Assessing business process models: a literature review on techniques for BPMN testing and formal verification

Tomás Lopes and Sérgio Guerreiro
Department of Computer Science and Engineering, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal and Information and Decision Support Systems (IDSS), Instituto de Engenharia de Sistemas e Computadores – Investigação e Desenvolvimento em Lisboa (INESC-ID), Lisboa, Portugal

Received 3 November 2022
Revised 2 February 2023
2 March 2023
Accepted 10 March 2023

Abstract

Purpose – Testing business processes is crucial to assess the compliance of business process models with requirements. Automating this task optimizes testing efforts and reduces human error while also providing improvement insights for the business process modeling activity. The primary purposes of this paper are to conduct a literature review of Business Process Model and Notation (BPMN) testing and formal verification and to propose the Business Process Evaluation and Research Framework for Enhancement and Continuous Testing (bPERFECT) framework, which aims to guide business process testing (BPT) research and implementation. Secondary objectives include (1) eliciting the existing types of testing, (2) evaluating their impact on efficiency and (3) assessing the formal verification techniques that complement testing.

Design/methodology/approach – The methodology used is based on Kitchenham's (2004) original procedures for conducting systematic literature reviews.

Findings – Results of this study indicate that three distinct business process model testing types can be found in the literature: black/gray-box, regression and integration. Testing and verification approaches differ in aspects such as awareness of test data, coverage criteria and auxiliary representations used. However, most solutions pose notable hindrances, such as BPMN element limitations, that lead to limited practicality.

Research limitations/implications – The databases selected in the review protocol may have excluded relevant studies on this topic. More databases and gray literature could also be considered for inclusion in this review.

Originality/value – Three main originality aspects are identified in this study as follows: (1) the classification of process model testing types, (2) the future trends foreseen for BPMN model testing and verification and (3) the bPERFECT framework for testing business processes.

Keywords BPMN, Business process, Business process testing, Model-based testing, Test automation, Formal verification

Paper type Literature review

1. Introduction

With business processes growing radically in size and complexity due to technological advancements and increased regulations (Paiva *et al.*, 2018), companies have been facing the challenge of maintaining their processes correct and compliant with the many imposed

This work was funded by national funds through Fundação para a Ciência e a Tecnologia (FCT) (No: UIDB/50021/2020 (INESC-ID)).

Business Process Management Journal
Vol. 29 No. 8, 2023
pp. 133-162
Emerald Publishing Limited
1463-7154
DOI 10.1108/BPMJ-11-2022-0557

requirements (Böhmer and Rinderle-Ma, 2015). In a world where constant adaptation is crucial, testing business processes and ensuring they continue to bring value, all the while dealing with change, becomes incredibly important (Guerreiro, 2020).

Business Process Testing (BPT) is defined by Paiva *et al.* (2018) as "the act of validating that end-to-end transactions through enterprise systems continue to work correctly as the underlying packaged applications evolve". It has become pivotal in guaranteeing the proper operation of business processes, with considerable time and effort often being reported in BPT activities.

Additionally, as a result of Business Process Management (BPM) (Dumas *et al.*, 2018) methodologies being used in distinct domains, such as Process-Driven Application creation (Schneid *et al.*, 2021), document management (Link Consulting, 2016) and model-driven Web service development (Yuan *et al.*, 2008), the need for robust and effective methods for assessing business process models has risen significantly. Through the application of testing and formal verification methods, organizations can build up their confidence in the designed processes and ensure that business rules and requirements are aligned while minimizing human effort (de Moura *et al.*, 2017). Insights provided by process assessment activities may also assist in expediting process redesign, which "despite all automation efforts,... has remained a manual, cognitively demanding task, making it time-consuming, labor-intensive and error-prone" (Beerepoot *et al.*, 2023).

Hence, a literature review focusing on classifying and aggregating the existing approaches and methodologies for testing Business Process Model and Notation (BPMN) 2.0 (Object Management Group, 2010) models in an automated fashion has the potential to help organizations reduce the time and effort required to test their business processes while also reducing human error rates, which are often substantial due to the highly repetitive nature of this activity. This review analyzes and compares techniques for verifying functional/behavioral properties, such as conformance to specific requirements (through testing) and structural properties, such as the absence of deadlocks and infinite loops or the presence of certain desired behaviors (through formal verification; Wynn *et al.*, 2009).

Although there are several distinct techniques for assessing a BPMN model's syntactic and structural correctness, testing models in terms of their desired behavior and possible flow paths can be a challenging task. Consequently, compiling and synthesizing current BPT techniques and investigating how formal verification techniques can be used for testing purposes becomes exceedingly important, enabling the creation of a knowledge base and identification of research gaps that can unify and provide support for future research (Kitchenham, 2004).

Furthermore, each existing solution is often designed for one specific purpose or tool, with a notable lack of shared knowledge. A framework for continuous BPT is proposed at the end of the review to tackle this problem, with the potential of bringing together and consolidating the existing concepts and serving as the foundation for new testing methods developed in the future by researchers and organizations alike.

