The Journal of Systems & Software 204 (2023) 111767

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

BPEL process defects prediction using multi-objective evolutionary search ☆

Marwa Daaji [a,b], Ali Ouni [b,*], Mohamed Mohsen Gammoudi [c], Salah Bouktif [d], Mohamed Wiem Mkaouer [e]

[a] Faculty of Sciences, University of Tunis El Manar, Tunisia
[b] ETS Montreal, University of Quebec, Montreal, QC, Canada
[c] Higher Institute of Multimedia Art, University of Manouba, Tunisia
[d] Department of Computer Science and Software Engineering, CIT, UAE University, United Arab Emirates
[e] Rochester Institute of Technology, Rochester, NY, USA

ARTICLE INFO

Article history:
Received 21 September 2022
Received in revised form 15 May 2023
Accepted 24 May 2023
Available online 1 June 2023

Keywords:
BPEL process
Anti-patterns
Multi-objective algorithms
Genetic programming

ABSTRACT

Web services are becoming increasingly popular technologies for modern organizations to improve their cooperation and collaboration through building new software systems by composing pre-built services. Such services are typically composed and executed through BPEL (Business Process Execution Language) processes. Like any other software artifact, such processes are frequently changed to add new or modify existing functionalities or adapt to environmental changes. However, poorly planned changes may introduce BPEL process design defects known as *anti-patterns* or *defects*. The presence of defects often leads to a regression in software quality. In this paper, we introduce an automated approach to predict the presence of defects in BPEL code using Multi-Objective Genetic Programming (MOGP). Our approach consists of learning from real-world instances of each service-based business process defect (*i.e.*, anti-pattern) type to infer prediction rules based on the combinations of process metrics and their associated threshold values. We evaluate our approach based on a dataset of 178 real-world business processes that belong to various application domains, and a variety of BPEL process defect types such as data flow and portability defects. The statistical analysis of the achieved results shows the effectiveness of our approach in identifying defects compared with state-of-the-art techniques with a median accuracy of 91%.

1. Introduction

Modern software systems heavily rely on the use of Web services which are typically composed and orchestrated through business processes. The global market for business process as a service is estimated at US$45.9 Billion at the end of 2020 and it is continuously growing to reach US$68.7 Billion by 2027, as announced by the *ReportLinker*.[1] Modern companies can quickly develop their business by composing new applications using pre-built components while minimizing capital investment, time to market, and risk (Riemann, 2016; Rosinosky et al., 2016). In this competitive environment, organizations have immense pressure not only to keep up with the speed of the business but also to continuously evolve, innovate and re-invent their Service-based Business Processes (SBPs) to succeed (Song et al., 2021).

Like any software, SBPs are susceptible to evolution by adding new functionalities that meet the company's needs or are being adapted to environment evolution. Indeed, bad practices and poorly planned changes may introduce design defects known also as *anti-patterns* leading to a decrease in the SBPs quality (Palma et al., 2013; Hertis and Juric, 2014). Even though Business Process Execution Language (BPEL) (Standard, 2007) is becoming a common standard for designing business processes, BPEL is liable to error (Ji et al., 2019). In fact, bad design practices and limited designers' and developers' skills may reduce the BPEL process quality by introducing defects (Palma et al., 2013; Hertis and Juric, 2014). In the past two decades, various defects were identified in SBPs affecting the control flow, data flow, and business process portability (Song et al., 2021; Ji et al., 2019; Song and Jacobsen, 2018; Lübke et al., 2019; Van der Aalst, 1998; Sadiq and Orlowska, 2000; Lenhard and Wirtz, 2016, 2013a; Lenhard, 2017; Koschmider et al., 2019).

☆ Editor: Heiko Koziolek.

* Corresponding author.
E-mail addresses: marwa.daagi@gmail.com (M. Daaji), ali.ouni@etsmtl.ca (A. Ouni), mohamedmohsen.gammoudi@isamm.uma.tn (M.M. Gammoudi), salahb@uaeu.ac.ae (S. Bouktif), mwmvse@rit.edu (M.W. Mkaouer).

[1] https://www.reportlinker.com/p03283202/?utm_source=GNW

https://doi.org/10.1016/j.jss.2023.111767
0164-1212/

Recently, Koschmider et al. (2019) suggested taxonomies for business process model anti-patterns including Syntax errors, Control-flow problems, Understandability problems, Composition defects, Data-flow-related defects, Rule-related defects and Process-related defects. These defects are represented through different models including textually, graphical, and formal models (Song et al., 2021; Palma et al., 2013; Trčka et al., 2009; Ramadan et al., 2018). One of the first attempts to address such defects was proposed by Trčka et al. (2009) where they employed temporal logic CTL to represent the defects used later by the model checking to analyze the business processes. Tracka et al. need to represent business processes using WorkFlow net with Data (WFD-net) before analyzing them because model checkers support only the Petri-net modeling languages as input to check defects. The WFD-net representation is considered a consuming task since it requires a clear start and end point and annotations related to the handling of data. In addition, temporal logic representation has weaknesses since some information can be lost when they only build the reachability graph of a WFD-net. Hence, a preprocessing that converts a business process into a WF-net is required to include information about the data operations in the states. In fact, performing the preprocessing transformations is a costly process, especially with a complex business process that may exhibit a higher possibility of defects. More recently, Palma et al. (2013) defined defects as textual inference rules based on a Backus–Naur Form (BNF) grammar. These rules are applied to identify BPEL process errors. However, such manual rules require a substantial calibration effort to manually inspect and validate such rules in practice, especially in complex business processes. Furthermore, Ramadan et al. (2018) expressed the defects using a graphical query language for BPMN models that supports data-minimization annotations. These queries allow specifying conflicts between data minimization and security requirements as defects.

Although there are rigorous definitions for data-flow defects, Song et al. (2021) defined them using textual description, and therefore authors used algorithms based on the idea of reaching definitions to detect different types of data-flow defects. Since the detection algorithms are time-consuming, Song et al. (2021) proposed a data flow defect prediction approach using machine learning based on process complexity metrics. Song et al. recommended the use of process complexity metrics to reduce the time of the prediction techniques since business processes can be black or gray boxes to third parties, but several process complexity metrics are available. Furthermore, researchers from business process management such as Roy et al. (2014) and Mendling et al. (2007) advocate the use of complexity metrics to predict process defects because they believe that process defects are highly relevant to the corresponding metrics. However, there is no information about the appropriate threshold that characterizes these metrics to predict such defects. In fact, such information will support developers to fix these defects by checking only the considered metrics with low cost and effort. Furthermore, Song et al.'s approach is applied to a limited number of BPEL defect types focusing on data flow-related bad practices while providing black-box prediction models.

Furthermore, while BPEL is a standard for specifying business processes, there are still differences on how BPEL is implemented by different vendors (Kloppmann et al., 2005). In fact, BPEL processes are written or transformed by different tools or developers who do not often follow established best practices when writing their BPEL process ' (Song et al., 2021). This can lead to poor process design/coding solutions, *i.e.*, defects, making it difficult to perform program analysis across different implementations of BPEL (Song et al., 2021). Furthermore, BPEL processes can be complex, with many different activities and interactions. This complexity can make it difficult to perform program analysis and can limit the effectiveness of program analysis techniques that rely on simplifying assumptions (Song et al., 2021).

To address these issues, we propose in this paper a new BPEL defects prediction approach for both data flow and portability defects using process complexity metrics. Our approach aims at predicting the presence of such defects in BPEL code. Hence, predicting BPEL defects allows on one hand business parties to exclude some business partners for B2B collaboration (Song et al., 2021) and on the other hand supports developers to pay more attention to the development of their BPEL processes to avoid potential defects. In addition, predicting BPEL defects using process complexity metrics is gaining increasing attention (Mendling, 2008; Roy et al., 2014; Mendling et al., 2007) from researchers since process defects are highly correlated with process metrics (Song et al., 2021). Our approach does not require any WFD-net representation to analyze the BPEL processes. Our approach consists of generating a set of defect prediction rules learned from real-world instances of business process defects using multi-objective generating (MOGP). Our MOGP-based approach is based on the adaptation of the Non-dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2002), designed for multi-objective optimization problems dealing with conflicting objectives. NSGA-II has shown good performance in solving many optimization problems (Ouni, 2020). In particular, we formulate the problem as an optimization problem with the aim of identifying for each defect type, the best combination of process complexity metrics and their threshold values. To the best of our knowledge, this is the first attempt to use a search-based approach for the business process defects prediction problem. Hence, one of the advantages of our approach is its ability to predict defects before they happen based on the set of complexity metrics and their associated threshold values, while static approaches only detect them after they happen. Consequently, our approach can help reduce the costs by early predicting the emergence of such defects, unlike static analysis-based approaches. Moreover, our approach can work in cases when the BPEL code is not available and the owner only provides some complexity metrics.

To evaluate the effectiveness of our approach, we designed an experimental study on a dataset including 178 real-world BPEL processes belonging to different application domains such as sales, tourism, finance, business, etc. We assess the effectiveness of our multi-objective approach compared to state-of-the-art multi and single-objective search algorithms, as well as advanced supervised learning techniques including Random Forest (RF), decision trees (C4.5), K-Nearest Neighbor (KNN), Logistic Regression (LR), Support Vector Machines (SVM) and Naive Bayes (NB). The achieved results indicate the effectiveness of our approach with 91% of accuracy, 87% of AUC, and 83% of F-measure for all the studied defect types. We also conduct a qualitative analysis of the potential factors that characterize the existence of BPEL defects based on a top-features analysis approach. The prediction rules analysis shows various complexity metrics do impact the existence of each defect type.

The main contributions of this paper can be summarized as follows:

1. A novel formulation of BPEL defects prediction as a multi-objective optimization problem. To the best of our knowledge, this is the first attempt to use a search-based approach for the business process defects prediction problem.
2. An empirical study of our approach compared to different state-of-the-art approaches based on a dataset of 178 real-world BPEL processes. The obtained results reveal that our proposal is more efficient than existing techniques with a median accuracy of 91% compared to existing search-based and machine-learning techniques.

3. Drawing on the findings and suggestions from our study, we discuss the implications of our research for developers, researchers, and BPEL business process users.
4. We provide our replication package containing the necessary materials to reproduce and extend our study (Song, 2018).

The rest of this article is structured as follows. First, we provide the necessary key concepts and we provide a motivating example in Section 2. In Section 3, we describe our NSGA-II-based approach. Section 4 reports and discusses the achieved results. The main implications of our results for BPEL developers, researchers, and business process users are discussed in Section 5. In Section 6, we report the threats that may affect the validity of our study. We review the related work in Section 7. Finally, we summarize the main contributions and outline our future research directions in Section 8.

2. Background and motivation

In this section, we provide the necessary key concepts related to our context including BPEL processes, data flow defects, and portability defects. Then, we provide a motivating example to explain the main challenges that drive our approach.

2.1. Definitions

Business Process Execution Language (BPEL). BPEL is a business process executable language based on XML created to describe Web services interactions (Arkin et al., 2005). BPEL provides a simple and easy manner to incorporate multiple Web services to build new complex services known as "business processes". These business processes involve the execution order of a set of activities. BPEL provides two types of activities: primitive and structured activities. Fig. 1 depicts the structure of a BPEL order process (Yu et al., 2008), showing both primitive and structured activities. This BPEL process consists of an online purchasing process from the e-business domain. First, an e-business merchant receives online orders and then processes them. Initially, the merchant should verify the order for validity before accepting it, and then a confirmation or cancellation of the order is sent to the customer according to the checking result. After receiving both the goods and the invoice, the customer deposits the payment to the bank, and then the payment is transferred to the provider if the customer received the desired goods. More specifically, primitive and structured activities are defined as follows.

- *Primitive activities* include common tasks such as *invoke*, *receive*, *reply* and *assign*. The *invoke* activity consists of invoking a synchronous service or initiating an asynchronous Web service. A synchronous invocation involves both input variables and output variables whereas asynchronous invocation supports only input variables. Receiving a message from an external partner is defined by *receive* activity that involves simply output variables. The action of sending a response to an external partner request that was previously received through a receive activity and involves only input variables is determined by the *reply* activity. The *assign* activity allows to assign a value to a data variable (Song et al., 2021; Hertis and Juric, 2014; Ouyang et al., 2007).
- *Structured activities* are defined to represent the control flow of BPEL processes and involve four main structured activities which are *sequence*, *if*, *while*, and *flow*. The sequential execution order of activities is realized by the *sequence* activity. The *if* activity described alternative structures. It involves an organized set of one or multiple conditions known as "case branches". The *while* activity supports iterative structures. The parallel structures are described by *flow* activity where multiple activities are executed simultaneously while the link is defined to synchronize multiple threads of a flow activity (Song et al., 2021; Hertis and Juric, 2014; Ouyang et al., 2007).

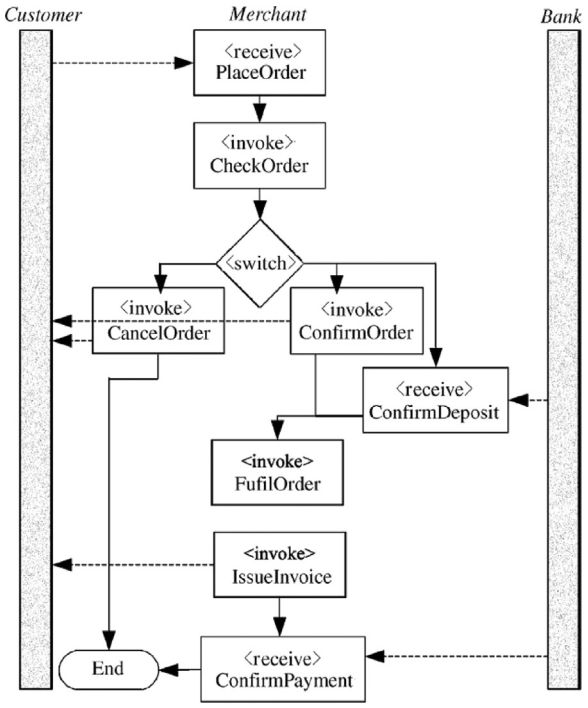


Fig. 1. Structure of a BPEL order process provided by Yu et al. (2008).

Anti-patterns. Known also as design defects, code smells, and bad smells. Anti-patterns are defined as poor design and implementation practices/behaviors. According to Kral and Zemlicka (2007), Král and Zemlicka (2009), such poor solutions to recurring design problems usually cause a decrease in software quality. They could be introduced during the initial design or during the evolution process unintentionally due to human factors or limited resources. For the sake of consistency, we denote such bad practices as "defects". Several types of defects may occur in BPEL processes such as:

Data flow defects. Several types of data-flow defects may occur in a process P. In this paper, we studied three common types of data flow defects that are common in practice according to Trčka et al. (2009) and Song et al. (2021), namely *missing input*, *redundant output*, and *lost output*.

- Missing Input: An activity a_i involves a missing input defect if there is one or more of its input variables $V \in I(a_i)$ are not defined by any activity that precedes a_i.
- Redundant Output: An activity a_i involves a redundant output defect if at least one of its output variables $V \in O(a_i)$ is not used by any activity that follows a_i.
- Lost Output: An activity a_i involves a lost output defect if at least one of its output variables $V \in O(a_i)$ is not used before being redefined by another activity a_k.

Portability defects. A BPEL process presents a portability defect if a selected element el_i from E in a process definition p is not supported by a suite of engines. We note that E defines the set of process elements, A defines the process of activities, S defines the communication activities where $A \subset E$ and $S \subset A$. If all

considered engines do not support an element of p, it is a fully nonportable process, else if some engines support this element, it is a partially portable process. The presence of portability defects involves additional effort and costs to refactor the process partially or from scratch (Lenhard and Wirtz, 2013a,b, 2016; Petcu and Vasilakos, 2014).

2.2. Motivating example

To illustrate the salient aspects of BPEL defects, we consider `TravelReservationService`,[2] a synchronous BPEL service. The BPEL file is too large, which consists of 364 lines of code. The main objective of this business process is to manage Hotel Reservation, Vehicle Reservation, and Airline Reservation and Itinerary. The BPEL service involves multiple serious lost output defects observed for example with the variables `ItineraryOut` and `ReserveVehicleIn` in the *CopyItineraryIn* assign activity. Moreover, two lost output errors related to the variable `ItineraryOut` are observed in the *CopyAirlineReservation* and *CopyVehicleReservation* assign activities. Furthermore, we report two redundant output defects observed with the variable `CancelAirlineOut` in the *CancelAirline* and *CancelVehicle* invoke activities. This output is not defined by any activity that follows the invoke activities.

Indeed, as pointed out by Song et al. (2021) redundant output represents an undesirable defect that does not cause fatal issues as missing input does; however, it is still undesirable in BPEL processes. In fact, output variables are generally provided when Web services are requested and since the output variable is not exploited or is lost this may lead to additional service invocation costs. To detect possible defects, practitioners should read through each line of code in a BPEL file to keep track of app variables and their flows and usage. Hence, the problem of detecting BPEL defects is still challenging, time-consuming, and requires non-trivial efforts to diagnose and fix these defects especially when the BPEL process is directly written by developers or transformed from abstract process models (Song et al., 2021). Furthermore, as pointed out by Song et al. "Business processes can be black or gray boxes to third parties in clouds, but several process complexity metrics are available" which indicates the inability to see the system's internal workings.

Hence, as pointed out by Roy et al. (2014), Mendling et al. (2007) complexity metrics support the prediction of process defects even in a complex business process since process defects are highly relevant to the corresponding metrics. In this context, to detect existing poor design practices in BPEL processes, tool support is needed which motivates us to build an automated approach to generate BPEL process defects prediction rules driven by common process complexity metrics.

3. The proposed approach

This section describes our BPEL defects prediction approach, which is a gray box approach that provides data flow and portability defects prediction rules based on process complexity metrics.

3.1. Approach overview

Fig. 2 describes the general overview of our approach. The idea consists of collecting a set of real-world BPEL processes that are affected by BPEL defects. These instances of each defect type will be used to generate defect prediction rules in the form of a combination of process complexity metrics and their threshold values. To generate such prediction rules, we use a search-based approach that adopts multi-objective genetic programming (MOGP) in order to find the best combinations of process complexity metrics and the right threshold value for each metric. As depicted in Fig. 2, our search-based approach consists of two main phases: the learning phase and the prediction phase. In the learning phase, we aim to build a business process defects prediction model for each defect type. This model learns from real-world BPEL defect examples using MOGP. In particular, we adopt the non-dominated sorting genetic algorithm (NSGA-II) (Deb et al., 2002) as a search algorithm. In the prediction phase, we test each defect prediction model on new BPEL processes (test data).

