPEL4People anti-patterns until now, so we focus on those issues with a solution to propose a concrete approach for specifying and detecting authorization constraint anti-patterns in web services orchestrated by BPEL4People.

3. BPEL4People and Anti-Patterns

Web Services Business Processes Execution Language focuses on business processes that orchestrate Web service interactions. However, in general, business processes are comprised of a broad spectrum of activities that most often require the participation of people to perform tasks, review or approve steps and enter data — for example, a credit approval scenario that may require approval on certain transaction limits or activity levels. These human interactions are now addressed in the new specifications. Human user interactions are currently not covered by the WS-BPEL, which is primarily designed to support automated business processes based on WS. In practice, however, many business process scenarios require user interaction. So far, we've seen that user interaction in business processes can get quite complex. Although BPEL specification 1.1 (and the upcoming BPEL 2.0) doesn't specifically cover user interactions, BPEL is appropriate for human work flows. Work flow services that leverage the rich BPEL support for asynchronous services are created today. In this fashion, people and manual tasks become just another asynchronous service from the perspective of the orchestrating process and the BPEL processes stay 100% standard.

3.1. BPEL4People

We now see the next generation of work flow specifications emerging around BPEL with the objective of standardizing the explicit inclusion of human tasks in BPEL processes. This proposal is called BPEL4People and was originally put forth by IBM and SAP in July 2005. Other companies, such as Oracle, have also indicated that they intend to participate in and support this effort.

The most important extensions introduced in BPEL4People are people activities and people links. People activity is a new BPEL activity used to define user interactions; in other words, tasks that a user has to perform. For each people activity, the BPEL server must create work items and distribute them to users eligible to execute them. People activities can have input and output variables and can specify deadlines. To specify the implementation of people activities, BPEL4People introduced tasks. Tasks specify actions that users must perform. Tasks can have descriptions, priorities, deadlines, and other properties. To represent tasks to users, we need a client application that provides a user interface and interacts with tasks: it can query available tasks, claim and revoke them, and complete or fail them.

To associate people activities and the related tasks with users or groups of users, BPEL4People introduced people links. People links are somewhat similar to partner links; they associate users with one or more people activities. People links are usually associated with generic human roles, such as process initiator, process stakeholders, owners, and

administrators [Ings et al. 2012]. BPEL4People extends the capabilities of WS-BPEL to support a broad range of human interaction patterns, allowing for expanded modeling of business processes within the WS-BPEL language. BPEL4People is comprised of two specifications including: i) WS-BPEL Extension for People which layers features on top of WS-BPEL to describe human tasks as activities that may be incorporated as first-class components in WS-BPEL process definitions; ii) Web Services Human Task introduces the definition of stand-alone human tasks, including the properties, behavior and operations used to manipulate them. Capabilities provided by Web Services Human Task may be utilized by Web services-based applications beyond WS-BPEL processes.

3.1.1. Integrating Authorization Constraints

BPEL4People support features to exclude some users from performing a task because of some tasks they had done before or force some user to perform a sequence of tasks. We call such requirement as authorization constraint, as the term is widely used in access control literature. In this section we will use GSPN to express the authorization constraints to facilitate formal analysis. Two kinds of authorization constraints, namely "4-eyes principle" and "chained execution", are proposed in BPEL4People specification. The "4-eyes principle", also known as "separation of duty", is a common scenario in many application areas when a decision must be made by two or more people independently of one another, often for the security reasons, and "chained execution" refers a process fragment where a sequence of steps must be executed by one person.

3.1.2. Separation of duty

The separation of duty (SoD) is a well-known principle in authorization to prevent fraud or error by requiring that at least two individuals are involved in some specific work. SoD is also useful when two persons have to co-operate in a work but none of them should know all the details. The basic form of SoD states that two given distinct tasks t1 and t2 must be performed by different individuals. This can be defined as states that person p0 cannot perform both t1 and t2. We can define variations of this similarly, e.g., "task t1 and t2 must be performed by different roles". We can also define SoD constraint for a specific person, e.g., "person A cannot invoke both task t1 and t2".

3.1.3. Binding of duty

"Binding of duty" (BoD) is the dual of SoD, which states that some distinct tasks must be performed by one person. BoD is used to define the responsibility of a person, e.g.: It states that if p0 performs t1, then p0 must also perform t2, and vice versa. SoD and BoD may be combined to define more complex constraints.