This paper is structured as follows. Section 2 explains some basic theoretical concepts required for the proper understanding of the review. Then, Section 3 contains the literature review itself, explaining the procedures followed, the results achieved and a final analysis. Finally, the conclusion is presented in Section 4.

2. Background

This section introduces the relevant theoretical background. Firstly, BPMN, a *de facto* standard used to model business processes, is covered. Then, the concepts of model-based testing (MBT) and model-driven engineering (MDE) are explained.

2.1 Business Process Model and Notation (BPMN)

BPMN is a graphical language used to model and execute business process models (Object Management Group, 2010). It comprises five types of elements (Object Management Group, 2010).

(1) Flow objects, which dictate the fundamental behavior of the process, comprising events, activities and gateways.

(2) Connecting objects, which connect flow objects to one another or to other elements.

(3) Data, used to "model the items. . . that are created, manipulated, and used during the execution of a process" (Object Management Group, 2010).

(4) Swimlanes, used to group and partition activities.

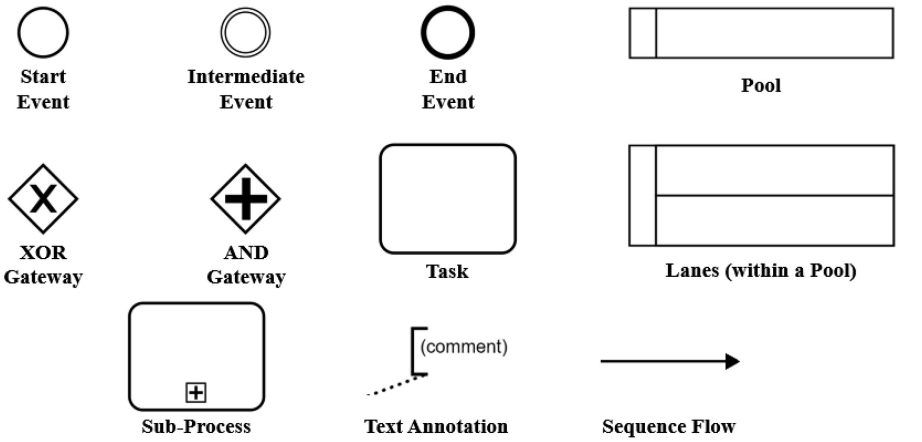
(5) Artifacts, used for supplementary process information.

A visual representation of a few essential elements of BPMN can be found in Figure 1. Events are used to trigger the start and completion of process instances and represent some form of change in behavior. Activities "represent points in a Process flow where work is performed" (Object Management Group, 2010), including Tasks (atomic) and Sub-Processes (representing another process). Exclusive (XOR) and Parallel (AND) Gateways are used to represent alternative and concurrent paths, respectively. These elements are connected using Sequence Flows, grouped into Pools and Lanes and can have Text Annotations to convey additional information about the process (Object Management Group, 2010).

2.2 Model-based testing and model-driven engineering

MBT is a common software testing approach that refers to the automatic derivation of test cases from system models (Paiva *et al.*, 2018; Schieferdecker, 2012). This technique aims to automate the design and implementation stages of software testing, which are often performed manually.

MBT comprises three core tasks (Schieferdecker, 2012).



Source(s): Figure by authors
Adapted from Object Management Group (2010, pp. 29–35)

Figure 1.
BPMN 2.0 elements

(1) Modeling,

(2) Defining test criteria and

(3) Generating tests.

These three core tasks often include other sub-tasks, such as defining the modeling notation and choosing the appropriate test generation techniques (Paiva *et al.*, 2018). For the purpose of this work, a fourth activity is also considered as a core task of MBT.

(4) Executing tests.

Some of the benefits of MBT include (Schieferdecker, 2012).

(1) Lower test writing costs,

(2) Better test documentation and

(3) Automatic test coverage measurement.

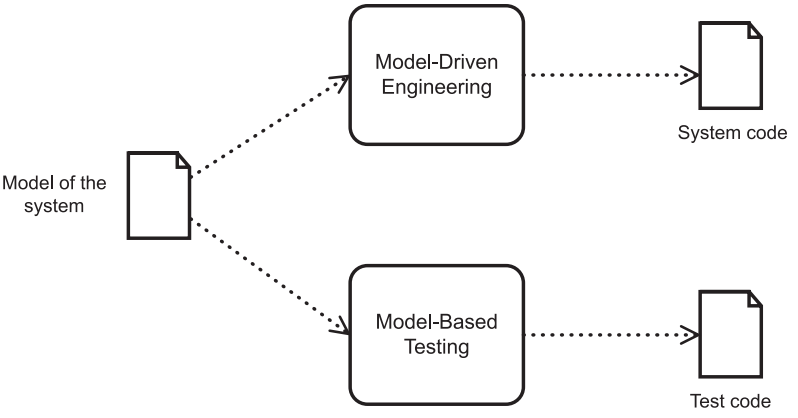
Organizations can make use of the full potential of MBT techniques by having an adequate MDE infrastructure implemented (Schieferdecker, 2012). MDE is a model-oriented software development approach which consists of "the systematic use of models as primary artifacts during a software engineering process" (Hutchinson *et al.*, 2011).