More specifically, in the learning phase, our approach takes as input (1) real-world examples of BPEL process defects, and (2) a list of process complexity metrics computed from these BPEL processes (step 1). Thereafter, optimal BPEL defect prediction rules will be generated from these BPEL processes using NSGA-II (step 2). Once generated, the obtained prediction rules will be used to predict defects from other BPEL processes (step 3). For each defect type, our approach generates the appropriate combination of metrics/thresholds that allows to predict as many as possible defects in the pool of examples. Currently, our approach supports the prediction of three types of data flow defects, which are *missing input*, *redundant output* and *lost output*, and two types of portability defects, the *partially portable process* and *non-portable process*.

3.2. Step 1: Process complexity metrics computing

Our approach is based on a set of process complexity metrics to characterize the various symptoms of BPEL process defects. In particular, we employ 31 business process complexity metrics from the literature (Mendling, 2008; Roy et al., 2014; Cardoso et al., 2006; Latva-Koivisto, 2001; Cardoso, 2007; Nissen, 1998). Table 1 provides the list of metrics we use in our approach. These metrics belong to 12 categories, such as the *size, density, partitionability, connector interplay, cyclicity* and *concurrency*. To calculate these metrics, we implemented a BPEL parser that walks through the BPEL file and generates a CSV file containing the metrics values.

3.3. Step 2: Generation of BPEL defects prediction rules using NSGA-II

In this subsection, we describe our adaptation of NSGA-II to generate prediction rules including (i) the search algorithm, (ii) solution representation, (iii) fitness function, and (iv) genetic operators.

3.3.1. Search algorithm

Our solution is designed using genetic programming (GP) where the solution encoding takes the form of a binary tree, the metrics represent the leaf nodes, and each internal node corresponds to the logical operator between them. We implement our GP-based approach using the non-dominated sorting genetic algorithm (NSGA-II) (Deb et al., 2002) as a search algorithm to predict BPEL process defects. Indeed, GP is a powerful extension to the genetic algorithm that extends the model of genetic algorithms to the space of computer programs (Deb et al., 2002; Ouni et al., 2015b, 2012; Almarimi et al., 2022). However, although it inherently has a variable length that makes it more flexible, it often grows in complexity, especially with a complex business process. Indeed, NSGA-II is mainly designed for multi-objective optimization problems dealing with conflicting objectives. It uses an elitist principle, an explicit diversity-preserving mechanism

[2] https://github.com/stilab-ets/BPELantipatterns/blob/main/BPEL_Dataset/TravelReservationService%20if.bpel

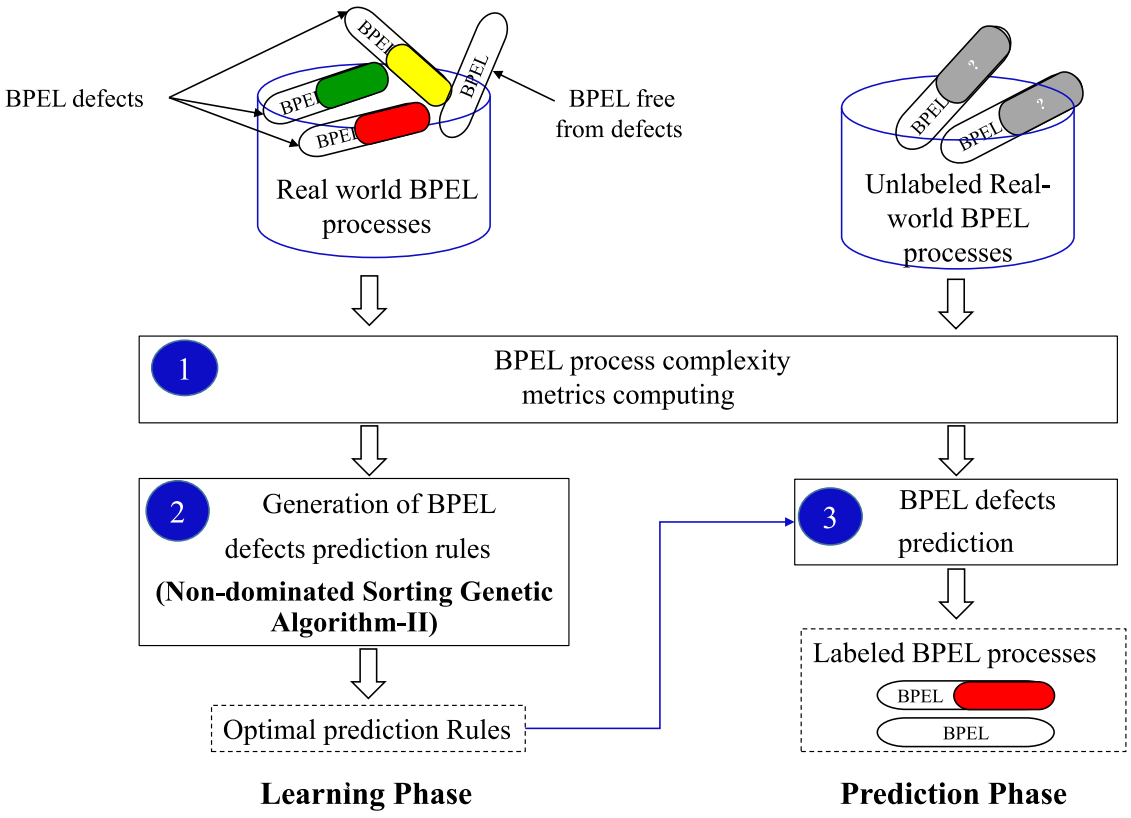


Fig. 2. Approach Overview.

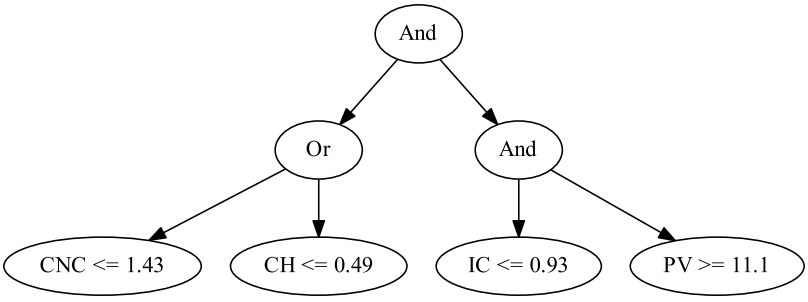


Fig. 3. Solution representation for the Missing_Intput defect.

(Crowding distance), and emphasizes the non-dominated solutions. As depicted in algorithm 1, NSGA-II starts by generating a population P_t as the parent population at tth generation with a size of N (line 1). Then, we consider Q_t as the offspring population generated at the tth generation using crossover and mutation operations (line 4). U_t refers to the combined population defined as $U_t = P_t \cup Q_t$(line 6). Next, the entire population is divided into multiple non-domination levels ($F1$, $F2$, and so on) using Fast-non-dominated-sort (line 7). In the selection process, the new population S_t is created to maintain diversity by selecting individuals of non-domination levels one at a time from $F1$ until $\mid S_t \mid \geq N$ (line 8–11).

Considering F_l is the last non-dominated level (line 12). If $\mid S_t \mid = N$, the new generation begins with $P_{t+1} = S_t$ (line 13–14), else individuals from 1 to (l−1) fronts are chosen for P_{t+1} (line 15–16), and the rest of population individuals are selected from F_l (line 17). The aforementioned procedure is reiterated until the size of $P_{t+1} = N$ (line 5).

3.3.2. Solution representation

BPEL defects prediction rules represent the solution for our problem, where its structure followed the form **IF - THEN**. The **IF** clause defines the set of conditions that satisfy the prediction of a given defect type. These conditions are the combination of BPEL process metrics and their relative threshold values. The combination of these metrics is defined using logical operators including *OR*, *AND*, and *XOR*. If a BPEL process satisfies the condition then, a specific type of defect is figured in **THEN** clause. Hence, a solution is designed in the form of a binary tree where the metrics represent the leaf nodes and each internal node corresponds to the logical operator between them. Fig. 3 depicts an example of a prediction rule that corresponds to the missing_input defect in a given iteration *i*.

3.3.3. Fitness function

A candidate solution for BPEL defects prediction is evaluated using a fitness function. To assess its effectiveness, we define two objective functions to be optimized based on three basic metrics, the True Positive (TP), False Positive (FP), and False Negative (FN) rates.

Table 1
List of process complexity metrics.

Type	Metric	Description
Size	NOA	The number of basic activities.
	NOAC	The number of all activities.
	Edges	The number of edges in the graph.
	Diameter	The number of nodes in the longest path from the entrance to the exit.
	CW	Different weights for different types of activities
	F_in	The number of input variables.
	F_out	The number of output variables.
	$F\_(I+O)$	The number of all variables.
	PL	The process length $PL = n_1 * log_2(n_1) + n_2 * log_2(n_2)$, where n_1 and n_2 are the number of unique structured activities and the number of unique variables, respectively.
	PV	The process volume $PV = (N1 + N2) * log_2(n_1) + n_2$; where $N1$ and $N2$ are the total number of structured activities and the total number of variables, respectively.
	PD	The process difficulty $PD = (n_1/2) * (N2/n_2)$.
Density	Δ	Δ is the density of the process graph and is defined as the ratio of the number of arcs to the maximum number of arcs for the same number of nodes. $\Delta = \frac{\|E\|}{\|N\|.(\|N\|-1)}$.
	CNC	CNC is the Coefficient of Connectivity and is defined as the ratio of the number of nodes to the number of arcs. $CNC = \frac{\|E\|}{\|N\|}$.
	$\bar{d_C}$	$\bar{d_C}$ is the Average Connector Degree. $\bar{d_C} = \frac{1}{\|C\|}\sum_{c\in C} d(c)$, where $C \in \{if, flow\}$, and $d(c)$ denotes the degree of c.
	$\widehat{d_C}$	$\widehat{d_C}$ defines the maximum degree of a connector $\widehat{d_C} = max\{d(c) \| c \in C\}$, where $d(c)$ denotes the degree of c.
Partitionability	Π	Π defines the separability and represents the ratio of the number of cut vertices to the number of nodes. $\Pi = \frac{\|\{n\in N \| n \ is \ a \ cut\_vertex\}\|}{\|N\|-2}$.
	Ξ	Ξ defines the sequentiality and is computed as the ratio of the number of arcs between non-connector nodes to the total number of arcs. $\Xi = \frac{\|E\cap(N\setminus C\times N\setminus C)\|}{\|E\|}$.
	Λ	Λ indicates the maximum nesting of structured blocks in a process. The metric depth Λ is computed by finding the maximum over the depths of all nodes.
Connector Interplay	*CH*	*CH* defines the Connector Heterogeneity. $CH = -\sum_{l\in C} p(l).(log_4(p(l)))$ where $pl = \frac{\|C_{(l)}\|}{\|C\|}$, and C_l is the set of connectors of type l.
	CFC	CFC defines the Control Flow Complexity. $CFC = CFC(t)$ where t is the BP top-level scope: $CFC(t) = 1$, $t \in$basic activities; $CFC(sequence) = \sum_{t\in sequence} CFC(t)$; $CFC(if) = n * \sum_{t\in if} CFC(t)$; $CFC(while) = log_2(CFC(t) + 2)CFC(t)$, $t \in while$; $CFC(flow) = (n-1) * \sum_{t\in flow} CFC(t)$, l is the number of cross-boundary links; $CFC(pick) = (2^n - 1) * \sum_{t\in pick} CFC(t)$. n is the number of paths in an alternative structure (“if”) or the number of activities in a parallel structure (“flow”)).
Cyclicity	CYC	CYC defines the Cyclicity and is computed as the ratio of the set of nodes which lie on some cycle to the total number of nodes in the process graph. $CYC = \frac{\|N_C\|}{\|N\|}$.
Concurrency	TS	The sum of all concurrent threads in a process.
Others	FI	The number of process invocations.
	FO	The number of service invocations.
Balance	DFC	*DFC* is the data flow complexity metric, $DFC = \frac{\|UI\cap UO\|}{\|UI\cup UO\|}$, where, $UI = \cup_{n\in N} I(n)$ is the union of all input variables of the activities in a (BPEL) process; $UO = \cup_{n\in N} O(n)$ is the union of all output variables of the activities in a (BPEL) process.
	IC	IC is referred to the input flow complexity, $IC = \frac{\|UI\cap UO\|}{\|UI\|}$.
	OC	OC is referred to the output flow complexity, $OC = \frac{\|UI\cap UO\|}{\|UO\|}$
Basic	M_b	M_b defines the basic portability metric. Its is computed as $M_{port}(p) = 1 - \frac{C_{port}(p)}{C_{new}(p)}$, where, $C_{new(p)}$ is the total number of elements in a process definition p that will be rewritten from scratch. $C_{port(p)}$ is the number of elements from the set E that have to be rewritten when porting the process.
Weighed Elements	M_e	M_e refers to the elements portability metric. It is defined as: $M_e(p) = 1 - \frac{\sum_{i=1}^{N_{el}} max_{j=1\ldots N_{ta}}(V(ta_j, el_i) * D_{ta}(taj))}{N_{el} * N_{engines}}$, $C_{new}(p) = N_{el} * N_{engines}$, where N_{el} is the number of process elements and $N_{engines}$ is the number of engines under consideration. $C_{port}(p) = \sum_{i=1}^{N} C_{el}(el_i)$ where C_{el} is the cost of the ith element from the set of elements E; $C_{el}(el_i) = max_{j=1\ldots N_{ta}}(V(ta_j, el_i) * D_{ta}(taj))$; $V(ta_j, el_i)$ denotes the testing function that returns 1 if el_i violates a test assertion ta_j and 0 otherwise. $D_{ta}(ta_j)$ denotes the degree of the test assertion ta_j.

(*continued on next page*)

Table 1 (*continued*).

Activity	M_a	M_a defines the activity portability metric. It is computed as: $M_a(p) = 1 - \frac{\sum_{i=1}^{Na} max_{j=1...Nta}(V(ta_j,a_i)*D_{ta}(taj))}{N_a*N_{engines}}$. $C_{new}(p) = N_a * N_{engines}$, where N_a is the number of process activities and $N_{engines}$ is the number of engines under consideration. $C_{port}(p) = \sum_{i=1}^{N} C_{el}(a_i)$ where C_{el} is the element cost of the ith activity from the set of activities A.
Service Communication	M_s	M_s refers to the service communication portability metric. It is defined as $M_s(p) = 1 - \frac{\sum_{i=1}^{Ns} max_{j=1...Nta}(V(ta_j,s_i)*D_{ta}(taj))}{N_s*N_{engines}}$. $C_{new}(p) = N_s * N_{engines}$, where N_s is the number of activities for service interaction and $N_{engines}$ is the number of engines under consideration. $C_{port}(p) = \sum_{i=1}^{N} C_{el}(s_i)$ where C_{el} is the element cost of the ith activity from the set of service interaction activities S.

Algorithm 1 Pseudo code of Generation t of NSGA-II algorithm

Input: P_t parent population
Output: P_{t+1}
Offspring population $Q_t = Genetic\_operator(P_t)$
while Stopping criteria not achieved **do**
  $R_t = P_t \cup Q_t$
  $(F1, F2, ...) = Fast - non - dominated - sort(R_t)$
  New population $P_{t+1} = \emptyset$, $i = 0$
  repeat
    Apply crowding-distance-assignment(F_i)
    $P_{t+1} = P_{t+1} \cup F_i$
    $i = i + 1$
  until $| P_{t+1} | \geq N$
  $Sort(F_i, \prec n)$
  $P_{t+1} = P_{t+1} \cup F_i[1 : N - |P_{t+1}|]$
  Offspring population $Q_{t+1} = Genetic\_operator(P_{t+1})$
  $t = t + 1$
end while

- **Objective function #1. Maximize the precision:** The precision is the ratio of positive classes that are correctly identified by the model over total positive records. Hence, *Precision* of a solution S is described as follows:

$$Max\{Precision(S)\} = Max\frac{TP}{TP + FP} \quad (1)$$

- **Objective function #2. Maximize the recall:** The recall computes the ratio of positive classes correctly detected. This metric gives how good the model is to recognize a positive class. Hence, *Recall* of a solution S is defined as follows:

$$Max\{Recall(S)\} = Max\frac{TP}{TP + FN} \quad (2)$$

3.3.4. Genetic operators

Crossover and mutation as genetic operators are used by population-based search algorithms to evolve candidate solutions in each generation and improve their fitness.

Crossover: is in charge of generating new solutions based on a recombination of current ones. In our adaptation, two parents are chosen randomly to exchange information based on a random, single cut-point crossover. The sub-trees around this cut-point are swapped between the two parents. Thus, the produced offsprings carry some information from both parents.

Mutation: is designed to preserve diversity from one generation to another by introducing a slight, random modification into candidate solutions. Mutation can alter multiple nodes. For a randomly chosen offspring, a mutation operator can either (1) substitute a metric with another one or change its threshold with another threshold value, or (2) substitute a logical operator with another one.

3.4. Step 3: Prediction phase

In this phase, we apply the obtained prediction models on a given BPEL process based on its complexity metrics values as input. Hence, the output corresponds to a binary response indicating whether the considered BPEL is infected by a defect or not.

4. Validation

To assess the performance of our proposed approach, we conduct an experimental study. We first describe our research questions. Then, we explain the experimental setup and report and discuss the obtained results.

4.1. Replication package

Our comprehensive dataset used in this study is publicly available at Anon (2021) for replication and extension purposes. Specifically, we provide the list of the studied BPEL processes in our dataset, along with the detailed results for all the studied algorithms and the results of Irace package for each defect class.

4.2. Research questions

To assess the effectiveness of our approach, we aim to answer the following four research questions:

RQ1 (SBSE Validation): How does the proposed NSGA-III approach perform compared to Random Search (RS), mono-objective algorithm (GA), and state-of-the-art multi-objective search algorithms?

RQ2 (Performance assessment comparing to ML): How does the proposed NSGA-II approach perform compared to machine learning algorithms?

RQ3 (Variability analysis): What types of BPEL process defects does our approach predict correctly?

RQ4 (Features weight): What are the most important features that can characterize the existence of BPEL defects?

RQ5 (Threshold values analysis): What are the threshold values associated the most influential features for each BPEL defect type?