3.2. Anti-Patterns

Changes resulting from the evolution of orchestrated with BPEL4People can degrade its design of web services and can often cause the appearance of poor solutions in the architecture: anti-patterns.

Patterns and anti-patterns exist to capture expertise, and to communicate knowledge. Anti-patterns are opposed to patterns which are good specifications (solutions) for recurring problems. An anti-pattern is a surface-level symptom that hints at the presence of a deeper, more serious problem. Anti-patterns could hinder [Palma et al. 2013] future maintenance and evolution of web services.

Anti-patterns detection is therefore important to assess the design of web services and ease their maintenance and evolution. However, methods and techniques for detection authorization constraint anti-patterns in web services orchestrated by BPEL4People do not yet exist. Anti-patterns have many definitions: they are ”poor” solutions to recurring design problems, known solutions for solving problems, which are not practical and usable, and wrong practices that are quite commonly used. Anti-patterns have symptoms and consequences and are induced by some root clauses.

The presence of anti-patterns must be identified in unsuccessful system and their absence shown in successful systems. Anti-patterns usually cause costly and complicated architectures, which are costly and difficult to maintain. Having anti-patterns in a system can hinder a project. Developers study anti-patterns to avoid pitfalls. But sometimes can create pitfalls in knowledge transfer if not applied appropriately. The idea that the use of anti-patterns in knowledge transfer may be a dangerous strategy if applied incorrectly than discovery and corrected before.

4. Specification of Authorization Constrain Anti-pattern in Web Services Orchestrated with BPEL4People

In this section we introduce the specification of 7 (seven) authorization constrain anti-pattern. These anti-patterns were adapted from the literature [Smith and Williams 2000], [Moha et al. 2012], [Palma et al. 2014b], [Palma et al. 2014a], [Palma et al. 2013] and [Zha 2006].

- Dangling Inputs Group of Users (DlGU): DIGU is an anti-pattern where one group of users remain unused.
- Dangling Outputs Group of Users (DOGU): DOGU is an anti-pattern where one group of users is assigned for one service but it was not defined.
- Duplicated Group of Users (DGU): DGU corresponds to a group of highly similar users.
- Ambiguous Name of groups of users (ANGU): Ambiguous Name is an anti-pattern where the developers use the names of groups that are very short or long, include too general terms, or even show the improper use of verbs, etc. Ambiguous names are not semantically and syntactically sound and impact the discover ability and the re-usability of a Web service.
- Missing User (MU): When a user that performs a task is excluded to perform another task that he/she is not one user potential of this another task.
- Strongly Missing Groups of Users (SMGU): When who performs a task was the members of one group and another task excludes the members of this group, however no member of this group is a potential user of this another task.
- Weakly Missing Groups of Users (WMGU): When who performs a task was the members of one group and another task excludes the members of this group though some member of this group is a potential user of this another task and others not.

5. Approach

With the aim of detecting anti-patterns in authorization constrain in web services orchestrated with BPEL4People, we used the approach shown in 1. This approach involves three major steps:

- Step 1 (Specification) concerns specifying rules for the detection of authorization constrain anti-patterns that, later on, will be applied on web services orchestrated with BPEL4People.
- Step 2 (Generation) transforms orchestrated with BPEL4People into an intermediary representation, i.e., more abstract and simplified, by filtering some process facts those are not required to apply rules to ease: (i) the implementation of the rules defined in the previous step and (ii) the further analysis of the processes.
- Step 3 (Detection) consists in applying the rules defined in Step 1 on the transformed processes in the previous step. Finally, a list of existing anti-patterns with the involved process fragments will be shown.

Static service antipatterns require only static analysis for their detection and, thus, only their structural properties are needed to detect them within the services. Coupled with the information about how these parts will be distributed across the web services orchestrated with BPEL4People, we use rules to detect the potential anti-patterns that may occur. Our approach for detecting anti-patterns leverages static analysis to detecting anti-patterns in authorization constrain in web services orchestrated with BPEL4People.

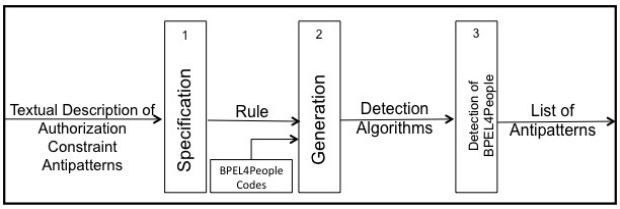


Figure 1. Proposed Detection Approach.