Combining MDE's ability to generate system code and MBT's ability to generate test code from the same system models, as Figure 2 illustrates, enables faster and more efficient software development. While MDE contributes towards automating the implementation of a system, MBT contributes towards automating its testing.

3. Literature review

A literature review was performed with the goal of doing a state-of-the-art analysis of existing approaches and methods for BPT – more specifically, BPMN model testing.

Before commencing the review, the authors did a preliminary analysis of existing literature reviews to determine if any of them address the concrete topic of BPMN testing. Although there is no existing review on this subject specifically (to the best of the authors' knowledge), one review, in particular, was found, authored by (Böhmer and Rinderle-Ma, 2015), covering approaches and challenges for process model testing (in a language-agnostic



Source(s): Figure by authors

Figure 2.
MDE and MBT used in combination

fashion). No focus is done on BPMN specifically, but the more general nature of the study means that many specific aspects of BPMN-related testing and test case generation were also tackled in this review.

Even so, this review was published in 2015 and, consequently, misses recent advancements and developments in the field. Not only that, but in the present study, due to the increasing popularity of the BPMN language and growing application in distinct domains, a stronger focus on the BPMN language is desired. This leaves the door open for analysis of studies that make use of specific features and elements of the BPMN language that may not be present in other languages, without an overwhelming amount of required technical background from other languages, frameworks, or scientific fields.

These were deemed sufficiently strong reasons to continue with the execution of this review.

3.1 Objectives and research questions

The main objectives of this literature review are to create a knowledge base for BPMN model testing and formal verification methods and to devise a framework to guide future BPT research and development.

Notwithstanding the possibility of still having a human component during the testing procedure, automating the generation and execution of tests leads to expedited and optimized process implementation. Enumerating and summarizing existing techniques for BPMN model testing and studying how verification methods may complement them lays the groundwork for creating a testing framework which, in turn, will make it easier for organizations to implement fully-fledged BPT practices while also allowing the identification of research gaps (Kitchenham, 2004).

This study also seeks to bring a greater conceptual, theoretical and practical unification of the field of BPT, with the potential to lay the groundwork for most research in this area and open the door for better, more sophisticated solutions with greater shared knowledge.

To tackle this topic, two baseline research questions were formulated using the Setting, Perspective, Intervention/Interest, Comparison and Evaluation (SPICE) framework (Booth, 2006) for formulating qualitative research questions. These research questions were incrementally refined following their initial formulation as a deeper understanding of the topic in question was gained. Each question contains sub-questions that arose as a result of the iterative refinement that the research questions were subject to throughout the execution of the review. These sub-questions aim to dive deeper into specific topics of BPT and BPMN testing that were deemed relevant as more knowledge on the broader topic was gained.

The research questions are:

RQ1. Which types of process model testing approaches are currently being used by process modelers and business analysts for BPT?

RQ1.1. What are the practical applications of each type of business process model testing?

RQ2. Which approaches are currently in use by process modelers and business analysts for testing and verification of BPMN models?

RQ2.1. What impact do said approaches have on testing efficiency?

RQ2.2. How can existing verification techniques complement BPMN testing?

3.2 Methodology

Kitchenham's (2004) original procedures for performing systematic literature reviews are a tried-and-true set of guidelines for performing literature reviews in the software engineering field. These comprise three major steps.

(1) *Planning*, where the study rationale is stated and the review protocol is designed, including the associated research questions,

(2) *Conducting*, meant for carrying out the developed review protocol, selecting and extracting data and analyzing the results of the document selection and

(3) *Reporting*, which involves stating the key findings and answering the established research questions.

The review methodology employed in this study was based on these procedures. Table 1 contains all the specific steps and sub-steps followed for the execution of this review.

According to Kitchenham (2004), the main motives behind a literature review consist of (1) summarizing existing information, (2) identifying research gaps and (3) creating a framework for future research. The objectives stated for this review are aligned with said motives. Furthermore, the tasks undertaken for findings systematization are directly related to each of the three motives.

(1) Study synthesis, classification and comparison output a summary of all information in the literature included in the review and leverage further discussion,

(2) Exploratory discussion enables critical evaluation of existing solutions, allowing the formulation of well-grounded answers to the proposed research questions and the identification of research gaps and

(3) The proposal of a framework for testing business processes further guides future research through the creation of a common starting point for any BPT techniques/approaches developed in the future.

3.3 Protocol design

This section covers the design of the research protocol, establishing a set of rules regarding inclusion criteria, database selection, query string definition and selection process design.

3.3.1 Defining inclusion criteria. A document qualifies for abstract screening if it abides by all the following inclusion criteria.

#	Steps	Sub-steps
1	Planning	State objectives and baseline research questions Design research protocol • Define inclusion criteria • Define databases • Define query strings • Describe selection process
2	Conducting	Perform study selection Describe and analyze results
3	Reporting	Systematize findings • Synthesize, classify and compare studies • Perform exploratory discussion • Propose framework for future research Present findings and answer research questions

Table 1.
Literature review steps and sub-steps

Source(s): Based on Kitchenham (2004)
Table by authors

(1) *Document type*: journal articles or meeting (conference/symposium and workshop proceedings) papers,

(2) *Research area*: Computer Science or Engineering and

(3) *Language*: English or Portuguese.