4.3. BPEL dataset

To assess our approach, we use a set of 178 existing BPEL processes proposed by Song et al. (2021) belonging to different domains such as sales, tourism, finance, business, government, logistics, and communications. The used dataset is available in our replication package.[3] In this study, we considered three common types of data flow defects and two common types of portability

[3] https://github.com/stilab-ets/BPELantipatterns/tree/main/BPEL_Dataset

Table 2
Distribution of the studied defects across the used Data Set.

Dataset	Defect type	# of instances
178 BPEL processes	*Missing_Input*	36
	Redundant_Output	72
	Lost_Output	24
	Non_Portable	139
	Partially_Portable	27

defects that correspond to the most recurrent defect in business process (Song et al., 2021; Lenhard and Wirtz, 2016). Data flow defects include *missing input, lost output* and *redundant output*, whereas portability defects include the *non portable process* and *partially portable process* (see Table 2).

4.4. Experimental setup

4.4.1. Analysis method

To answer **RQ1**, we first compare our approach with the Random Search (RS) algorithm (Karnopp, 1963; Harman et al., 2010) to justify the need for adopting an intelligent technique for the problem of BPEL defects prediction. Although RS is considered a simple technique of search algorithms, it may fail to identify optimal solutions as it lacks the ability to use the genetic operators (Harman et al., 2010). Furthermore, to justify the adoption of NSGA-II as a multi-objective optimization algorithm, we compare its performance to a mono-objective Genetic Algorithm (GA) (Mitchell, 1998) where we aggregate both objectives into a single objective to justify the multi-objective formulation. Our single-objective fitness function maximizes the F-measure that corresponds to the weighted average of precision and recall, defined as follows:

$$Max\{F\text{-}measure\} = Max\frac{2 * precision * recall}{precision + recall} \tag{3}$$

Prediction performance. To assess the performance of each algorithm in predicting BPEL defects, we use F-measure, Accuracy, and Area Under Curve (AUC) (Kessentini and Ouni, 2017; Ouni et al., 2017b; Read et al., 2011; Hanley and McNeil, 1982) metrics, which are defined as follows:

$$F - measure = \frac{2 * (Recall * Precision)}{Recall * Precision} \tag{4}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5}$$

$$AUC = \frac{1 + \frac{TP}{TP+FN} - \frac{FP}{FP+TN}}{2} \in [0, 1] \tag{6}$$

In particular, AUC is designed to quantify the global performance of a binary classifier. The model that achieves an AUC score close to 1 is considered as a good model with the highest measure of class separability (Hanley and McNeil, 1982).

Algorithms performance. We assess the effectiveness of NSGA-II over different multi-objective evolutionary algorithms (MOEAs) to justify its adoption for the BPEL defect prediction problem. The considered MOEAs include NSGA-III (Deb and Jain, 2014), Strength-Pareto Evolutionary Algorithm (SPEA2) (Zitzler et al., 2001) and Indicator-Based Evolutionary Algorithm (IBEA) (di Pierro et al., 2007). We consider these algorithms as they are the most used ones in software engineering based on recent studies (Ouni, 2020; Harman et al., 2012; Mkaouer et al., 2015).

To compare the performance of the studied MOEAs, we employ common performance indicators including the *hypervolume*(HV), *Inverted Generational Distance* (IGD), and Contribution (IC) (Meunier et al., 2000; Coello and Sierra, 2004; Bezerra et al., 2017; Brockhoff et al., 2008) which are defined as follows:

- Hypervolume (HV): quantifies the part of volume covered by the non-dominated solution set. If HV threshold value is near to 1, the solution is considered closer to the optimal Pareto front (Brockhoff et al., 2008).
- Inverted Generational Distance (IGD): computes the average distance in objective space between Pareto front solution and the nearest individual in the true Pareto front (Coello and Cortés, 2005). The best MOEA refers to this that has IGD value close to zero.
- Contribution (IC): measures the percentage of non-dominated solutions that are in the Reference Front (RF) (Meunier et al., 2000). While IC depends on the number of obtained non-dominated solutions, it penalizes an algorithm if it generates 'few but excellent solutions. The higher IC, the better.

To answer **RQ2**, we compare our NSGA-II approach to Song et al. (2021) which use supervised learning algorithms including decision trees (C4.5), K-Nearest Neighbor (KNN), Support Vector Machines (SVM) and Naive Bayes (NB) based on the same complexity metrics and BPEL processes. Furthermore, we compare our approach to two additional widely-used machine learning algorithms, Random Forest (RF) and Logistic Regression (LR). These algorithms are commonly used in solving several machine-learning problems in software engineering. To assess the performance of each algorithm in predicting BPEL defects, we use Area Under Curve (AUC), F-measure, and Accuracy as performance metrics, previously defined in RQ1.

ML preprocessing. Before using supervised learning algorithms, we study the multicollinearity between the used features. Indeed, the presence of multicollinearity issues can lead to unexpected interpretations of the model. Checking for multicollinearity can help to get the most insight out of data and do not overlook important relationships. More specifically, multicollinearity occurs when independent features (metrics) are correlated. Thus, we conducted a pairwise correlation coefficients analysis between the features. If the degree of correlation between two features $\geq$ 0.8 (Berry et al., 1985), then, we discard one variable and preserve the second for prediction.

Validation technique. To validate our study, we used a 10-fold cross-validation procedure. The used BPEL processes dataset is randomly partitioned into 10 equal-sized subsamples. For each defect type, a single fold of BPEL processes is retained as the validation data for testing the model, and the remaining nine folds are used as training data. The cross-validation process is then repeated 10 times, with each of the 10 sub-samples used exactly once as the validation data. Hence, all the performance metrics are computed automatically by iterating over all subsamples based on a comparison between the predicted defects and the expected one.

To answer **RQ3**, we analyze whether our approach has a bias towards the prediction of some particular defect types. Hence, we

Table 3
Evolutionary search algorithms parameters tuning.

Algorithms	Defects	Parameters			
		Population size	Maximum number of Generation	Crossover probability	Mutation probability
NSGA-II, NSGA-III, IBEA, SPEA2, GA, RS[a]	*Missing_Input*	60	500	0.21	0.96
	Redundant_Output	20	1000	0.64	0.15
	Lost_Output	20	800	0.96	0.09
	Non_portable	80	800	0.96	0.05
	Partially_Portable	80	900	0.93	0.18

[a]For RS, crossover and mutation probabilities are not considered, since both operators were disabled.

Table 4
Machine learning Algorithms parameters tuning.

Algorithm	Parameters	Values
KNN	K nearest neighbors	1
SVM	Kernel function	Radial basis function
NB	useSupervisedDiscretization	TRUE
RF	Maximum depth of the tree	10
C4.5	Confidence Factor	0.25
LR	Ridge	1.0E −8

investigate the stability of NSGA-II in predicting each defect type, in terms of our three considered performance measures, accuracy, F1, and AUC. A low-performance score variability across the considered defect types indicates the stability of our approach.

To answer **RQ4**, we analyze the factors influencing the emergence of defects in business processes. The identification of such factors can provide further insights and supports developers to avoid potential defects in their SBPs. In our experiments, we analyze the list of features that appear in the generated prediction rules. More specifically, we compute for each defect type, the percentage of rules in which a feature appears across all the generated optimal rules by NSGA-II. The more a feature appears in the prediction rules, the more it is important to characterize that defect.

To answer **RQ5**, we study the distribution of the threshold values for the Top-5 features in each BPEL defect type. Such information allows developers to pay more attention to these metrics to avoid the presence of defects. In our experiments, we analyze the Pareto optimal solutions which are the combination of the most influential BPEL metrics and their relative threshold values. More specifically, we analyze for each feature in the Top-5 list, the distribution of its threshold values.

4.4.2. Statistical test method

Due to the stochastic nature of the used algorithms, our statistical validation consists of running each algorithm 31 times, following Arcuri and Briand's guidelines (Arcuri and Briand, 2011). To gain statistical evidence when comparing our approach with each algorithm, it is necessary to employ suitable statistical tests. To this end, we analyze the statistical significance based on the non-parametric statistical hypothesis test called Wilcoxon signed rank test (Wilcoxon et al., 1970) (significance level $\alpha = 0.05$). The Wilcoxon test is suitable for our experimental study since it does not impose normally distributed data. Moreover, we adjusted the p-values using Bonferroni correction (Hochberg, 1988; Arcuri and Briand, 2014) due to multiple pairwise comparisons. Moreover, we use Cliff's delta statistic, *d*, Cliff (1993) to quantify the effect size and thus to assess the difference magnitude. The effect size is considered negligible when $|\delta| < 0.147$, small when $0.147 \leq |\delta| < 0.33$, medium when $0.33 \leq |\delta| < 0.474$ and large otherwise (Romano et al., 2006).

4.4.3. Parameter setting using irace package for search algorithms

Meta-heuristic search algorithms have an essential step before being used which is parameter tuning. The irace package (López-Ibáñez et al., 2016) provides an automatic configuration tool for tuning optimization algorithms, designed to find optimal parameters configuration of a (target) algorithm. Irace receives as input:

1. A parameter space definition corresponding to the parameters of the target algorithm that will be tuned. In this work, we set four main parameters which are the size of the population, number of generations, Crossover, and Mutation.
2. A set of instances that correspond to the BPEL dataset for which the parameters must be tuned. The used BPEL processes data-set is randomly partitioned into 10 equal-sized sub-samples. For each defect type, a single fold of BPEL processes is retained as the validation data for testing the model, and the remaining nine folds are used as training data.
3. A set of options for irace that define the configuration scenario. We set the maximum number of runs (invocations of targetRunner) that will be performed to maxExperiments = 1000. It determines the maximum budget of experiments for the tuning. Furthermore, we set the statistical test to a t-test with Bonferroni's correction for multiple comparisons. Indeed, the statistical test used in irace identifies statistically bad performing configurations that can be discarded from the race in order to save budget. The statistical test is used to compare the cost of the configurations, which has an effect on the tuning results. the best configurations should obtain the best average solution cost.

To predict different BPEL defects, the scenario was to provide for each defect type the best configuration that maximizes the hypervolume computed by the target algorithm. Therefore, since irace looks for the minimum cost of the solution (configuration), we negate the hypervolume value before ranking the configurations from the best to the worst. Finally, to make a fair comparison, we set the best configuration for all the algorithms to assess their performances.

Table 3 reports the best configuration for each defect class. For the supervised learning algorithms, we use WEKA v3.6.11, and all the parameters were set in Table 4.

4.5. Experimental results

This section reports the obtained results for RQ1, RQ2, RQ3 and RQ4.

Table 5
The obtained results of the search algorithms for each defect types.

BPEL defects	Metrics	GA	IBEA	NSGA-II	NSGA-III	RS	SPEA2
Missing_Input	F1	34.0%	**82.9%**	**82.9%**	**82.9%**	34.0%	**82.9%**
	Accuracy	25.0%	**94.4%**	**94.4%**	**94.4%**	22.2%	**94.4%**
	AUC	50.0%	**92.2%**	**92.2%**	**92.2%**	50.0%	**92.2%**
Redundant_Output	F1	61.9%	**89.9%**	**89.9%**	**89.9%**	60.2%	**89.9%**
	Accuracy	66.7%	**94.1%**	91.3%	**94.1%**	58.7%	91.3%
	AUC	64.8%	**92.3%**	**92.3%**	**92.3%**	55.9%	**92.3%**
Lost_Output	F1	40.0%	50.0%	**58.3%**	**58.3%**	32.5%	**58.3%**
	Accuracy	86.1%	**88.9%**	**88.9%**	**88.9%**	83.3%	**88.9%**
	AUC	70.3%	69.3%	**73.4%**	**73.4%**	64.6%	**73.4%**
Non_Portable	F1	87.4%	85.2%	**89.3%**	85.7%	87.4%	87.3%
	Accuracy	77.8%	80.6%	**83.3%**	77.8%	77.8%	**83.3%**
	AUC	50.0%	**75.0%**	**75.0%**	**75.0%**	50.0%	**75.0%**
Patially_Portable	F1	44.4%	66.7%	**70.8%**	66.7%	34.3%	66.7%
	Accuracy	75.0%	**94.4%**	91.7%	91.7%	75.0%	91.7%
	AUC	63.8%	84.8%	**86.6%**	79.5%	63.8%	**86.6%**
All_Defects	F1	44.4%	**82.9%**	**82.9%**	**82.9%**	34.3%	**82.9%**
	Accuracy	75.0%	**94.1%**	91.3%	91.7%	75.0%	91.3%
	AUC	63.8%	84.8%	**86.6%**	79.5%	55.9%	**86.6%**

4.5.1. Results for RQ1 (SBSE validation)

Fig. 4 and Table 5 report the obtained results of MOEAs compared to mono-objective algorithms. Over 31 runs, MOEAs achieved the highest performance values regarding F-measure, Accuracy, and AUC in all defect types. As reported in Table 5, NSGA-II achieved 82.9% of F-measure, whereas GA and RS reached only 44.4% and 34.3%, respectively. Furthermore, NSGA-II reported a 91.3% of accuracy while GA and RS achieved 75%, respectively. Finally, NSGA-II reached 86.6% of AUC whereas GA and RS reported 63.8% and 55.9%, respectively. The Wilcoxon test results reported in Tables 6, 7, and 8 reveal that NSGA-II outperforms both RS and GA with a significant difference and large effect size in most of the defect types. Therefore, our formulation passes the sanity check and the obtained results justify the need for an intelligent search technique to explore the search space.

Compared to other MOEAs, we observe that NSGA-II and SPEA2 achieve the best median value of F-measure and AUC for all defect types. However, NSGA-II performs better than SPEA2 in the prediction of Non_portable and partially_portable defects. Indeed, NSGA-II achieves 89.3% and 70.8% of F1 while SPEA2 reports only 87.3% and 66.7%, respectively. Furthermore, we report that NSGA-II achieves 91.3% of accuracy for all defect types whereas IBEA reaches the highest median score with 94.1%, however, the difference is not statistically significant with negligible effect size.

Moreover, Table 9 reports the results of the performance indicators, hypervolume (HV), inverted generational distance (IGC), and contribution (IC). We observe that the HV, IGD, and IC are non-zero values, which justifies that the problem is not a single objective problem. In fact, these performance indicators are typically used to measure the quality of a Pareto front approximation in multi-objective optimization problems where there are two or more conflicting objective functions (Riquelme et al., 2015; Blank and Deb, 2020; Deb and Jain, 2002). In this context, the precision and recall representing our objective functions are often conflicting especially with the imbalanced nature of our dataset. In order to increase the true positive, the number of false positives is often increased, resulting in reduced precision according to Ma and He (2013).

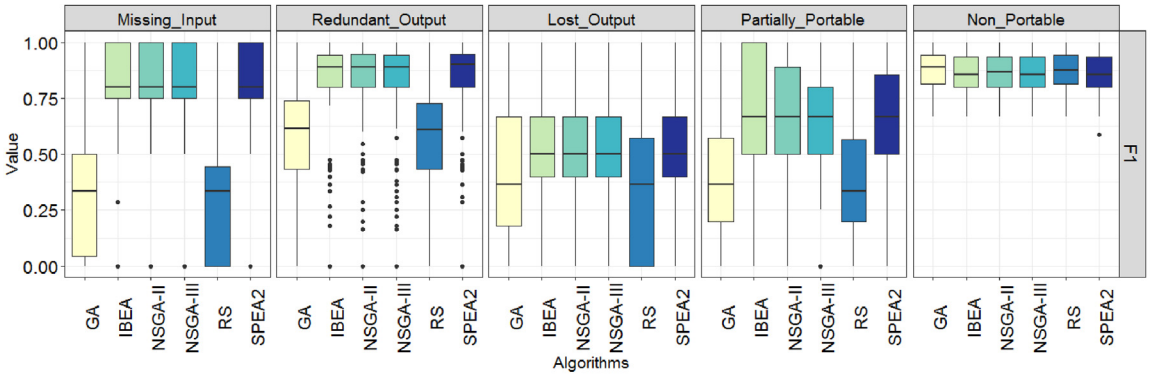
For the IGC indicator, the scores are not significantly different. We observe that NSGA-II, SPEA2, and NSGA-III achieve the same distance from approximations to the Pareto front with an average of 0.092. For the IC indicator, we observe that NSGA-II and NSGA-III achieve the best IC score, especially in the prediction of missing input with 0.026. However, the scores are barely distinguishable between all the algorithms. For the HV indicator, IBEA reaches 0.74 while NSGA-II achieves 0.72 for all defect types with no significant differences compared to other MOEAs. Hence, the obtained results from the different comparisons of NSGA-II with RS, GA, and other MOEAs in terms of performance indicators motivate our choice of this search technique to solve the BPEL defect prediction problem.

4.5.2. Results for RQ2 (Performance evaluation with ML)

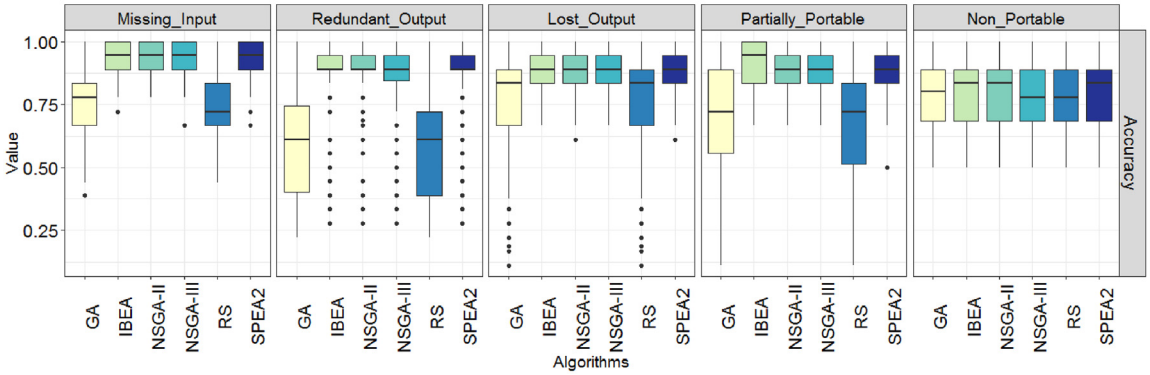
Fig. 6 reports the achieved results by NSGA-II compared to each supervised learning algorithm. We observe that for all the defect types except the Non_portable defect, NSGA-II outperforms the supervised learning algorithms concerning accuracy and AUC. As depicted in Fig. 6(b), NSGA-II reaches 91.3% of accuracy while Random Forest achieves only a median value of 87% for all defect types. Moreover, Fig. 6(c) shows that NSGA-II achieves a high AUC with a median value of 87% for all defect types while Random Forest achieves only 82.7%. For the F-measure, we observe that Random Forest achieves a score of 86.7% which outperforms NSGA-II achieving 82.8%, even though NSGA-II provides a higher F-measure in the prediction of two defects out of five, including the *Redundant_ Output* and *Non_portable*, as shown in Fig. 6(a).