5.1. Experiments and Results

This section applies the proposed detection approach for seven authorization constraint anti-patterns in Web services orchestrated by BPEL4People anti-patterns, specified in Sections 4.

- Step 1 concerns specifying rules for the detection of authorization constrain anti-patterns that, later on, will be applied on Web services orchestrated with BPEL4People.
- Step 2 transforms orchestrated BPEL4People code into an intermediary representation, i.e., an more abstract and simplified code, by filtering some processes, those are not required to apply rules. This step consists also in implementing the rules showed in 2. We implement the rules in a modularized way, i.e., we implement each side of different logical operators (e.g., AND, OR) in a rule and join them afterwards to check the conformance with the defined conditions using the approach in [8].
- Finally, Step 3 applies the rules defined in Step 1 on the transformed processes in the previous step. The detection step follows the specification of the rules and the

transformation of BPEL4People codes into an intermediary representation, i.e., more abstract and simplified files XML. We show with a small scale experiment the effectiveness of our proposed approach.

We specify seven anti-patterns and we analyze and perform anti-pattern detection of these seven anti-pattern for three Web services orchestrated by BPEL4People using our approach.

- Employee-of-month: this Web services is started and as a first step, the people are determined that qualify as voters for the Employee of the month. Next, all the voters identified before get a chance to cast their votes. After that, the election result is determined by counting the votes casted. After the result is clear, two different people from the set of people entitled to approve the election either accept or reject the voting result. In case any of the two rejects, then there is no Employee of the month elected in the given month, and the process ends. In case all approvals are obtained successfully, the employees are notified about the outcome of the election, and a to-do is created for the elected Employee of the month to prepare an inaugural speech. Once this is completed, the process completes successfully;
- WS-PurchSys: this is a BPEL4People source code for a purchasing process, it is showed in Algorithm 1. Four tasks are defined: manager approve, finance approve, notify staff, and purchase. The potential owners of each task are: manager approve (Alan); finance approve (Ben); purchase (Ben, Cindy, Diana); notify staff (Diana, Edward). The excluded owner of purchase is the actual owner of finance approve. The excluded owner of notify staff is the actual owner of purchase; and
- Web-Service1: this Web services consists of two groups of users (”Group 1” and ”Group 2”) and five services (”Service 1”, ”Service 2”, ”Service 3” (”Service 3” excludes who performed the ”Service 2”), ”Service 4” (”Service 4” excludes who performed the ”Service 1”) and ”Service 5”).

We could detect the seven authorization constrain antipatterns specified in Section 4 in the three Web services orchestrated with BPEL4People given as an examples.

We present in Table 1 the anti-patterns detection results for the three Web services orchestrated with BPEL4People followed by some discussion about accuracy and performance detection. Initially some anti-patterns were not detected in these Web services orchestrated with BPEL4People. Subsequently some changes were inserted in these codes to test the accuracy and performance of the detection algorithm then anti-patterns were detected.

Through our experiment, we aim to show the accuracy and performance of the detection algorithms in terms of precision, recall and detection time, presented in Table 1. Antipattern detection algorithms have at least a precision of 99%. Assuming that the anti-patterns have a negative impact on the design, we target a recall of 100% for anti-patterns, which ensures that we do not miss any existing anti-patterns. The precision concerns the detection accuracy of our specified rules and the corresponding detection algorithms.

The primary threat to the validity concerns the external validity of our results, i.e., generalizing the proposed approach to other Web services orchestrated with BPEL4People. We perform specification and detection of seven authorization constrain anti-patterns. However, we ran the experiment on a set of three Web services orchestrated