Any document that does not meet all of these criteria is excluded.

Accepted studies were limited to articles or conference/symposium and workshop proceedings papers, allowing a sufficiently vast pool of documents developed in an academic context to be returned by the search while also discarding other possible sources of information that may be more informal or unreliable such as white papers.

Filtering by research area also allows the preemptive elimination of documents that could otherwise constitute false positives.

No publication date limitation was imposed. Hence, it is implicit that the date limit for the publication of documents corresponds to the date the queries were run on (March 22, 2022).

Finally, the language was limited to English and Portuguese due to the authors and reviewers of this research being fluent in these two languages.

3.3.2 Defining databases. The *Web of Science Core Collection* is a comprehensive collection of citation indexes of journals, books and proceedings according to several criteria related to quality and impact. This collection, essentially a "database of databases" (see Clarivate, 2021 for indexing details and more information on database coverage), can be queried using the *Web of Science* [1] platform. This collection/platform pair proved a good choice for this review, ensuring high-quality sources while offering instrumental search operators not present in other platforms that enabled a suitable refinement of the query strings used, which would not have been possible using other available platforms. This platform also provides vital functionalities for literature reviews that facilitate abstract screening, filtering and results exporting and analysis, making the review procedure significantly more manageable.

3.3.3 Defining the query strings. The following search strings were used to query the aforementioned collection.

S1 (TI=(("business process" OR "process model") AND (verif OR valid* OR test*)) AND AB=BPMN)*

OR

(TI=(BPMN AND (verif OR valid* OR test*)))*

OR

(AB=(BPMN NEAR (verif OR valid* OR test*)))*

S2 (TI=("business process test*" OR "test* business process*" OR (("model-based testing" OR "model based testing" OR MBT OR ("test case*")) NEAR/5 (BPM OR BPMN OR "business process"))))*

OR

AB=(AB=("business process test*" OR "test* business process*" OR (("model-based testing" OR "model based testing" OR MBT OR ("test case*")) NEAR/5 (BPM OR BPMN OR "business process"))))*

Query S1 searches for documents with key terms in the title (field tag TI) and/or the abstract (field tag AB) more closely related to research question RQ2, such as BPMN and test, using the wildcard operator (*) as a form of manual stemming to enable the matching of those terms

with any variants of the root word (for instance, the search term valid* matches the words valid, validate, validation and such). The query can be split into three disjoint fragments.

(1) The first fragment searches for documents with "business process" or "process model" and a variant of either verify, validate or test in the title, as well as the term BPMN in the abstract.

(2) The second fragment searches for documents with BPMN and a variant of either verify, validate or test in the title.

(3) The third fragment searches for documents with BPMN and a variant of either verify, validate or test in the abstract, with both terms being at most 15 words apart from one another.

Using the TI and AB field tags in the search string allows the search to only return documents whose primary focus is the topic in question. Additionally, Web of Science does not allow the usage of the NEAR operator in combination with the ALL field tag and replacing the occurrences of NEAR with AND led to a significant number of false positives.

The inclusion of the term validate (valid*) in the query string is justified by the term being commonly used as a synonym of test (test*) in the context of BPMN model analysis. Moreover, the term BPMN was included explicitly as a mandatory term in all three fragments, given the desired focus on BPMN of question RQ2.

As for query string S2, it searches specifically for documents which apply MBT to business process models, using the same logic as string S1 regarding manual stemming and field tag usage. This query string is split into two fragments, with both fragments being equal except for the field tag: they each search for documents with titles/abstracts (respectively) containing either: a variant of the expression "business process testing"; or the phrase "Model-Based Testing" within 5 words of BPM, BPMN or "business process".

The main goal of this query string is to help answer research question RQ1, with all of the terms in the query being directly related to BPT and MBT.

In contrast with query string S1, the presence of the term BPMN is not enforced. This enables the analysis of the potential application of MBT methods designed for other languages to BPMN process models.

Overall, combining the somewhat strict query formulation (regarding, for instance, field tags) with the comparatively more modest inclusion criteria led to both a manageable amount of documents to review and a sufficiently good relevance rate for each query string.

3.3.4 Describing the selection process. The selection process was divided into four stages.

(1) Searching (using the specified query strings),

(2) Applying the filters (per the specified inclusion criteria),

(3) Screening abstracts and

(4) Screening full-text documents.

The number of valid documents was recorded after each stage of the selection process, with documents that did not make it past any given stage being discarded.

For stage 3, the reviewers read the abstracts and discarded any evidently extraneous documents. For stage 4, the reviewers downloaded and read the full-text documents and excluded any documents deemed redundant (for instance, if multiple studies written by the same authors contained equal or equivalent information) or irrelevant (focused on topics outside the scope of this review).

The documents deemed relevant at the end of stage 4 are the ones effectively analyzed, summarized and reviewed.