Based on these results, we can conclude that NSGA-II provides promising results compared to supervised learning algorithms. This could be justified by the fact that NSGA-II had a better trade-off between both positive (*i.e.*, presence of defect) and negative (*i.e.*, absence of defect) accuracy, which indicates that our approach is advantageous over supervised learning algorithms when building prediction rules for imbalanced data-sets.

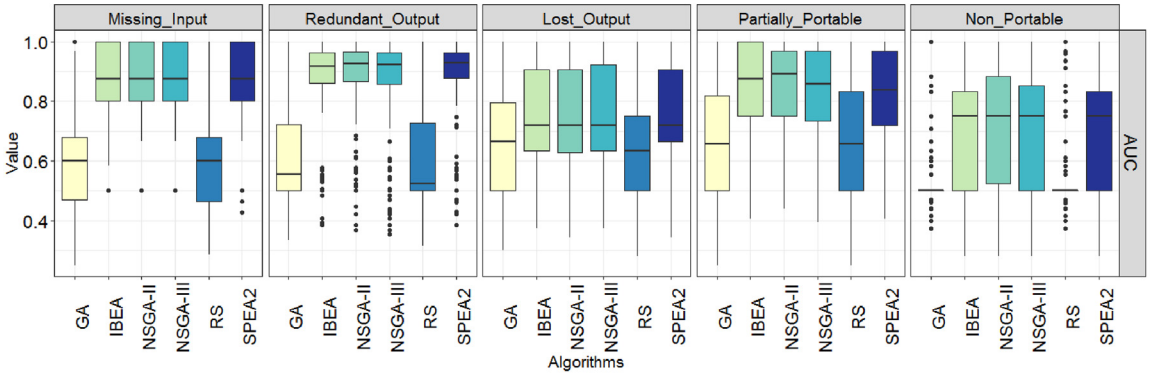
Although NSGA-II achieves better accuracy compared to ML, MOEAs typically require a larger computational time that increases with the initial population size. We advocate that our



(a) F-measure



(b) Accuracy



(c) AUC

Fig. 4. The achieved performance metrics for NSGA-II compared to IBEA, SPEA2, NSGA-II, RS and GA algorithms.

BPEL defects prediction problem is not a real-time or time-critical problem requiring immediate training of the model. That is, a model can be built periodically (e.g., each week or month) and used afterward. While ML models are considered as fast-running models compared to MOEAs models which typically require a larger computational time, for this problem instance, we rather focus on the accuracy of the model. In fact, we computed the runtime for the NSGA-II and for RF while increasing the number of iterations as shown in Fig. 5. We observed that for one iteration, NSGA-II needs 45 s to build the model whereas RF requires only 0.01 s. By increasing the number of iterations to 31, the runtime of NSGA-II increased to achieve 2046 s while RF still maintained a fast-running model with 0.04 s. These results can be explained by the fact that NSGA-II did not discard any variables and preserved all the features for prediction. Hence, NSGA-II requires more runtime to build the model since it combines important features and tests many combinations, to arrive at even more useful features. However, for the RF algorithm, we consider the correlation between independent variables and we discard those having high correlation scores.

Table 6
The statistical F1 achieved by NSGA-II algorithm compared to NSGA-III, IBEA, SPEA2, GA, and RS.

Antipatterns	Statistical Test	F1					
		NSGA3	NSGA2	IBEA	SPEA	RS	GA
	Score	**82.9%**	**82.9%**	**82.9%**	**82.9%**	***34.0%***	***34.0%***
Missing_Input	p-values	o - - -++	- o - - ++	- - o - ++	- - - o++	++++o -	++++ - o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**89.9%**	**89.9%**	**89.9%**	**89.9%**	60.2%	61.9%
Redundant_Output	p-values	o - - -++	- o - - ++	- - o - ++	- - - o++	++++o -	++++ - o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**58.3%**	**58.3%**	50.0%	**58.3%**	32.5%	40.0%
Lost_Output	p-values	o - - -++	- o - - ++	- - o - ++	- - - o++	++++o -	++++-o
	effsize	oNNNMM	NoNNSS	NNoNSS	NNNoSS	SSSSoN	SSSSNo
	Score	66.7%	**70.8%**	66.7%	66.7%	34.3%	44.4%
Partially_portable	p-values	o - - -++	- o - - ++	- - o - ++	- - - o++	++++o -	++++ - o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	85.7%	**89.3%**	85.2%	87.3%	87.4%	87.4%
Non_portable	p-values	o - - - - -	- o - - - -	- - o - - -	- - - o- -	- - - - o-	—– o
	effsize	oNNNNN	NoNNNN	NNoNNN	NNNoNN	NNNNoN	NNNNNo

Table 7
The statistical Accuracy achieved by NSGA-II algorithm compared to NSGA-III, IBEA, SPEA2, GA, and RS.

Antipatterns	Statistical Test	Accuracy					
		NSGA3	NSGA2	IBEA	SPEA	RS	GA
	Score	**94.4%**	**94.4%**	**94.4%**	**94.4%**	22.2%	25.0%
Missing_Input	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**94.1%**	**91.3%**	**94.1%**	**91.3%**	58.7%	66.7%
Redundant_Output	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**88.9%**	**88.9%**	**88.9%**	**88.9%**	83.3%	86.1%
Lost_Output	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNMM	NoNNMM	NNoNMM	NNNoMM	MMMMoN	MMMMNo
	Score	91.7%	**91.7%**	**94.4%**	91.7%	75.0%	75.0%
Partially_portable	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	77.8%	**83.3%**	80.6%	**83.3%**	77.8%	77.8%
Non_portable	p-values	o—–	-o—-	–o—	—o–	—-o-	——o
	effsize	Onnnnn	NoNNNN	NNoNMM	NNNoNN	NNNNoN	NNNNNo

4.5.3. Results for RQ3 (Variability analysis)

Fig. 7 depicts the performance of NSGA-II by analyzing its variability towards the prediction of each particular BPEL defect type. We observe that NSGA-II is relatively stable towards the prediction of the considered defect. NSGA-II achieved more stable results of accuracy for all defect types with a score ranging from 83.3% to 94%. However, we observe that the detection of *Lost_Output* and *Non_portable* defects, exhibits a relatively higher variability in terms of AUC ranging from 73% to 92% and F1 ranging from 58% to 89%. Looking deeper into the dataset, we speculate that this variability can be due to the imbalanced nature of the used dataset especially for the *Lost_Output* and *Partially_portable* defects. We note that the stability aspect of NSGA-II shown in Fig. 7 cannot be generalized for the prediction of control flow or any other BPEL process defects.

4.5.4. Results for RQ4 (Features influence)

Table 10 report the most important features used to predict the presence of BPEL defects. For each feature, we calculate the number of optimal solutions in which the feature appears in the set of optimal solutions (*i.e.*, the Pareto front). For example, to avoid the *missing input*, the developer should consider IC (Input flow Complexity) which appeared in 100% among all the obtained optimal rules. Indeed, the smallest IC value has a higher risk to the occurrence of the *missing input* defect. Hence, BPEL developers

Table 8
The statistical AUC achieved by NSGA-II algorithm compared to NSGA-III, IBEA, SPEA2, GA, and RS.

Antipatterns	Statistical Test	AUC					
		NSGA3	NSGA2	IBEA	SPEA	RS	GA
	Score	**92.2%**	**92.2%**	**92.2%**	**92.2%**	***50.0%***	***50.0%***
Missing_Input	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**92.3%**	**92.3%**	**92.3%**	**92.3%**	55.9%	64.8%
Redundant_Output	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNLL	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo
	Score	**73.4%**	**73.4%**	**73.4%**	**73.4%**	64.6%	70.3%
Lost_Output	p-values	o—++	-o–++	–o-++	—o++	++++o-	+++++o
	effsize	oNNNSS	NoNNSS	NNoNSS	NNNoSS	SSSSoN	SSSSNo
	Score	79.5%	**86.6%**	**84.8%**	**86.6%**	63.8%	63.8%
Partially_portable	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNMM	NoNNLL	NNoNLL	NNNoLL	MLLMoN	MLLMNo
	Score	**75.0%**	**75.0%**	**75.0%**	**75.0%**	50.0%	50.0%
Non_portable	p-values	o—++	-o–++	–o-++	—o++	++++o-	++++-o
	effsize	oNNNNN	NoNNLL	NNoNLL	NNNoLL	LLLLoN	LLLLNo

Table 9
The achieved performance indicators values of each MOEAs in terms of Hypervolume (HV), Inverted Generational Distance (IGD) and Contribution (IC).

	HV				IGD				IC			
	NSGA-II	SPEA2	IBEA	NSGA-III	NSGA-II	SPEA2	IBEA	NSGA-III	NSGA-II	SPEA2	IBEA	NSGA-III
Missing_Input	**0.767**	0.757	0.762	0.752	**0.081**	0.085	**0.081**	**0.081**	**0.026**	0.024	0.025	**0.026**
Redundant_Output	0.95	0.95	**0.96**	0.95	**0.06**	**0.06**	**0.06**	**0.06**	**0.02**	**0.02**	**0.02**	**0.02**
Lost_Output	0.67	0.67	**0.70**	0.67	0.08	0.08	0.08	0.08	0.07	0.07	0.07	0.07
Non_Portable	0.534	0.549	0.553	0.538	0.101	0.100	0.101	0.101	0.042	0.043	0.042	0.043
Partially_Portable	0.69	0.69	0.73	0.67	0.14	0.13	0.14	0.13	0.05	0.05	0.05	0.05
Average	0.723	0.722	**0.740**	0.716	**0.092**	**0.092**	0.094	**0.092**	**0.043**	0.042	0.042	**0.043**

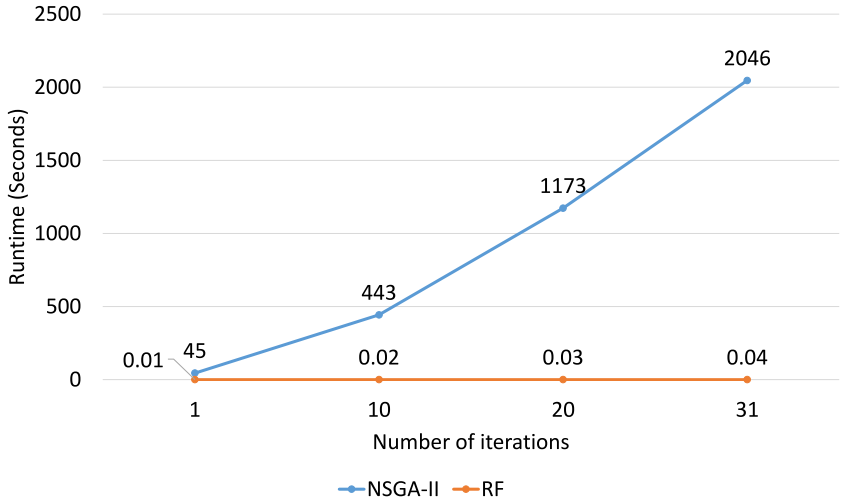
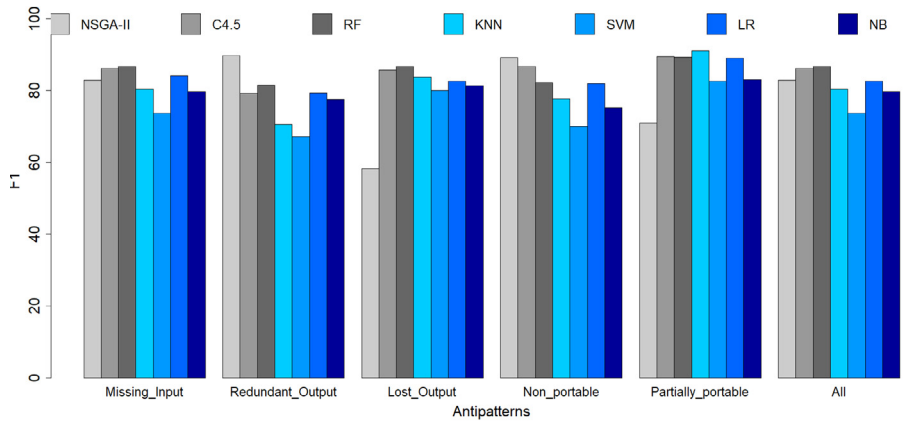
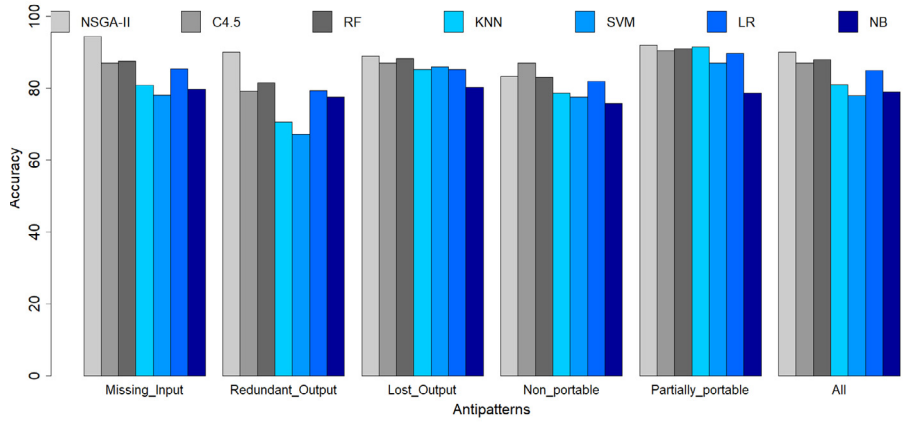


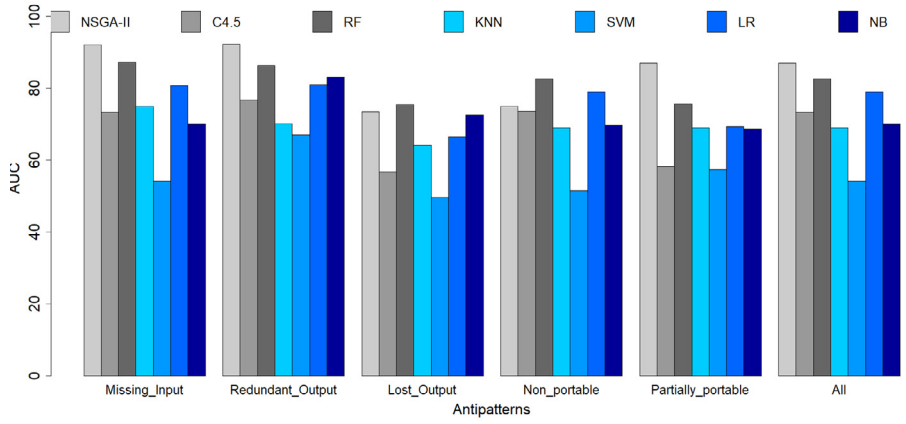
Fig. 5. The required runtime for NSGA-II and RF.



(a) F-measure



(b) Accuracy



(c) AUC

Fig. 6. The achieved performance metrics for NSGA-II compared to ML algorithms.

should pay more attention to the number of necessary input variables defined in the BPEL process. For the *redundant output*, we observe that the most influential features are related to OC (Output flow Complexity), F_out (Number of Output variables), and Sequentiality which appeared in 100%, 74% and 74% of the optimal rules, respectively. In fact, OC indicates the number of the required output variables defined in the BPEL process. Thus, BPEL developers should pay more attention to this metric because if OC is much less than 1, the BPEL process may have a redundant output defect. Similarly, for the *lost output*, many metrics are involved to detect this defect, where we observe that Ms, CNC, Diameter, and TS are the most important features that can mark the presence of lost output. For the *non-portable*, it is shown that Me (Weighed Elements metric) is the most dominant feature that

Table 10
The most influential features for each BPEL defect.

Features	*Missing_Input*	*Redundant_Output*	*Lost_Output*	*Non_Portable*	*Partially_Portable*
DFC	15%	**71%**	**85%**	60%	**71%**
Separability	10%	62%	81%	60%	58%
CFC	15%	67%	**85%**	67%	52%
FI	8%	65%	84%	63%	50%
dC_max	8%	65%	79%	60%	47%
PV	6%	61%	84%	60%	48%
Ms	10%	67%	**92%**	61%	**73%**
NOAC	6%	**71%**	84%	60%	48%
DeltaSJX	10%	59%	82%	61%	52%
FO	10%	65%	82%	66%	50%
CYC	8%	68%	82%	71%	50%
Edges	6%	61%	82%	65%	48%
F_(I+O)	16%	65%	81%	63%	48%
Depth	10%	61%	82%	66%	53%
F_out	8%	**74%**	81%	65%	45%
OC	13%	**100%**	84%	60%	60%
IC	**100%**	61%	81%	63%	56%
Diameter	8%	65%	**87%**	61%	53%
F_in	8%	65%	82%	63%	48%
NOA	8%	**71%**	77%	65%	50%
CH	10%	68%	81%	60%	50%
Sequentiality	8%	**74%**	**85%**	65%	50%
dC_avg	8%	61%	**85%**	61%	45%
CW	6%	68%	82%	61%	47%
PD	10%	60%	84%	65%	56%
Ma	6%	**71%**	81%	58%	**75%**
Mb	6%	65%	82%	**81%**	62%
Me	6%	68%	84%	**90%**	**94%**
CNC	6%	61%	**87%**	63%	58%
PL	6%	61%	82%	61%	45%
TS	8%	65%	**88%**	61%	52%

Table 11
The subset of the most influential metrics for each BPEL defect.

NSGA-II	Process metrics
Missing_Input	IC, CFC
Redundant_Output	DFC, OC, NOAC, NOA, F_Out, Sequentiality
Lost_Output	DFC, CFC, MS, Diameter, Sequentiality, dC_avg, CNC, TS
Non_Portable	Me
Partially_Portable	Ma, Ms, Me

can characterize such defect. However, for *partially portable*, Me, Ma, and Ms are reported as the most influential features that can characterize such defect.