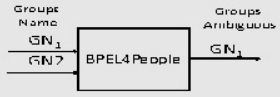
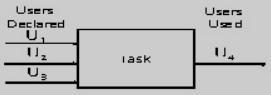
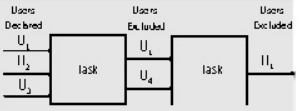
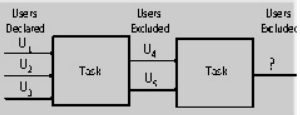
Anti-patterns	Rule	Diagrams
Dangling Inputs Group of Users (DIGU)	RULE_CARD DanglingInputsGroupUsers { RULE InputsGroupUsersUnused {DIGU ≥ 0} IF{DIGU≥ 0} THEN DanglingInputsGroupUsers}	
Dangling Outputs Group of Users (DOGU)	RULE_CARD DanglingOutputsGroupUsers { RULE OutputsGroupUsersUnused {DOGU ≥ 0} IF{DOGU ≥ 0} THEN DanglingOutputsGroupUsers}	
Dangling Group of Users (DGU)	RULE_CARD DanglingGroupUsers { RULE GroupUsersUnused {DGU ≥ 0} IF{DGU ≥ 0} THEN Dangling GroupUsers }	
Ambiguous Name of groups of users (ANGU)	RULE_CARD AmbiguousNameGroupsUsers { RULE AmbiguousName {ANGU ≥ 0} IF{ANGU ≥ 0} THEN AmbiguousNameGroupsUsers}	
Missing User (MU)	RULE_CARD DanglingOutputResource { RULE OutputResourceUnused {MU ≥ 0} IF{MU ≥ 0} THEN DanglingOutputResource}	
Strongly Missing Groups of Users (SMGU)	RULE_CARD StronglyMissingGroupsUsers { RULE StronglyMissing {SMGU ≥ 0} IF{SMGU ≥ 0} THEN StronglyMissingGroupsUsers}	
Weakly Missing Groups of Users (WMGU)	RULE_CARD WeaklyMissingGroupsofUsers { RULE WeaklyMissing {WMGU ≥ 0} IF{WMGU ≥ 0} THEN WeaklyMissingGroupsofUsers}	

Figure 2. Rules specifications and diagrams.

with BPEL4People to minimize this threat. We plan to replicate the experiment on others Web services orchestrated with BPEL4People as a future work. We do not execute the Web services orchestrated with BPEL4People, hence analyze the processes statically, and the injections were performed internally, which are threats to the internal validity. However, we plan to execute the web services orchestrated with BPEL4People flows to dynamically analyze them in the future.

The subjective nature to define rule cards is a threat to construct validity. However, we lessen this threat by defining the rule cards after a thorough literature review. Finally, the conclusion validity threats refer to the relation between the treatment and the out come. We paid full attention not to violate the assumptions of the performed statistical tests. We mainly used non-parametric tests that do not require to make any presumption about the data distribution. The threats to internal validity concern the possibility of replicating this study. To minimize this threat, we provide all the details required to replicate the study, including the source code repositories and the raw data used to compute the statistics on our web services. One major challenge to minimize the threat to the external validity is the very limited availability of open-source Web services orchestrated with BPEL4People.

Finally, a list of anti-patterns, among the seven specified, existing in three Web services orchestrated by BPEL4People is showed in Table 1.

6. Conclusions and Future Work

In this paper we assessed the authorization constrain in Web services orchestrated with BPEL4People and eased the maintenance and evolution of these architectures.

Table 1. Experiments results.

ANTIPATTERN	METRIC	PRECI-SION	RECALL	DETECT. TIME (s)
Dangling Inputs Group of Users (DIGU)	NDIGU = 0	97%	100%	3,34
Dangling Outputs Group of Users (DOGU)	NDOGU = 0	98%	100%	3,04
Dangling Group of Users (DGU)	NDGU = 0	100%	100%	3,53
Ambiguous Name of groups of users (ANGU)	NANGU = 0	98%	100%	3,64
Missing User (MU)	NOU = 0	98%	100%	3,48
Strongly Missing Groups of Users (SMGU)	NSMGU = 0	98%	100%	3,21
Weakly Missing Groups of Users (WMGU)	NWMGU = 0	100%	100%	3,07

We proposed an approach to specify authorization constrain anti-patterns and detect them in Web services orchestrated with BPEL4People. We present our static analysis-based approach to detect antipatterns in BPEL4People codes. We have also proposed to assess the design and QoS of Web services orchestrated with BPEL4People. To achieve this goal, we propose an approach to specify authorization constrain anti-patterns and detect them in Web services orchestrated with BPEL4People. Finally, we want to quantify the impact of detection anti-patterns on the maintenance and evolution of Web services orchestrated with BPEL4People.