3.4 Study selection

The study selection process can begin with the protocol carefully designed, following the stages outlined in Subsection 3.3.4.

The search was made on, using query strings S1 and S2.

The search using query string S1 yielded 193 results. Next up, the filters corresponding to the inclusion criteria specified were applied—173 documents remained. After screening each of the abstracts, 37 documents were deemed relevant. Finally, the full-text screening stage eliminated 16 of those documents, leaving 21 relevant documents.

On the flip side, the search using query string S2 returned a lower amount of results: 36. 34 documents remained after applying the filters. 16 documents were deemed relevant after abstract screening. Finally, reading the full-text documents eliminated 2 of those documents, leaving 14 documents.

The union of both document sets resulted in a total of *32 relevant documents*, given that 3 were returned by both queries.

Figure 3 contains a visual representation of this selection process as a flow diagram.

3.5 Description of results

This section includes statistical data analysis on the set of 32 studies that made it through the selection process.

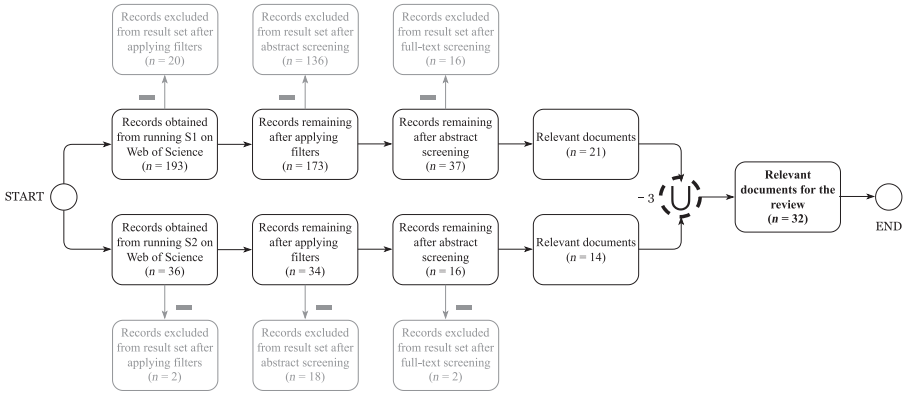
3.5.1 Publication year. The publication year distribution for the 32 selected articles can be found in Figure 4.

This histogram shows that none of the selected documents were published before 2007, which is almost certainly related to the release of the specification document for BPMN 1.0 dating February 2006. It is also worth noting that there was a noticeable increase in published documents about this subject starting in 2016, showing a growing interest in the topic.

3.5.2 Document types. Figure 5 shows the distribution of documents according to their source type (proceedings or journals) for each query string.

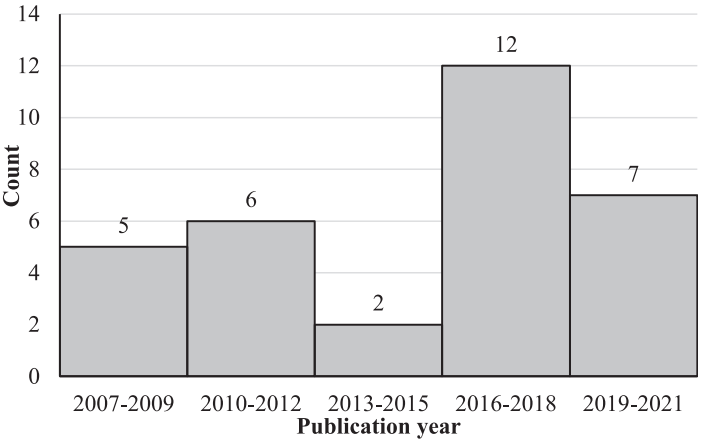
Interestingly, the percentage of proceedings papers is significantly higher for query string S2 when compared to query string S1.

3.5.3 Common terms. Figure 6 shows a word cloud of the most common terms found across all studies. As expected, among the most common words are terms such as *test*, *bpmn* and *model*. Other interesting occurrences are *cpn*, *data* and *graph*.