To further assess the performance of NSGA-II towards the prediction of BPEL defects, we trained NSGA-II using only the most influential metrics for each defect type. Table 11 illustrates the most influential metrics responsible for the detection of data flow defects and portability defects. This subset of metrics may be available as pointed out by Song et al. (2021) or could be computed from available BPEL code. Table 12 reports the obtained results of NSGA-II using the most influential metrics compared to the results using all the metrics. Overall, observe that the results are either the same or better than when using all the features, in terms of accuracy, F1, and AUC for all defects except for the Missing_Input which achieves the same results. For the redundant_output, NSGA-II improves the accuracy and the AUC scores by 3% and 2%, respectively. For the lost_output, we note that NSGA-II increases the F1 and the AUC by more than 3% and 4% respectively. For the non_portable, we observe an increase of F1, accuracy, and AUC by 4%, 5%, and 18%, respectively, which justifies that Me is the most influential metric responsible for the non-portability of the process. For the partially_portable, we observe an increase of F1, accuracy, and AUC by 20%, 6%, and 19%, respectively.

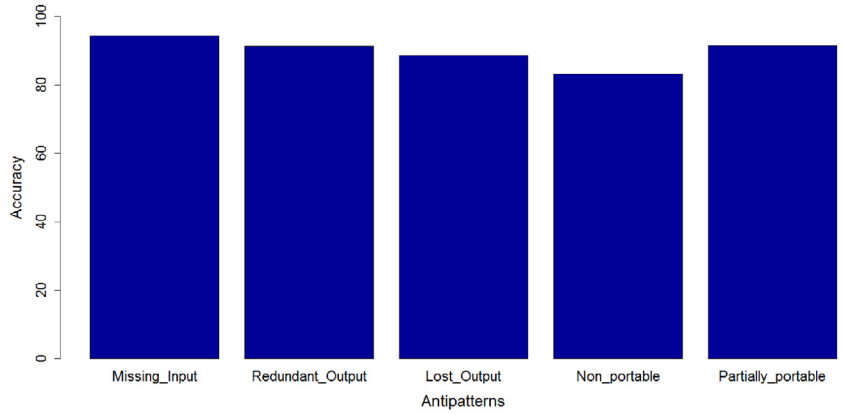
Based on the obtained results, we also observe that when we reduce the search space using only the most important features, NSGA-II performs better. This performance justifies that influential features help NSGA-II to converge faster and better. Although NSGA-II achieves promising accuracy compared to ML and the computation time is decreased slightly, ML models are still the fast-running models compared to NSGA-II models. However, it is worth noting we note that the BPEL defects prediction problem is not a real-time or time-critical problem requiring immediate training of the model as practitioners can run it periodically (e.g., on a weekly or monthly basis as their BPEL files evolve with new customers' needs).

4.5.5. Results for RQ5 (Threshold values analysis)

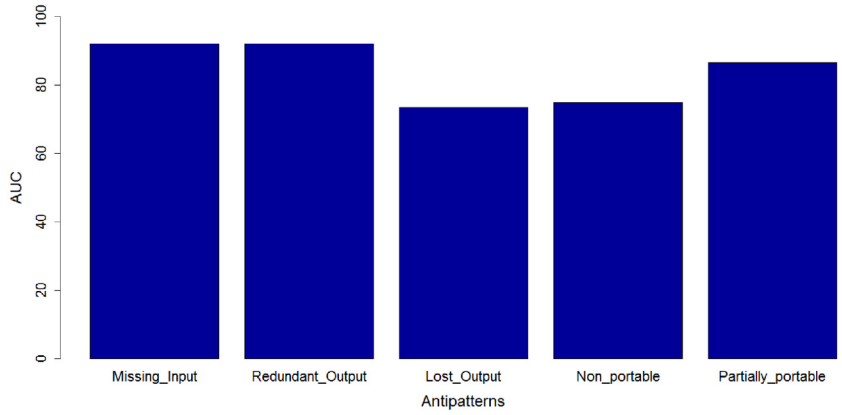
For each defect type, we provide an example of the obtained prediction rule as well as the distribution of the threshold values for the top-5 most influential features.

Missing input. Figs. 13 and 8 depict an example of a prediction rule for each missing input defect obtained in the Pareto optimal solutions generated by NSGA-II, and the distribution of the threshold values for the top-5 features, respectively. As explained in Section 3, a prediction rule represents a combination of a set of process metrics and their relative threshold values using logical operators. We observe from both figures that IC is a critical metric when its threshold is less than or equal to 0.93 (in most prediction rules) or less than or equal to 0.92 for the rest of the rules. The remaining metrics including DFC, F_(I+O), OC, and CFC contribute to the prediction of missing input with a threshold value less than or equal to 0.88, 321, 0.93, and 4629052, respectively.

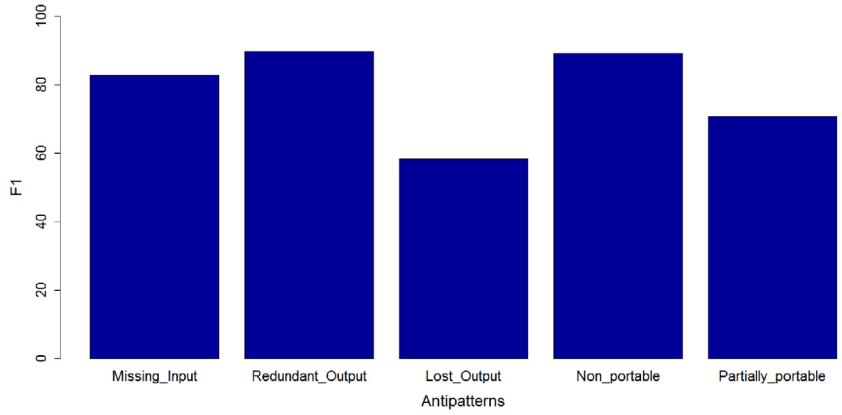
Redundant input. Figs. 14 and 9 report an example of an obtained prediction rule for the redundant input defect, and the distribution of the threshold values for the top-5 features, respectively. We observe that the OC metric is associated with a threshold value less or equal to 0.98 in most of the generated rules (less or equal to 0.96 in some other rules). The other metrics including DFC, F_out, NOAC, and Sequentially have also



(a) Accuracy



(b) AUC



(c) F-measure

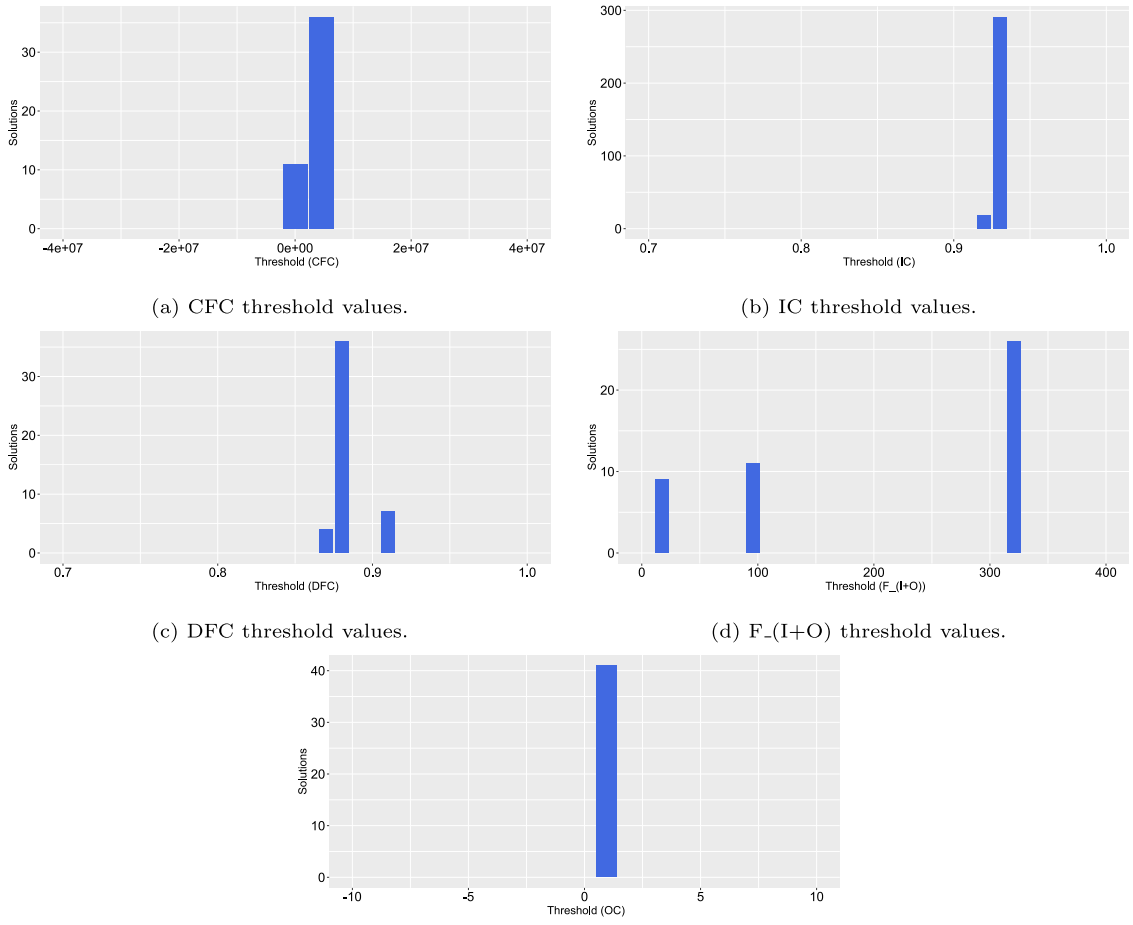
Fig. 7. The accuracy, AUC, and F-measure scores achieved by NSGA-II for each defect type.

importance since they appeared in more than 70% of the rules (as shown in RQ4). Hence, NOAC is a critical metric when its threshold is greater or equal to 19. DFC, F_out, and Sequentially are critical metrics when their threshold values are less than or equal to 0.98, 13, and 0.74, respectively.

Lost output. Figs. 15 and 10 report an example of an obtained prediction rule for the lost output defect, and the distribution of the threshold values for the top-5 features, respectively. The metric Ms is among the most influential metrics (as shown in RQ4) with a threshold value ranging from 0.94 to 0.97, where 0.97 is the dominant threshold value in the most generated rules. Furthermore, TS is associated with threshold values greater or equal to 4. For the features DFC, Diameter, and CNC, the threshold values are ranging from 0.96 to 0.98 for DFC, 25 to 35 for Diameter, and 1.06 to 1.17 for CNC. It is worth noting that based on the results from Fig. 10, DFC is a critical metric when its threshold is less than or equal to 0.98. Diameter and CNC are critical metrics when their threshold values are greater or equal to 34 and 1.17, respectively.

Table 12
Comparison between NSGA-II results using all metrics and NSGA-II results using the most influential metrics.

NSGA-II	Process metrics	F1	Accuracy	AUC
Missing_Input	All metrics	82.9%	94.4%	92.2%
	IC, CFC	**82.9%**	**94.4%**	**92.2%**
Redundant_Output	All metrics	89.9%	91.3%	92.3%
	DFC, OC, NOAC, NOA, F_Out, Sequentiality	**90.9%**	**94.1%**	**94.5%**
Lost_Output	All metrics	58.3%	88.9%	73.4%
	DFC, CFC, MS, Diameter, Sequentiality, dC_avg, CNC, TS	**61.9%**	88.9%	**77.2%**
Non_Portable	All metrics	89.3%	83.3%	75.0%
	Me	**93.1%**	**88.9%**	**93.5%**
Partially_Portable	All metrics	70.8%	91.7%	79.5%
	Ma, Ms, Me	**90.0%**	**97.2%**	**98.4%**



(a) CFC threshold values.

(b) IC threshold values.

(c) DFC threshold values.

(d) F_(I+O) threshold values.

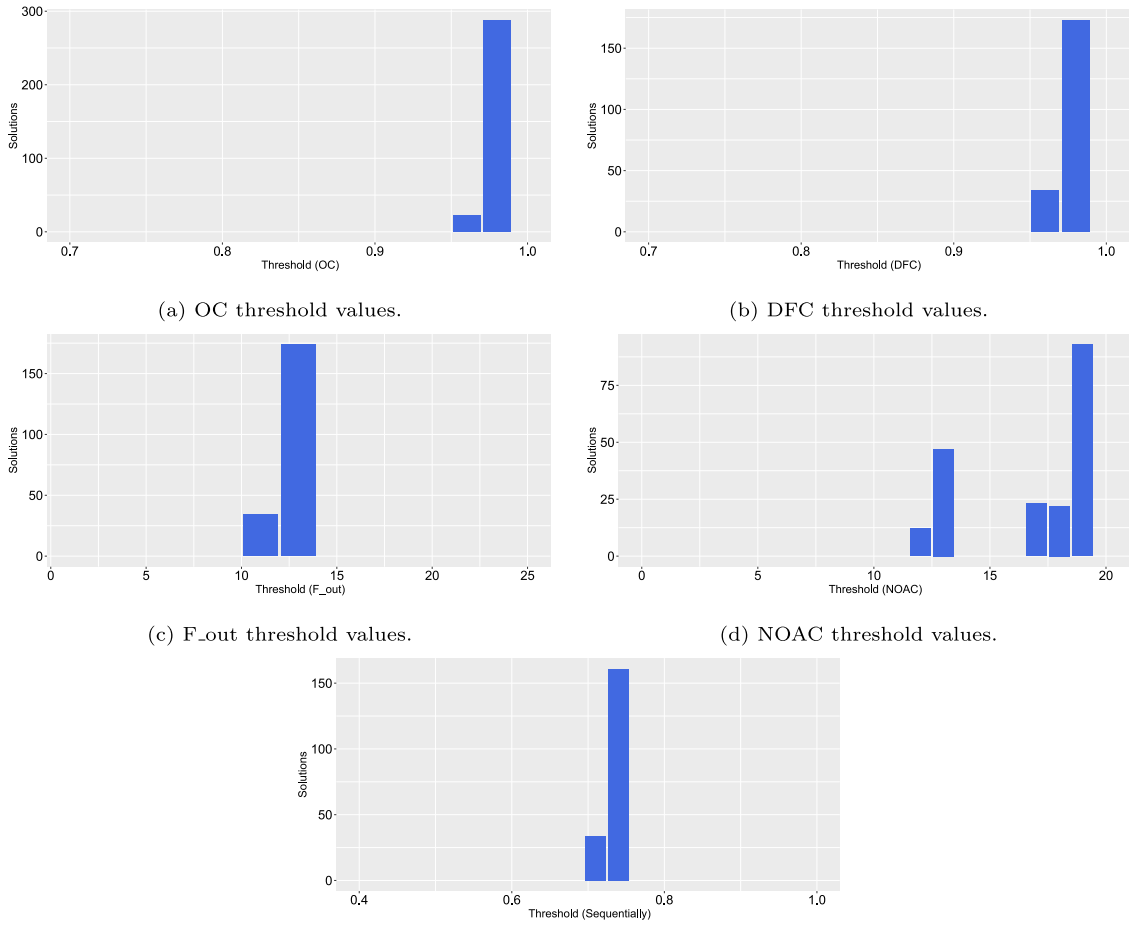
(e) OC threshold values.

Fig. 8. Distribution of the threshold values for the Top-5 influential metrics in Missing Input defect.

Non-portable BPEL. Figs. 16 and 11 report an example of an obtained prediction rule for the non-portable BPEL defect, and the distribution of the threshold values for the top-5 features, respectively. Me is the most important feature that appeared more than 90% in the generated rules with a threshold value that is less than or equal to 0.96 in most of the generated rules. The remaining metrics including CFC, CYC, Mb, and PD are also associated with various threshold values that are less than or equal to 240, 0.25, 0.99, and 10.5, respectively.

Partially portable BPEL. Figs. 17 and 12 report an example of an obtained prediction rule for the partially portable BPEL defect, and the distribution of the threshold values for the top-5 features, respectively. We observe that Me is the dominant metric that appeared more than 90% in the prediction rules with a threshold value that is less than or equal to 0.96 in most of the generated rules. The remaining metrics include DFC, Ma, Ms, and Mb are also critical metrics when they provide a threshold value less than or equal to 0.89, 0.98, 0.97, and 0.83, respectively.

Overall, the distributions of the threshold values associated with each feature provide valuable knowledge that can help



(a) OC threshold values.

(b) DFC threshold values.

(c) F_out threshold values.

(d) NOAC threshold values.

(e) Sequentially threshold values.

Fig. 9. Distribution of the threshold values for the Top-5 influential metrics in Redundant Output defect.

practitioners avoid the presence of such defects. For instance, during the initial design or the evolution of a given BPEL file, when a specific metric approaches a critical threshold, developers can take the right actions to prevent the emergence of defects.

5. Discussion

In this section, we discuss the implications of our results for business process developers, researchers, and the application scenarios of BPEL defects prediction.