The contribution of this paper is specifying and detect authorization constrain anti-patterns in Web services orchestrated with BPEL4People, by leveraging static code analysis and information about prospective deployment of the application components using Web services orchestrated with BPEL4People. We applied and validated the detection algorithms using three Web services orchestrated by BPEL4People showed in Section 5.1, in terms of precision and recall. We specified and detected seven authorizations constrain antipatterns in an initial experiment with tree Web services orchestrated by BPEL4People. Results showed that this approach has an average detection precision of more than 97% and recall of 100%. Hence, we make a strong recommendation that designers be exposed to antipatterns repeatedly until the pattern is well established. Only then should anti-patterns be used to strengthen the pattern in the designer's mind.

7. Acknowledgements

We would like to thank the CNPQ, CAPES, UERN, UFERSA, IFRN, SEDUC-CE and Canada Chair.

References

(2006). *Verifying BPEL Workflows Under Authorisation Constraints*, volume 4102 of *Lecture Notes in Computer Science*. Springer.

Bertino, E., Crampton, J., and Paci, F. (2006). Access control and authorization constraints for ws-bpel. In *Web Services, 2006. ICWS '06. International Conference on*, pages 275–284.

Bianculli, D., Ghezzi, C., and Spoletini, P. (2007). A model checking approach to verify bpel4ws workflows. In *Service-Oriented Computing and Applications, 2007. SOCA '07. IEEE International Conference on*, pages 13–20.

Dumas, M., van der Aalst, W. M., and ter Hofstede, A. H. (2005). *Process-aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, Inc., New York, NY, USA.

H.J.A Holanda, J. Merseguer, G. C. and Serra, A. B. (2010). Performance evaluation of web services orchestrated with ws-bpel4people. In *International Journal of Computer Networks & Communications*, volume 2, pages 117–134. AIRCC Publishing Corporation.

Holmes, T., Vasko, M., and Dustdar, S. (2008). Viebop: Extending bpel engines with bpel4people. In *PDP*, pages 547–555. IEEE Computer Society.

Ings, D., Clément, L., König, D., Mehta, V., Mueller, R., Rangaswamy, R., Rowley, M., and Trickovic, I. (2012). Web services human task (ws-humantask) specification version 1.1. OASIS Committee Specification Draft 12 / Public Review Draft 05.

Moha, N., Palma, F., Nayrolles, M., Conseil, B., Guéhéneuc, Y.-G., Baudry, B., and Jézéquel, J.-M. (2012). Specification and Detection of SOA Antipatterns. In Liu, C., Ludwig, H., Toumani, F., and Yu, Q., editors, *Service-Oriented Computing*, volume 7636 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg.

Palma, F., Dubois, J., Moha, N., and Guéhéneuc, Y. (2014a). Detection of REST patterns and antipatterns: A heuristics-based approach. In *Service-Oriented Computing - 12th International Conference, ICSOC 2014, Paris, France, November 3-6, 2014. Proceedings*, volume 8831 of *Lecture Notes in Computer Science*, pages 230–244. Springer.

Palma, F., Moha, N., and Gueheneuc, Y.-G. (2013). Detection of process antipatterns: A bpel perspective. In *Enterprise Distributed Object Computing Conference Workshops (EDOCW), 2013 17th IEEE International*, pages 173–177.

Palma, F., Moha, N., Tremblay, G., and Guéhéneuc, Y. (2014b). Specification and detection of SOA antipatterns in web services. In *Software Architecture - 8th European Conference, ECSA 2014, Vienna, Austria, August 25-29, 2014. Proceedings*, pages 58–73.

Sinnig, D., Gaffar, A., Reichart, D., Forbrig, P., and Seffah, A. (2005). Patterns in model-based engineering. In Jacob, R., Limbourg, Q., and Vanderdonckt, J., editors, *Computer-Aided Design of User Interfaces IV*, pages 197–210. Springer Netherlands.

Smith, C. U. and Williams, L. G. (2000). Software performance antipatterns. In *Proceedings of the 2Nd International Workshop on Software and Performance*, WOSP '00, pages 127–136, New York, NY, USA. ACM.

Smith, C. U. and Williams, L. G. (2003). More new software antipatterns: Even more ways to shoot yourself in the foot. In *29th International Computer Measurement Group Conference, December 7-12, 2003, Dallas, Texas, USA, Proceedings*, pages 717–725. Computer Measurement Group.