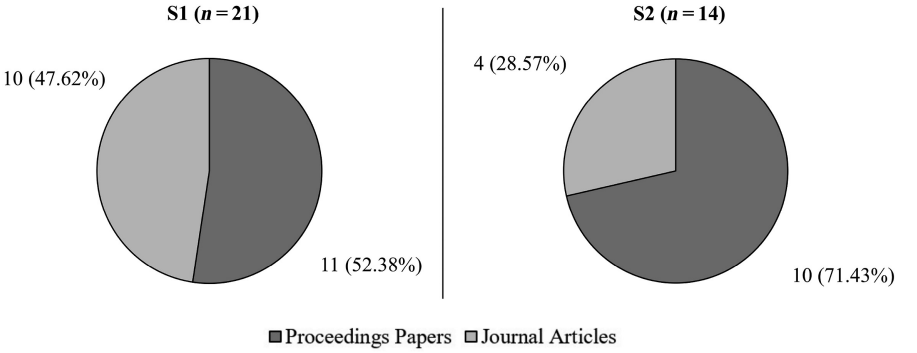
Source(s): Figure by authors

Figure 3.
Study selection process



Source(s): Figure by authors

Figure 4.
Publication year distribution of selected studies



Source(s): Figure by authors

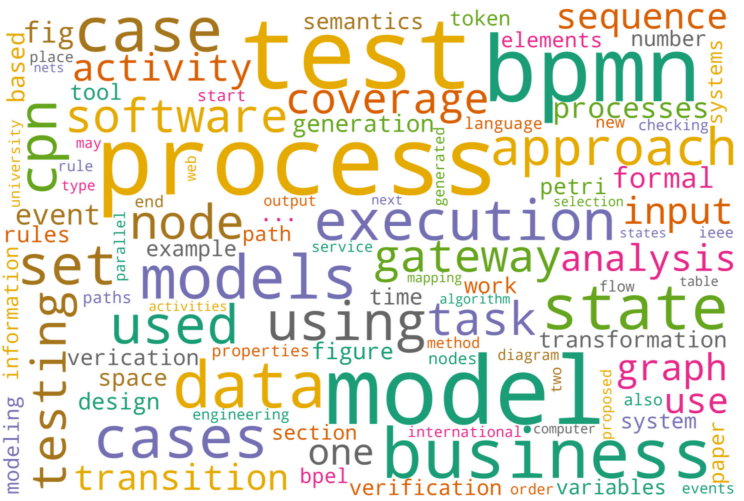
Figure 5.
Number and percentage of documents per source type

3.6 Summary

The 32 documents contained various approaches for business process model testing and verification. The rest of this section reviews each paper, structured per the taxonomy as shown in Table 2.

3.6.1 BPMN model testing. Studies which cover different approaches for BPMN model testing are (Buchs *et al.*, 2009; Makki *et al.*, 2017; de Moura *et al.*, 2017; Paiva *et al.*, 2018; Schneid *et al.*, 2021; Seqerloo *et al.*, 2019; Yotyawilai and Suwannasart, 2014).

Buchs *et al.* (2009) present a methodology and tool that generate test cases from BPMN models using a transformation to algebraic Petri nets, higher-level Petri nets "where tokens are belonging to domains defined by algebraic specifications" (Buchs *et al.*, 2009), using user-defined data structures and Algebraic Abstract Data Types that dictate the associated valid operations and axioms. Test cases are automatically generated and represented using Hennessy-Milner Logic (HML), based on manually written HML formulas, using model-checking techniques such as Decision Diagrams. However, this study does not include an evaluation section. Not only that, but the rule-writing step is entirely manual and requires a firm knowledge of theoretical background and logic-related concepts that constrain its



Source(s): Figure by authors

Figure 6.
Word cloud of all selected studies, generated from the full-text documents

potential usage by a non-technical audience. Furthermore, this technique implies the existence of multiple models of the same process that may divert over time as the original model is changed. This is a common problem among many approaches—most of them use intermediate auxiliary representations of the process using different formalisms or languages instead of performing algorithms and analyses directly on the original models.

A very distinct approach is pioneered by Makki *et al.* (2017), which uses a Capture and Replay framework for semi-automated regression detection and testing of BPMN process models. The *Capture* phase consists of capturing execution snapshots of business process models in production. The *Replay* phase involves having the user isolate the process from external dependencies using mocks and specifying a specific past execution, generating a test case in which the end state of the new execution is compared to the final snapshot of the old execution to detect regression. An implementation of this approach in the jBPM (Java Business Process Model) execution engine is also described. This approach is evaluated in terms of performance overhead and fault tolerance. While it remains an interesting approach for regression test case generation, manual work is still required to select the appropriate past executions, specify mocks and specify exclusions for the regression detection mechanism.

A different two-step approach is presented by de Moura *et al.* (2017) for end-to-end test case generation from BPMN models. In the first step, the XML Process Definition Language (XPDL) representation of the model is traversed and an Excel file is generated, which contains information about all process flows, with each process/pool being handled separately. Each column corresponds to a specific task, whereas each row corresponds to a possible flow. Tasks present in each flow are marked with an X. Parallel paths are considered separate flows and sub-processes are treated as a single task. A recursion delimiter controls how many times the same sequence flow can be executed. Based on this table, test scripts can be automatically generated and run using Selenium and Cucumber. An example is shown, but no evaluation is performed. Additionally, the set of BPMN elements supported is quite limited and test data extraction, although mentioned, is not described.