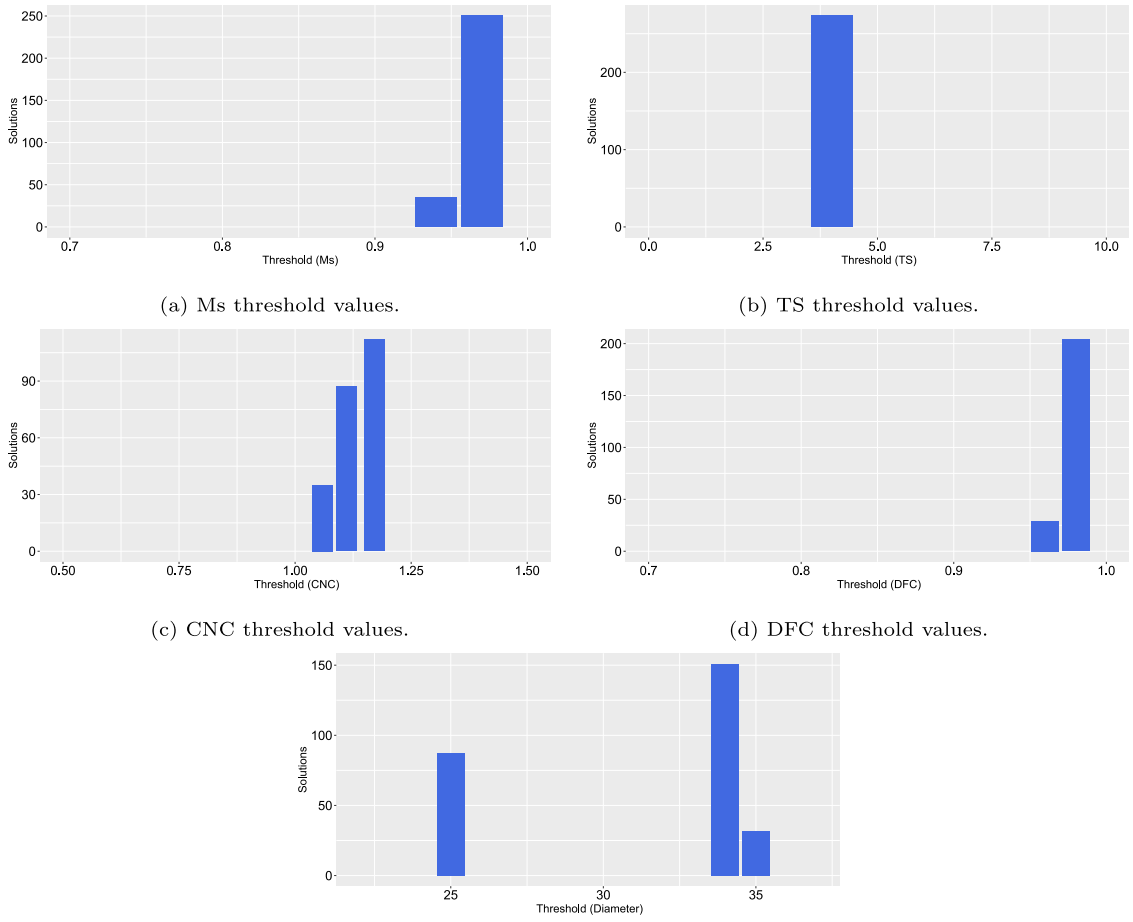
5.1. For business process developers

We can support the BPEL developers to avoid additional maintenance costs and effort in the future. Our NSGA-II approach aims at supporting designers and developers in the prediction of BPEL defects as early as possible. In fact, based on the obtained results from the prediction rules, BPEL designers can reduce additional efforts and costs when identifying potential defects in their BPEL processes. Indeed, our prediction rules aim not only at predicting data flow or portability defects but also at characterizing the specific type of defect that occurs, such as the *missing input* or *redundant output* from the data flow defect category. Furthermore, our approach serves as a decision support for BPEL developers regarding process portability. It can help the developers to decide whether they invest in rewriting the process partially or from scratch. This support may help avoid unnecessary efforts and costs of rewriting the whole process or simply selecting another fully portable process.

In addition, the provided prediction rules give more insights about the most important features and their related thresholds that can be responsible for introducing defects in the process. For example, the input flow complexity metric (IC) is strongly correlated with the presence of missing input defects when its threshold value is less than 0.93. This metric represents the proportion of the required input variables that have been defined in a BPEL process. Moreover, the output complexity metric (OC) plays a crucial role to predict redundant output defects with a threshold value of less than 0.98. This metric estimates the proportion of output variables that can be used in a BPEL process. Interestingly, the lost output involves several metrics to be combined including Ms, TS, Diameter, DFC, and CNC. On the other hand, to support developers in deciding whether the business process is portable or not, (Me) is strongly correlated to the partially portable and non-portable defects with a threshold value of less than 0.96. Hence, based on the obtained results developers should pay more attention to these metrics and their threshold values to avoid data flow and portability defects.

5.2. For researchers

Researchers can gain insights to improve business process quality based on complexity metrics. Since business processes may appear as black boxes to BPEL users, the research community can explore process complexity metrics that are available for these black boxes to predict business process defects. Indeed, recent works provide information about the correlation



(a) Ms threshold values.

(b) TS threshold values.

(c) CNC threshold values.

(d) DFC threshold values.

(e) Diameter threshold values.

Fig. 10. Distribution of the threshold values for the Top-5 influential metrics in Lost Output defect.

between process complexity metrics and process defects (Song et al., 2021). However, there is no information about the appropriate threshold that characterizes these metrics to detect such defects. Hence, our search-based approach addresses this problem and allows to consider threshold values for each metric to characterize a defect. Furthermore, researchers can use such knowledge to develop dedicated BPEL defect correction tools based on these metrics and their appropriate thresholds. Such a correction tool would be less time-consuming by checking only the considered metrics provided by our approach.

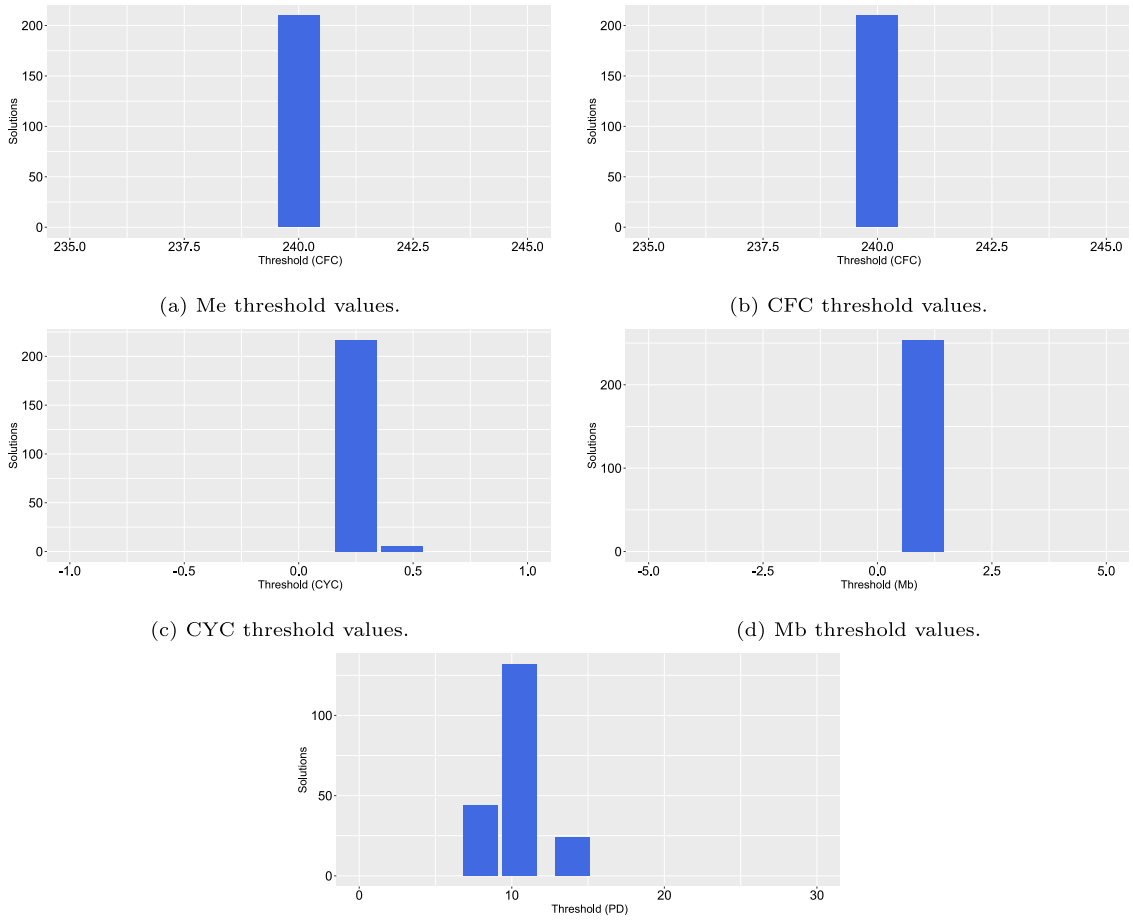
Business process selection criteria for composing new applications. With the growth of available SBPs, selecting relevant SBPs for composing new applications satisfying user requirements is a challenging task. In fact, QoS is regarded as the most important criterion to distinguish among functionally similar SBPs. However, QoS can be an insufficient criterion to build a robust and agile application. Hence, our approach can be considered as a pre-filter step to the selection process. In fact, based on the information provided by the rules violation, we can discard business processes that involve defects taking into consideration the QoS constraints. Hence, the rest of the business processes in the pool will be the eligible candidates to the selection process.

5.3. Application scenarios of BPEL defects prediction

Our BPEL prediction method can be applied in a range of scenarios BPEL prediction method can be applied in a range of scenarios:

- Process quality assurance: BPEL prediction methods can be used to periodically assess the quality of BPEL-based business processes. By identifying potential defects, our method can help to pinpoint potential defects to fix them more efficiently as soon as they are introduced in the process (Song et al., 2021).
- Process verification: BPEL prediction methods can be used to verify the correctness of BPEL-based business processes. This can help to ensure that the data flow and control flow in the process are correct and that there are no defects (Hua et al., 2014).
- Process optimization: BPEL prediction methods can be used to optimize the design of BPEL-based business processes. By identifying potential defects, the method can help to simplify the process and to reduce the risk of errors (Song et al., 2021).

Moreover, we believe that combining BPEL prediction methods and BPEL program analysis can be a powerful approach to improve the accuracy and effectiveness of BPEL analysis. In fact, BPEL defect prediction methods can be used to guide program analysis. By predicting the likelihood of defects in the BPEL process, these methods can help analysts prioritize their analysis efforts on the areas that are most likely to be problematic like the data flow. This can save time and resources, and improve the effectiveness of the analysis. Hence, this can help to focus the analysis on the most important areas and to reduce the amount of analysis needed.



(a) Me threshold values.

(b) CFC threshold values.

(c) CYC threshold values.

(d) Mb threshold values.

(e) PD threshold values.

Fig. 11. Distribution of the threshold values for the Top-5 influential metrics in Non portable defect.

6. Threats to validity

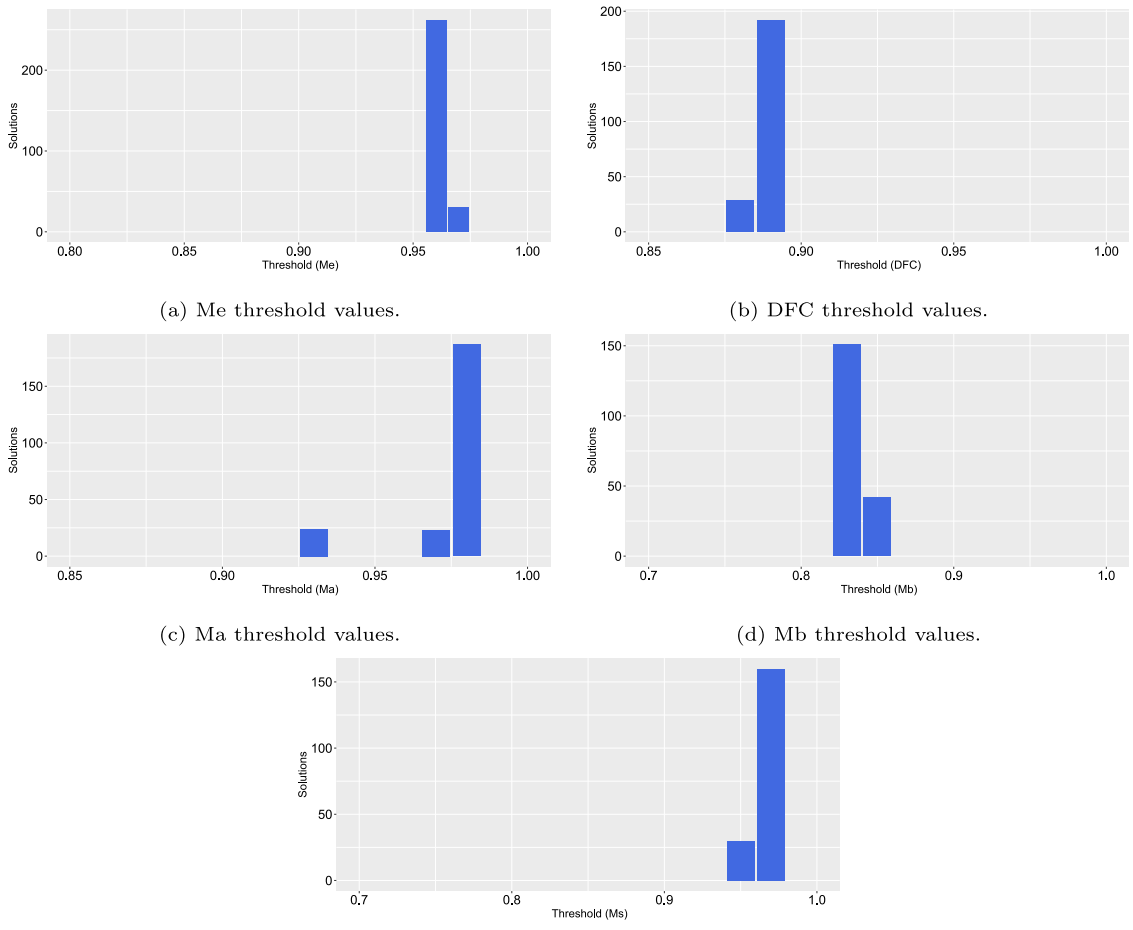
Different threats may influence our empirical study. We mainly identify threats to external, internal, and construct validity.

Threats to external validity. A potential threat to external validity can be related to the number of BPEL process defects that are employed in our empirical evaluation. We considered five types of defects which constitute a wide representative group of common defects that are studied in the literature (Song et al., 2021; Lenhard and Wirtz, 2013a,b; Palma et al., 2013). Furthermore, a possible external threat to validity could be associated with the size of the used dataset which contains only 178 real-world BPEL processes. Furthermore, a possible external threat to validity could be associated with the size of the processes in our dataset as we cannot generalize our findings to other datasets with more complex processes (involving process model subgraphs, e.g., deadlock) or for large or complex BPEL processes (with graphs of $|N| > 1000$).To address this issue, we plan in the future to extend our dataset with additional more complex BPEL processes and additional BPEL defects including control flow defects. Moreover, another threat to validity could be related to the used classification algorithms, where we chose to compare our approach to different MOEAs, random search, and six different supervised learning algorithms as commonly used algorithms for BPEL defects prediction (Song et al., 2021).

Threats to internal validity. The validity of our experimental results could be influenced by the stochastic nature of the optimization algorithms (Harman et al., 2012; Arcuri and Briand, 2011). To address this potential issue, we conducted 31 runs of each algorithm and select the median value in each iteration using statistical tests with Wilcoxon and effect size cliff's delta. Another threat to validity can be related to errors in our experiments. Although, we have verified our experiments and the processes belonging to a widely used dataset, errors that we did not notice could exist. Internal threats to validity could be related also to the technique of training and test sets split. To address this problem, we adopt the K-fold cross-validation technique with $K = 10$ which is widely employed in machine learning. As future action, we plan to analyze the impact of changing the number of K folders on the performance of our approach.

Threats to construct validity. Threats to construct validity could be associated to the set of BPEL engines under consideration to compute the BPEL portability metrics. The used benchmark includes ActiveBPEL5.0.2,[4] bpel-g 5.3,[5] OpenESB Sun BPEL Service Engine 2.3.1,[6] Orchestra 4.9.0,[7] Petals EasyBPEL 4.1.0,[8] WSO2

4 https://osdn.net/projects/sfnet_activebpel502/
5 https://code.google.com/archive/p/bpel-g/
6 http://openesb-dev.org/
7 http://orchestra.ow2.org/
8 https://petals.linagora.com/



(a) Me threshold values.

(b) DFC threshold values.

(c) Ma threshold values.

(d) Mb threshold values.

(e) Ms threshold values.

Fig. 12. Distribution of the threshold values for the Top-5 influential metrics in the partially portable defect.

Business Process Server 3.1.0[9] and two proprietary engines from major middle-ware vendors, which represent the most common BPEL engines (Lenhard and Wirtz, 2016). While multiple platforms and middleware systems exist, our BPEL processes were executed using Apache ODE 1.3.6,[10] which is a common runtime environment that can help ensure BPEL portability through computing accurate portability metrics. Apache ODE is an open-source BPEL engine that supports multiple platforms and middleware systems (Harrer et al., 2012). Hence, while we believe there is little bias towards the use of Apache ODE, other runtime environments can be explored. Moreover, to avoid the mono-metric bias for the portability measurements, we used four metrics that represent the standard metrics to evaluate BPEL portability (Lenhard and Wirtz, 2013b, 2016).

7. Related work

In this section, we outline the related works that can be divided into two main categories (1) Web service design anti-patterns, and (2) Service-based Business Processes defects.

7.1. Web service anti-patterns

Web service anti-pattern detection has received growing attention in the literature over the last few years. A significant research effort was addressed to the detection and correction of service design problems and anti-patterns. Dudney et al. (2003) have proposed the first book that provides informal definitions of recurrent Web service anti-patterns. Král and Zemlicka (2009) reports symptoms characterizing seven common SOA anti-patterns that violate standard SOA principles. Similarly, Rotem-Gal-Oz (Anon, 2012) described the symptoms of a set of Web service anti-patterns. Palma et al. (2014) also proposed a rule-based approach based on service interface metrics to detect the key symptoms characterizing an anti-pattern.

More recently, Ouni et al. (2017c, 2015a) proposed an automated approach to detect Web service anti-patterns using genetic programming (GP). The proposed approach provides Web service defects detection rules learned from existing instances of each defect type and combined both service design metrics and threshold values. Later, Ouni et al. (2017a) proposed an SVM-based approach for Web service design defects detection using simulated annealing (SA) technique to set the best parameters for SVM. Furthermore, Rodriguez et al. (2013, 2010) and Coscia et al. (2011) proposed a set of directives that should be taken into consideration by service providers to avoid bad practices while writing WSDLs. Recently, Saidani et al. (2020) proposed

9 https://wso2.com/products/business-process-server/
10 http://ode.apache.org/

Table 13
Existing approaches for business process defects analysis.

Reference	Approach	Business Process defects	Type of the approach
Van der Aalst (1998)	Petri net analysis techniques	Control flow	White box
Sadiq and Orlowska (2000)	Graph reduction approach for analyzing process models	Control flow	White box
Trčka et al. (2009)	Petri net model checking based on temporal-logic formulation	Data flow	White box
Meda et al. (2010)	Graph-based data flow detection	Data flow	White box
Roy et al. (2014)	Petri analysis and graph-theoretic techniques	Control flow	White box
Palma et al. (2013)	BPEL inference rules	1- Control flow 2- Data flow	White box
Lenhard and Wirtz (2013a)	Portability profile technique to identify portability issues	BPEL code portability	White box
Lenhard and Wirtz (2016)	Measuring the portability of executable service-oriented process	BPEL code portability	White box
Kabbaj et al. (2015)	Data flow anti-patterns verification	Data flow	White box
Song et al. (2021)	1- Reaching definitions based algorithms to detect data flow defects in BPEL	Data flow	White box
	2- Machine learning based approach to predict data flow defects using process metrics		Black/Gray box
Daagi et al. 2023 (this study)	NSGA-II based approach to predict data flow and portability defects using process metrics	1- Data flow 2- Process code Portability	Gray box

an approach using a multi-label classifier chair to enhance the detection accuracy while handling the interleaving characteristics of various anti-patterns symptoms.