Paiva *et al.* (2018) present a tool called ETAP-Pro (End-to-end Test Automation Platform for Processes), a platform for test case generation from BPMN models. This program finds all possible paths in a BPMN model in XPDL format using a Depth-First Search (DFS). For

#	Group label	Documents	# Docs
1	BPMN model testing	Buchs *et al.* (2009) Makki *et al.* (2017) de Moura *et al.* (2017) Paiva *et al.* (2018) Schneid *et al.* (2021) Seqerloo *et al.* (2019) Yotyawilai and Suwannasart (2014)	7
2	BPEL process testing	Blanco *et al.* (2009) Jahan *et al.* (2016) Ma *et al.* (2008) Nahak *et al.* (2019) Guangquan *et al.* (2007)	5
3	Testing based on other models	Bures *et al.* (2017) Yuan *et al.* (2008)	2
4	BPMN formal verification	Corradini *et al.* (2021) Lam (2010) Meghzili *et al.* (2020) Dechsupa *et al.* (2018) Dechsupa *et al.* (2019) Mendoza *et al.* (2010) Nazaruka *et al.* (2016) Dechsupa *et al.* (2021) Durán *et al.* (2018) Kheldoun *et al.* (2017) Kog *et al.* (2012) Szpyrka *et al.* (2017) Wong and Gibbons (2011) Yamasathien and Vatanawood (2014)	14
5	Other	Braghetto *et al.* (2011) Böhmer and Rinderle-Ma (2016) Dijkman and van Gorp (2010) Rachdi *et al.* (2016)	4

Table 2.
Selected document taxonomy

Source(s): Table by authors

parallel branches, all possible activity orders are generated, adding robustness in the face of possible race conditions. This tool generates test script skeletons in the Gherkin language. However, the test scripts themselves must be written manually. Additionally, this paper omits many of the implementation details, such as the test script skeleton generation.

Arguably the most significant contribution to the problem being investigated is provided by Schneid *et al.* (2021). This study presents an approach for semi-automatic regression test generation and execution for Process-Driven Applications, split into three steps. In the first step, *Preparation*, the tool performs an analysis of the process artifacts (which include BPMN models, Decision Model and Notation (DMN) decision models and HTML (HyperText Markup Language) forms) to discover all possible process execution paths, using an adapted DFS, as well as relevant data fields, in order to create what are called *test templates*, one per path or combination of parallel paths. These templates also include value suggestions which lead the flow down each path. In the second step, *Specification*, the user can specify test cases based on the test templates (and its suggested values) in a semi-automated fashion using a wizard. These test cases are stored using a custom Domain-Specific Language (DSL). Mocks can also be specified to isolate the test from external dependencies. The third step, *Generation*, is done

automatically after the specification is complete and consists of the generation of test code for a specific workflow engine. A visual representation of this approach in the form of a BPMN diagram can be found in Figure 7.

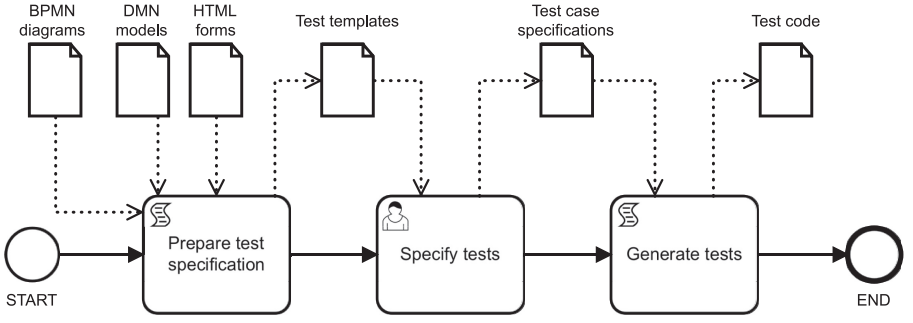
The authors also include a prototype which generates code for the Camunda BPM engine. This prototype was used in the evaluation, where BPM experts, process analysts and volunteer students tried the tool and answered a short questionnaire which inquired about certain pragmatic qualities of the tool, such as ease of use, utility and time-saving. Performance was also evaluated. Although there are shortcomings—such as the inability to handle complex expressions when generating value suggestions (the authors suggest the use of a constraint solver to deal with this issue but leave its implementation as future work), the manual work involved in test specification and the somewhat limited subset of BPMN elements supported (not directly stated in the paper)—this study contributes significantly towards answering the problem at hand.

Seqerloo *et al.* (2019) present a distinct approach to test case generation. A BPMN model is transformed into a state graph, where states represent gateways and transitions represent paths. Parallel blocks are represented as a single state. This state graph is traversed using a DFS. This state graph is then converted into a format that can be used as input for the Spec Explorer tool, following a specific mapping, then used to generate the test cases. However, as shown by the evaluation, this approach can find paths that constitute false positives. Additionally, this approach does not automatically determine values for the required data for the generated tests.

Finally, Yotyawilai and Suwannasart (2014) showcase another flow graph-based BPMN test case generation technique. Firstly, the tool searches for variables which require user input. The tool user is then prompted to specify additional missing properties about the variables, such as minimum and maximum values or length. Afterward, all the information about each model element is retrieved using an XML parser and used to create a flow graph. Ultimately, test cases are generated by traversing the flow graph using a DFS and simultaneously using the corresponding model element data. Each test case is specified in a table which contains the test path, inputs and expected outputs. However, the authors leave the tool's implementation as future work.

3.6.2 BPEL process testing. Studies which focus on testing processes represented in the Business Process Execution Language (BPEL) are (Blanco *et al.*, 2009; Guangquan *et al.*, 2007; Jahan *et al.*, 2016; Ma *et al.*, 2008; Nahak *et al.*, 2019).