Motivated by the harmful effects of these anti-patterns, a growing number of studies have been dedicated to their correction. One of the first attempts to fix Web service anti-patterns was proposed by Athanasopoulos et al. (2015) who presented a Web service interface decomposition approach to correct the multi-service anti-pattern using a greedy algorithm to create new interfaces based on cohesive operations. Hirsch et al. (2018) proposed a code-first approach and tool support to develop discoverable Web services while spotting and removing Web service anti-patterns. Hirsch et al. reported that bad Web service design choices and poor coding practices are responsible for Web service discoverability issues due to WSDL textual and structural information problems that should be detected. Ouni et al. (2016, 2018) proposed an automated graph-based approach called Service Interface Modularization (SIM) to correct the multi-service anti-pattern. The approach consists of decomposing large service interfaces exposing a huge number of non-cohesive and semantically unrelated operations into smaller cohesive interfaces.

To do this, the authors captured structural and semantic relationships between operations to guide the decomposition process where strongly related operations form a new interface. However, the main limitations of the study could be related to the lack of coupling between interfaces. To address this challenge, Daagi et al. (2017) proposed an automated approach to enhance the interface design quality using Formal Concept Analysis (FCA) based on the computation of three cohesion measures. These metrics allow capturing the hidden relationship between operations through the generated formal concepts. Hence, new cohesive and loosely coupled service interfaces are built. Recently, Boukharata et al. proposed an automated multi-objective search-based approach using the non-dominated sorting genetic algorithm (NSGA-II) to improve Web service interfaces modularity by searching for the optimal decomposition of service interfaces through optimal cohesion and coupling (Boukharata et al., 2019).

However, one of the principal limits of the existent approaches is related to the non-consideration of the developer's point of view that may conduct to less relevant services interfaces. To address this issue, Wang et al. proposed an interactive search-based approach to enhance the design quality of Web service interfaces while taking into consideration the developer in the process (Wang et al., 2021).

7.2. Service-based business processes defects

Over the last two decades, business process design quality has gained increasing attention from literature, where Service-based Business Processes are error-prone through frequent changes that may lead to introducing process defects. Various research works addressed this problem from three perspectives, control flow defects, data flow defects, and BPEL portability defects as summarized in Table 13. In the following, we review the main research works from each perspective.

Control flow defects. Ensuring the soundness of control flow business processes is very important assuming a control flow defect detection firstly. Sadiq and Orlowska (2000) proposed a process model reduction technique that detects two structural anti-patterns, deadlock and lack of synchronization. However, it cannot determine the reported anti-patterns' position in the business process. Van et al. also addressed the detection of these two defects using petri analysis techniques to verify the correctness of workflows (Van der Aalst, 1998). Laue and Awad (2010), proposed a BPMN-Q queries-based approach that can not only detect six control flow defects in BPMN business processes but also locate the main part responsible for the defect.

In Roy et al. (2014), petri analysis and graph-theoretic techniques are used to analyze and detect syntactic and structural defects occurring in industrial processes, where dead activities and several edges between them refer to the most prevalent anti-patterns. However, the proposed approach supports only BPMN

as a process modeling language for the considered business processes. Palma et al. (2013) defined six control flow defects using classical inference rules. However, these textual rules should be applied to a simplified business process after a transformation step. Moreover, it is a time-consuming and non-trivial task to manually inspect and validate these detection rules that do not consider BPEL complexity metrics.

Data flow defects. Many research works addressed the detection and correction of process data flow defects. Trčka et al. (2009) proposed a temporal-logic formulation of nine data flow defects that can be later supported by model checkers. Similarly to Trčka et al. (2009), Meda et al. (2010) suggested a graph-based data flow detection approach in both nested and unstructured workflows. Moreover, Kabbaj et al. (2015) proposed a data flow defects verification approach during process modeling. Furthermore, Palma et al. (2013) defined classical inference rules that detect dangling inputs or outputs in BPEL business process. However, the main limit relies on the detection rules, which are described in a textual format with no consideration of the BPEL complexity metrics. Hence, the application of these rules requires additional effort for the BPEL transformation step to simplify a complex business process and second an implementation effort because it is difficult to manually inspect and validate these rules. In the same field, Song et al. (2010) defined criteria for process adaptation that allows to preserve data flow correctness and proposed an empirical study on data flow defects in business processes. However, the main limitation of Song's study is how to find the best threshold value for each metric since the detection process manages quantitative information to support developers avoiding such anti-patterns. Indeed, our prediction rules give more insights about the most important features and their related thresholds that can be responsible for introducing defects in the process. Furthermore, Song et al. (2021) studied the correlation between metrics and data flow defects to improve the classification performance of machine learning algorithms. However, in our work, we used MOEAs where the model is trained on all features without any multicollinearity process. In fact, multicollinearity is the presence of a high correlation between two or more independent variables which makes the coefficients of the variables unstable and therefore difficult to interpret. Although the presence of multicollinearity caused the model to give results that are both misleading and confusing (Kim, 2019), NSGA-II did not discard any variables and preserved all the features for prediction. Hence, NSGA-II automatically combines important features and tests many combinations, to find the optimal features combinations. Moreover, we considered new business process defects related to the process portability issue which are: Non-Portable and Partially Portable defects that are not considered in Song's study.

BPEL process portability. Although control flow and data flow defects are the most recurrent process defects, portability defects have gained increasing attention from the literature. Indeed, the first attempts to quantify and detect portability anti-patterns are reported by Lenhard and Wirtz (2013a,b, 2016). In the first contribution, Lenhard and Wirtz (2013a) proposed a portability profile technique to identify portability issues in BPEL code. The BPEL portability profile defines test assertions that describe a normative requirement of the profile that should be respected. The assertions are based on data of an analysis of the BPEL conformance of a large set of BPEL engines. The output of the conformance test is a dataset indicating the support for every feature of the language specification by each engine. Next, Lenhard and Wirtz (2013b) proposed a set of metrics that allows to measure and quantify the BPEL process portability levels based on a set of BPEL engines. Recently, Lenhard and Wirtz (2016) proposed an empirical evaluation of the defined portability metrics from a theoretical and a practical aspect. From a theoretical aspect, the metrics are normalized complexity metrics satisfying the properties of non-negativity, symmetry, and monotonicity, but not null value and additivity. On the other hand, the practical evaluation reports that the metrics are resilient to moderate changes in the data underlying the computation. However, the main limitation of these studies is to find the best threshold value for each used metric that supports developers whether they invest in rewriting the process partially or from scratch.

8. Conclusion

In this paper, we proposed a new automated BPEL defects prediction approach. Our approach consists of generating defect prediction rules for each BPEL defect type based on the Non-dominated Sorting Genetic Algorithm II (NSGA-II). The best candidate prediction rule consists of combinations of BPEL complexity metrics and their threshold values. We assessed the effectiveness of our approach on a benchmark of 178 existing BPEL processes that belong to different domains and five BPEL defect types. The experimental study of the achieved results reports the effectiveness of our search-based approach to identify all the existing defects compared to six supervised learning algorithms, three common multi-objective optimization algorithms with 91% of accuracy.

As future work, we plan to extend our experimental study with additional BPEL processes and other emerging process issues such as control flow defects to better generalize our results. We also plan to consider additional portability metrics related to BPEL code level and additional BPEL engines to assess the effect on the prediction results. Furthermore, it would be interesting to investigate the trade-off between the accuracy of the BPEL defect prediction models and their execution runtime.

CRediT authorship contribution statement

Marwa Daaji: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Investigation, Writing – original draft. **Ali Ouni:** Conceptualization, Methodology, Validation, Supervision, Resources, Writing – review & editing, Funding acquisition, Project administration. **Mohamed Mohsen Gammoudi:** Methodology, Validation, Supervision. **Salah Bouktif:** Methodology, Validation, Writing – review & editing. **Mohamed Wiem Mkaouer:** Methodology, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We shared the associated data in a replication package.

Acknowledgment

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), grant RGPIN-2018-05960.

Appendix A. BPEL defects prediction models

See Figs. 13–17.

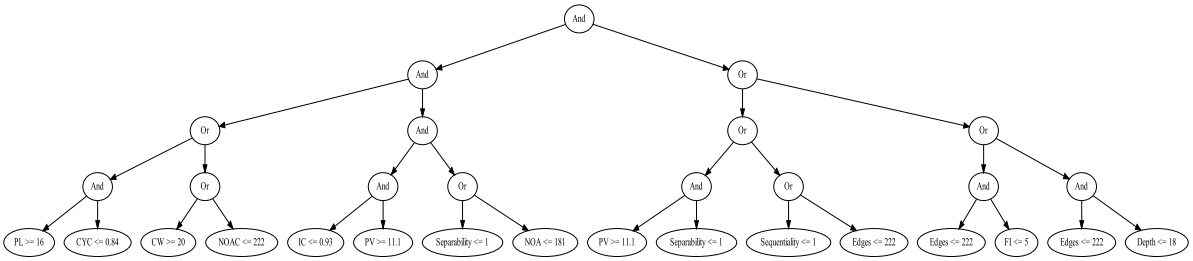


Fig. 13. Prediction rule from the Pareto optimal solutions for the missing input defect.

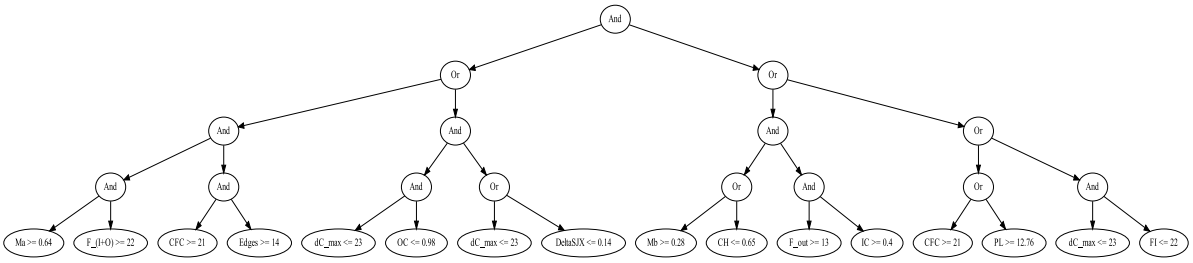


Fig. 14. Prediction rule from the Pareto optimal solutions for the redundant output defect.

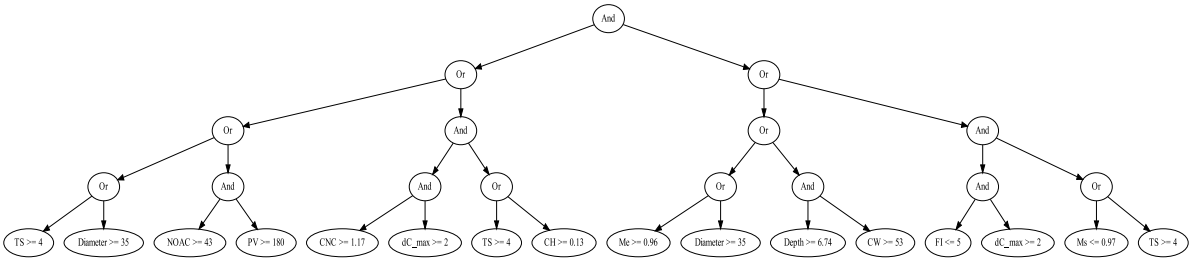


Fig. 15. Prediction rule from the Pareto optimal solutions for the lost output defect.

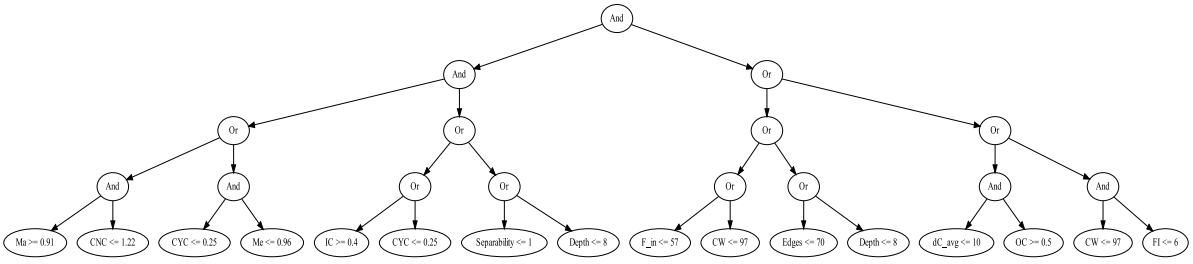


Fig. 16. Prediction rule from the Pareto optimal solutions for the non-portable defect.

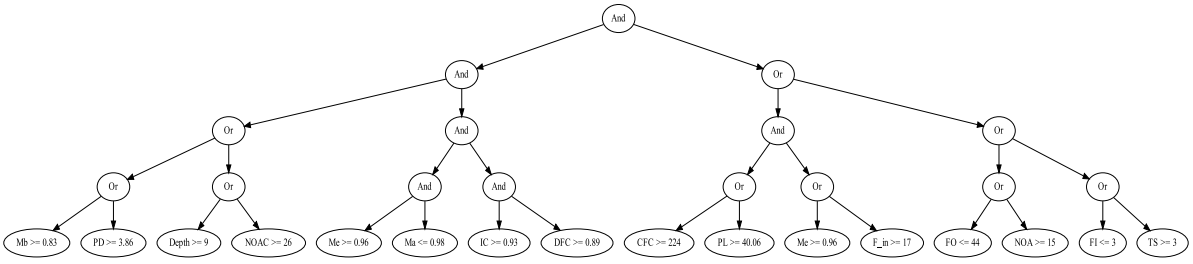


Fig. 17. Prediction rule from the Pareto optimal solutions for the partially portable defect.

References

Almarimi, N., Ouni, A., Chouchen, M., Mkaouer, M.W., 2022. Improving the detection of community smells through socio-technical and sentiment analysis. J. Software: Evol. Process e2505. http://dx.doi.org/10.1002/smr.2505.

Anon, 2012. SOA Patterns. Manning Shelter Island.

Anon, 2021. Dataset used in the study. Available at : https://github.com/stilabets/BPELantipatterns.

Arcuri, A., Briand, L., 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: 2011 33rd International Conference on Software Engineering. ICSE, pp. 1–10. http://dx.doi.org/10.1145/1985793.1985795.

Arcuri, A., Briand, L., 2014. A Hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw. Test. Verif. Reliab. 24 (3), 219–250. http://dx.doi.org/10.1002/stvr.1486.

Arkin, A., Askary, S., Bloch, B., Curbera, F., Goland, Y., Kartha, N., Liu, C.K., Thatte, S., Yendluri, P., Yiu, A., 2005. Web services business process execution language version 2.0. WS-BPEL TC OASIS.

Athanasopoulos, D., Zarras, A.V., Miskos, G., Issarny, V., Vassiliadis, P., 2015. Cohesion-driven decomposition of service interfaces without access to source code. IEEE Trans. Serv. Comput. 8 (4), 550–562. http://dx.doi.org/10.1109/TSC.2014.2310195.

Berry, W.D., Feldman, S., Stanley Feldman, D., 1985. Multiple Regression in Practice, no. 50. Sage, http://dx.doi.org/10.4135/9781412985208.

Bezerra, L.C., López-Ibánez, M., Stützle, T., 2017. An empirical assessment of the properties of inverted generational distance on multi-and many-objective optimization. In: International Conference on Evolutionary Multi-Criterion Optimization. Springer, pp. 31–45. http://dx.doi.org/10.1007/978-3-319-54157-0_3.

Blank, J., Deb, K., 2020. Pymoo: Multi-objective optimization in Python. IEEE Access 8, 89497–89509. http://dx.doi.org/10.1109/ACCESS.2020.2990567.

Boukharata, S., Ouni, A., Kessentini, M., Bouktif, S., Wang, H., 2019. Improving web service interfaces modularity using multi-objective optimization. Autom. Software Eng. 26 (2), 275–312. http://dx.doi.org/10.1007/s10515-019-00256-4.

Brockhoff, D., Friedrich, T., Neumann, F., 2008. Analyzing hypervolume indicator based algorithms. In: International Conference on Parallel Problem Solving from Nature. pp. 651–660. http://dx.doi.org/10.1007/978-3-540-87700-4_65.

Cardoso, J., 2007. Complexity analysis of BPEL web processes. Software Process: Improv. Pract. 12 (1), 35–49. http://dx.doi.org/10.1002/spip.302.

Cardoso, J., Mendling, J., Neumann, G., Reijers, H.A., 2006. A discourse on complexity of process models. In: International Conference on Business Process Management. Springer, pp. 117–128. http://dx.doi.org/10.1007/11837862_13.

Cliff, N., 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. Psychol. Bull. 114 (3), 494. http://dx.doi.org/10.1037/0033-2909.114.3.494.

Coello, C.A.C., Cortés, N.C., 2005. Solving multiobjective optimization problems using an artificial immune system. Genet. Program. Evol. Mach. 6 (2), 163–190. http://dx.doi.org/10.1007/s10710-005-6164-x.

Coello, C.A.C., Sierra, M.R., 2004. A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm. In: Mexican International Conference on Artificial Intelligence. Springer, pp. 688–697. http://dx.doi.org/10.1007/978-3-540-24694-7_71.

Coscia, J.L.O., Mateos, C., Crasso, M., Zunino, A., 2011. Avoiding wsdl bad practices in code-first web services. In: 12th Argentine Symposium on Software Engineering. ASSE2011, pp. 1–12.

Daagi, M., Ouniy, A., Kessentini, M., Gammoudi, M.M., Bouktif, S., 2017. Web service interface decomposition using formal concept analysis. In: 2017 IEEE International Conference on Web Services. ICWS, pp. 172–179. http://dx.doi.org/10.1109/ICWS.2017.30.

Deb, K., Jain, S., 2002. Running performance metrics for evolutionary multi-objective optimization.

Deb, K., Jain, H., 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. IEEE Trans. Evol. Comput. 18 (4), 577–601. http://dx.doi.org/10.1109/TEVC.2013.2281535.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197. http://dx.doi.org/10.1109/4235.996017.

di Pierro, F., Khu, S.-T., Savic, D.A., 2007. An investigation on preference order ranking scheme for multiobjective evolutionary optimization. IEEE Trans. Evol. Comput. 11 (1), 17–45. http://dx.doi.org/10.1109/TEVC.2006.876362.

Dudney, B., Asbury, S., Krozak, J.K., Wittkopf, K., 2003. J2EE Antipatterns. John Wiley & Sons.

Hanley, J.A., McNeil, B.J., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143 (1), 29–36. http://dx.doi.org/10.1148/radiology.143.1.7063747.

Harman, M., Mansouri, S.A., Zhang, Y., 2012. Search-based software engineering: Trends, techniques and applications. ACM Comput. Surv. 45 (1), 1–61. http://dx.doi.org/10.1145/2379776.2379787.

Harman, M., McMinn, P., De Souza, J.T., Yoo, S., 2010. Search based software engineering: Techniques, taxonomy, tutorial. In: Empirical Software Engineering and Verification. Springer, pp. 1–59. http://dx.doi.org/10.1007/978-3-642-25231-0_1.

Harrer, S., Lenhard, J., Wirtz, G., 2012. BPEL conformance in open source engines. In: 2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications. SOCA, pp. 1–8. http://dx.doi.org/10.1109/SOCA.2012.6449467.

Hertis, M., Juric, M.B., 2014. An empirical analysis of business process execution language usage. IEEE Trans. Softw. Eng. 40 (8), 738–757. http://dx.doi.org/10.1109/TSE.2014.2322618.

Hirsch, M., Rodriguez, A., Rodriguez, J.M., Mateos, C., Zunino, A., 2018. Spotting and removing WSDL anti-pattern root causes in code-first web services using NLP techniques: A thorough validation of impact on service discoverability. Comput. Stand. Interfaces 56, 116–133. http://dx.doi.org/10.1016/j.csi.2017.09.010, URL https://www.sciencedirect.com/science/article/pii/S0920548917300892.

Hochberg, Y., 1988. A sharper Bonferroni procedure for multiple tests of significance. Biometrika 75 (4), 800–802. http://dx.doi.org/10.2307/2336325.

Hua, G., Shi, Y., Chunwei, W., 2014. A correctness verification approach of the BPEL exception handling CPN model based on temporal property. J. Networks 9 (10), 2743. http://dx.doi.org/10.4304/jnw.9.10.2743-2750.

Ji, S., Li, B., Zhang, P., 2019. XCFG based data flow analysis of business processes. In: 2019 5th International Conference on Information Management. ICIM, pp. 71–76. http://dx.doi.org/10.1109/INFOMAN.2019.8714686.

Kabbaj, M.I., Bétari, A., Bakkoury, Z., Rharbi, A., 2015. Towards an active help on detecting data flow errors in business process models. Int. J. Comput. Sci. Appl. 12, 16–25.

Karnopp, D.C., 1963. Random search techniques for optimization problems. Automatica 1 (2), 111–121. http://dx.doi.org/10.1016/0005-1098(63)90018-9, URL https://www.sciencedirect.com/science/article/pii/0005109863900189.

Kessentini, M., Ouni, A., 2017. Detecting android smells using multi-objective genetic programming. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems. MOBILESoft, pp. 122–132. http://dx.doi.org/10.1109/MOBILESoft.2017.29.

Kim, J.H., 2019. Multicollinearity and misleading statistical results. Korean J. Anesthesiol. 72 (6), 558–569. http://dx.doi.org/10.4097/kja.19087.

Kloppmann, M., König, D., Leymann, F., Pfau, G., Rickayzen, A., Von Riegen, C., Schmidt, P., Trickovic, I., 2005. WS-BPEL extension for sub-processes–BPEL-SPE. p. 49, Joint White Paper, IBM and SAP.

Koschmider, A., Laue, R., Fellmann, M., 2019. Business process model anti-patterns: A bibliography and taxonomy of published work.

Kral, J., Zemlicka, M., 2007. The most important service-oriented antipatterns. In: International Conference on Software Engineering Advances. ICSEA 2007, p. 29. http://dx.doi.org/10.1109/ICSEA.2007.74.

Král, J., Zemlicka, M., 2009. Popular SOA antipatterns. In: 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns. pp. 271–276. http://dx.doi.org/10.1109/ComputationWorld.2009.80.

Latva-Koivisto, A.M., 2001. Finding a Complexity Measure for Business Process Models. Helsinki University of Technology, Systems Analysis Laboratory.

Laue, R., Awad, A., 2010. Visualization of business process modeling anti patterns. Electron. Commun. EASST 25, http://dx.doi.org/10.14279/tuj.eceasst.25.344.

Lenhard, J., 2017. Improving process portability through metrics and continuous inspection. In: Grambow, G., Oberhauser, R., Reichert, M. (Eds.), Advances in Intelligent Process-Aware Information Systems: Concepts, Methods, and Technologies. Springer International Publishing, Cham, pp. 193–223. http://dx.doi.org/10.1007/978-3-319-52181-7_8.

Lenhard, J., Wirtz, G., 2013a. Detecting portability issues in model-driven BPEL mappings (S). In: The 25th International Conference on Software Engineering and Knowledge Engineering. Boston, MA, USA, June 27-29, 2013, Knowledge Systems Institute Graduate School, pp. 18–21.

Lenhard, J., Wirtz, G., 2013b. Measuring the portability of executable service-oriented processes. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference. pp. 117–126. http://dx.doi.org/10.1109/EDOC.2013.21.

Lenhard, J., Wirtz, G., 2016. Portability of executable service-oriented processes: Metrics and validation. Serv. Orient. Comput. Appl. 10 (4), 391–411. http://dx.doi.org/10.1007/s11761-016-0195-4.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58. http://dx.doi.org/10.1016/j.orp.2016.09.002, URL https://www.sciencedirect.com/science/article/pii/S2214716015300270.

Lübke, D., Unger, T., Wutke, D., 2019. Analysis of data-flow complexity and architectural implications. In: Lübke, D., Pautasso, C. (Eds.), Empirical Studies on the Development of Executable Business Processes. Cham, pp. 59–81.

Ma, Y., He, H., 2013. Imbalanced learning: foundations, algorithms, and applications. http://dx.doi.org/10.1002/9781118646106.

Meda, H.S., Sen, A.K., Bagchi, A., 2010. On detecting data flow errors in workflows. J. Data Inf. Qual. (JDIQ) 2 (1), 1–31. http://dx.doi.org/10.1145/1805286.1805290.

Mendling, J., 2008. Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, first ed. Springer Publishing Company, Incorporated.

Mendling, J., Neumann, G., Van Der Aalst, W., 2007. Understanding the occurrence of errors in process models based on metrics. In: OTM Confederated International Conferences" on the Move to Meaningful Internet Systems". Springer, pp. 113–130. http://dx.doi.org/10.1007/978-3-540-76848-7_9.

Meunier, H., Talbi, E.-G., Reininger, P., 2000. A multiobjective genetic algorithm for radio network optimization. In: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), Vol. 1. pp. 317–324. http://dx.doi.org/10.1109/CEC.2000.870312.

Mitchell, M., 1998. An Introduction to Genetic Algorithms. MIT Press.

Mkaouer, W., Kessentini, M., Shaout, A., Koligheu, P., Bechikh, S., Deb, K., Ouni, A., 2015. Many-objective software remodularization using NSGA-III. ACM Trans. Software Eng. Methodol. (TOSEM) 24 (3), 1–45. http://dx.doi.org/10.1145/2729974.

Nissen, M.E., 1998. Redesigning reengineering through measurement-driven inference. MIS Q. 509–534. http://dx.doi.org/10.2307/249553.

Ouni, A., 2020. Search-based software engineering: Challenges, opportunities and recent applications. In: Proceedings of the Genetic and Evolutionary Computation Conference. GECCO, pp. 1114–1146. http://dx.doi.org/10.1145/3449726.3461425.

Ouni, A., Daagi, M., Kessentini, M., Bouktif, S., Gammoudi, M.M., 2017a. A machine learning-based approach to detect web service design defects. In: 2017 IEEE International Conference on Web Services. ICWS, pp. 532–539. http://dx.doi.org/10.1109/ICWS.2017.62.

Ouni, A., Gaikovina Kula, R., Kessentini, M., Inoue, K., 2015a. Web service antipatterns detection using genetic programming. In: Annual Conference on Genetic and Evolutionary Computation. pp. 1351–1358. http://dx.doi.org/10.1145/2739480.2754724.

Ouni, A., Kessentini, M., Inoue, K., Cinnéide, M.Ó., 2017b. Search-based web service antipatterns detection. IEEE Trans. Serv. Comput. 10 (4), 603–617. http://dx.doi.org/10.1109/TSC.2015.2502595.

Ouni, A., Kessentini, M., Sahraoui, H., Hamdi, M.S., 2012. Search-based refactoring: Towards semantics preservation. In: 28th IEEE International Conference on Software Maintenance. ICSM, pp. 347–356.

Ouni, A., Kessentini, M., Sahraoui, H., Inoue, K., Hamdi, M.S., 2015b. Improving multi-objective code-smells correction using development history. J. Syst. Softw. 105, 18–39.

Ouni, A., Kula, R.G., Kessentini, M., Ishio, T., German, D.M., Inoue, K., 2017c. Search-based software library recommendation using multi-objective optimization. Inf. Softw. Technol. 83, 55–75.

Ouni, A., Salem, Z., Inoue, K., Soui, M., 2016. SIM: An automated approach to improve web service interface modularization. In: 2016 IEEE International Conference on Web Services. ICWS, IEEE, pp. 91–98. http://dx.doi.org/10.1109/ICWS.2016.20.

Ouni, A., Wang, H., Kessentini, M., Bouktif, S., Inoue, K., 2018. A hybrid approach for improving the design quality of web service interfaces. ACM Trans. Internet Technol. (TOIT) 19 (1), 4. http://dx.doi.org/10.1145/3226593.

Ouyang, C., Verbeek, E., van der Aalst, W.M., Breutel, S., Dumas, M., ter Hofstede, A.H., 2007. Formal semantics and analysis of control flow in WS-BPEL. Sci. Comput. Programm. 67 (2), 162–198. http://dx.doi.org/10.1016/j.scico.2007.03.002, URL https://www.sciencedirect.com/science/article/pii/S0167642307000500.

Palma, F., Moha, N., Guéhéneuc, Y.-G., 2013. Detection of process antipatterns: A BPEL perspective. In: 2013 17th IEEE International Enterprise Distributed Object Computing Conference Workshops. pp. 173–177. http://dx.doi.org/10.1109/EDOCW.2013.26.

Palma, F., Moha, N., Tremblay, G., Guéhéneuc, Y.-G., 2014. Specification and detection of SOA antipatterns in web services. In: European Conference on Software Architecture. Springer, pp. 58–73. http://dx.doi.org/10.1007/978-3-319-09970-5_6.

Petcu, D., Vasilakos, A.V., 2014. Portability in clouds: Approaches and research opportunities. Scalable Comput.: Pract. Exp. 15 (3), 251–270. http://dx.doi.org/10.12694/scpe.v15i3.1019.

Ramadan, Q., Strüber, D., Salnitri, M., Riediger, V., Jürjens, J., 2018. Detecting conflicts between data-minimization and security requirements in business process models. In: Pierantonio, A., Trujillo, S. (Eds.), Modelling Foundations and Applications. Springer International Publishing, Cham, pp. 179–198. http://dx.doi.org/10.1007/978-3-319-92997-2_12.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2011. Classifier chains for multi-label classification. Mach. Learn. 85 (3), 333. http://dx.doi.org/10.1007/s10994-011-5256-5.

Riemann, U., 2016. Benefits and challenges for business process management in the cloud. In: Web-Based Services: Concepts, Methodologies, Tools, and Applications. IGI Global, pp. 2096–2121. http://dx.doi.org/10.4018/IJOCI.2015040104.

Riquelme, N., Von Lücken, C., Baran, B., 2015. Performance metrics in multi-objective optimization. In: 2015 Latin American Computing Conference. CLEI, pp. 1–11. http://dx.doi.org/10.1109/CLEI.2015.7360024.

Rodriguez, J.M., Crasso, M., Mateos, C., Zunino, A., 2013. Best practices for describing, consuming, and discovering web services: A comprehensive toolset. Softw. - Pract. Exp. 43 (6), 613–639. http://dx.doi.org/10.1002/spe.2123.

Rodriguez, J.M., Crasso, M., Zunino, A., Campo, M., 2010. Automatically detecting opportunities for web service descriptions improvement. In: Conference on E-Business, E-Services and E-Society. Springer, pp. 139–150. http://dx.doi.org/10.1007/978-3-642-16283-1_18.

Romano, J., Kromrey, J.D., Coraggio, J., Skowronek, J., 2006. Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluating group differences on the NSSE and other surveys. In: Annual Meeting of the Florida Association of Institutional Research. pp. 1–33.

Rosinosky, G., Youcef, S., Charoy, F., 2016. An efficient approach for multi-tenant elastic business processes management in cloud computing environment. In: 2016 IEEE 9th International Conference on Cloud Computing. CLOUD, pp. 311–318. http://dx.doi.org/10.1109/CLOUD.2016.0049.

Roy, S., Sajeev, A., Bihary, S., Ranjan, A., 2014. An empirical study of error patterns in industrial business process models. IEEE Trans. Serv. Comput. 7 (2), 140–153. http://dx.doi.org/10.1109/TSC.2013.10.

Sadiq, W., Orlowska, M.E., 2000. Analyzing process models using graph reduction techniques. Inf. Syst. 25 (2), 117–134, The 11th International Conference on Advanced Information System Engineering.

Saidani, I., Ouni, A., Mkaouer, M.W., 2020. Web service API anti-patterns detection as a multi-label learning problem. In: International Conference on Web Services. pp. 114–132. http://dx.doi.org/10.1007/978-3-030-59618-7_8.

Song, W., 2018. BPEL proceses for reserach only. Available at : https://github.com/stilab-ets/BPELantipatterns.

Song, W., Jacobsen, H., 2018. Static and dynamic process change. IEEE Trans. Serv. Comput. 11 (01), 215–231. http://dx.doi.org/10.1109/TSC.2016.2536025.

Song, W., Ma, X., Cheung, S., Hu, H., Jian Lü, J., 2010. Preserving data flow correctness in process adaptation. In: 2010 IEEE International Conference on Services Computing. pp. 9–16. http://dx.doi.org/10.1109/SCC.2010.24.

Song, W., Zhang, C., Jacobsen, H.-A., 2021. An empirical study on data flow bugs in business processes. IEEE Trans. Cloud Comput. 9 (1), 88–101. http://dx.doi.org/10.1109/TCC.2018.2844247.

Standard, O., 2007. Web services business process execution language version 2.0. p. 64, URL http://docsoasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Trčka, N., van der Aalst, W.M.P., Sidorova, N., 2009. Data-flow anti-patterns: Discovering data-flow errors in workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (Eds.), Advanced Information Systems Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 425–439. http://dx.doi.org/10.1007/978-3-642-02144-2_34.

Van der Aalst, W.M., 1998. The application of Petri nets to workflow management. J. Circuits Syst. Comput. 8 (01), 21–66.

Wang, H., Kessentini, M., Ouni, A., 2021. Interactive refactoring of web service interfaces using computational search. IEEE Trans. Serv. Comput. 14 (1), 179–192. http://dx.doi.org/10.1109/TSC.2017.2787152.

Wilcoxon, F., Katti, S., Wilcox, R.A., 1970. Critical values and probability levels for the Wilcoxon rank sum test and the Wilcoxon signed rank test. Sel. Tables Math. Stat. 1, 171–259.

Yu, J., Han, Y.-B., Han, J., Jin, Y., Falcarin, P., Morisio, M., 2008. Synthesizing service composition models on the basis of temporal business rules. J. Comput. Sci. Tech. 23 (6), 885–894. http://dx.doi.org/10.1007/s11390-008-9196-x.

Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-Rep. 103, http://dx.doi.org/10.3929/ethz-a-004284029.

Marwa Daagi is currently a PhD student the University of Tunis El Manar, member of Software Technology and Intelligence Research Group at ETS Montreal. Her PhD project is concerned with the application of intelligent search and machine learning to improve the quality of Web service-based software systems. Her current research interests are Search-Based Software Engineering, web services, refactoring, Software maintenance and software quality.

Ali Ouni received the PhD degree in computer science from the University of Montreal in 2015. He is currently an associate professor with the Department of Software Engineering and IT, ETS Montreal, University of Quebec, where he leads the Software Technology and Intelligence (STIL) Research Lab, since 2017. His research interests include software engineering including software maintenance and evolution, refactoring of software systems, software quality, service-oriented computing, and the application of artificial intelligence techniques to software engineering. He was a program committee member and reviewer in several journals and conferences. He is a member of the ACM and IEEE Computer Society.

Mohamed Mohsen Gammoudi is currently a full Professor in Computer Science Department at ISAMM, University of Manouba. He is the responsible of the research Team ECRI, RIADI Laboratory. He obtained his HDR in 2005 at the Faculty of Sciences of Tunis (FST). He received his PhD in 1993 at Sophia Antipolis Laboratory (I3S/CNRS) in the team of Professor Serge Miranda, France. He was hired as a Visiting Professor between 1993 and 1997 at the Federal University of Maranhao, Brazil.

Salah Bouktif (Member, IEEE) received the degree in engineering and the master's degree in industrial computing from the School of Computer Science, University of Tunis, Tunisia, and the Ph.D. degree (Hons.) in computer science from the University of Montreal, in 2015. He is currently an Associate Professor with the College of Information Technology (CIT), United Arab Emirates University. Before joining CIT, in 2007, he was a Postdoctoral Fellow with the Department of Computer Engineering, Polytechnic School of Engineering, Montreal. His research interests include energy prediction, data mining, big data analytics, search-based software engineering, and software quality assurance.

Mohamed Wiem Mkaouer received the PhD degree from the University of Michigan-Dearborn in 2016. He is currently an assistant professor with the Software Engineering Department, in the B. Thomas Golisano College of Computing and Information Sciences, Rochester Institute of Technology. His research interests include software quality, refactoring, and software testing. He is the lead of the Software Maintenance and Intelligent Evolution (SMILE) Research Group. He is a member of the Association for Computing Machinery and the IEEE Computer Society.