Blanco *et al.* (2009) present a technique for test case generation of business processes written in BPEL based on an evolutionary method (specifically, a metaheuristic algorithm) called Scatter Search. This method relies on combining solutions in a Reference Set to obtain



Source(s): Figure by authors
Adapted from Schneid *et al.* (2021, p. 34)

Figure 7.
BPMN testing approach

new solutions with better levels of quality and diversity. The BPEL process is turned into a state graph, which then undergoes a search process with the goal of obtaining maximum transition coverage. Information about covered transitions is kept track of throughout the algorithm, which continues until all transitions are covered in the test cases (or until the number of test cases reaches a predefined maximum). However, the evaluation presented only compares the presented method to random test generation.

Nahak *et al.* (2019) also use a metaheuristic algorithm called Ant Colony Optimization for test case generation from BPEL processes in a very similar fashion: a state graph is generated and a similar search process is carried out, with the optimality of a test suite being determined by a balance between transition coverage and the number of test cases. In contrast with the previous approach, this study features a more detailed evaluation, establishing comparisons with random test case generation and two other test case generation methods, including the previous approach, showing better results regarding number of tests required to reach the desired coverage and test generation time.

Ma *et al.* (2008) demonstrate a technique for transforming BPEL processes to Stream X-Machines (SXMs), a variant of typical finite state machines with internal memory and transition inputs and outputs (Ipate and Holcombe, 2008), capable of modeling both control and data flows (Ma *et al.*, 2008). Test cases are generated using a generalization of the W-method, one of the most popular methods for finite state machine test case generation (Ipate and Banica, 2007). This approach is not evaluated.

The usage of BPEL makes these three methods and the ones described throughout the rest of this Subsection potentially unfeasible to be used by business analysts and modelers due to the technical skills required to capture and design business processes using this language (one of the main reasons for this review's focus on BPMN). Additionally, there is no complete mapping from BPEL to BPMN and creating one is far from trivial (Weidlich *et al.*, 2008).

Jahan *et al.* (2016) present a test generation technique from BPEL processes that relies on a sequence of intermediate transformations. Firstly, the BPEL process is turned into a Colored Petri Net (CPN). A CPN is a high-level Petri net that supports the declaration of primitive data types (called Color Sets), allowing the modeling of data flows. This format is used to check the correctness of the model through the construction of a reachability graph. Afterward, a Control Flow Graph (CFG) is built from the reachability graph, which eliminates superfluous states. A DFS is used to find a set of paths based on state and transition coverage criteria. A constraint solver determines the test data directly from the BPEL specification. Lastly, the test paths and test data are combined to create the full test cases, which can be converted into executable Java code. A visual representation of this approach in the form of a BPMN diagram can be found in Figure 8.

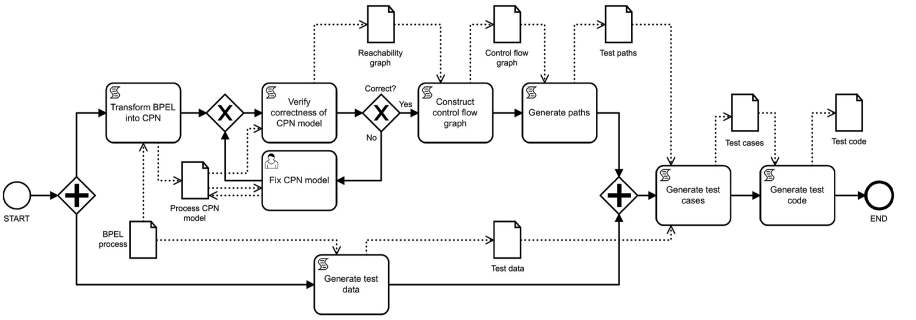


Figure 8.
BPEL testing approach

Source(s): Figure by authors
Adapted from Jahan *et al.* (2016, p. 140)

This study also briefly compares the presented approach with a technique solely based on reachability graphs and one solely based on CFGs. Despite the drawback previously mentioned associated with BPEL usage, the use of intermediate models in different formats proves to be advantageous in this case, given the potential for this method to be combined with other BPMN-to-CPN mappings (cf. Dechsupa *et al.*, 2018; Meghzili *et al.*, 2020). Having said that, the solution, as presented, is not suitable for situations where full path coverage is desired, which would require modifications to the test path generation algorithm.

A distinct approach is described by Guangquan *et al.* (2007). BPEL processes are mapped to an extension of Unified Modeling Language (UML) Activity Diagrams that allow them to model a broader range of BPEL behaviors. The outputted diagram is then traversed using a DFS and test cases are generated according to multiple coverage criteria. However, several details regarding test case generation are missing.

3.6.3 Testing based on other models. Studies which cover business process model testing using a distinct language or formalism to represent the model are (Bures *et al.*, 2017; Yuan *et al.*, 2008).

A technique is presented by Bures *et al.* (2017) named Prioritized Process Test, which generates path-based test cases from a process model represented as a directed graph, where nodes represent decision points and edges represent actions. Priorities can be assigned to actions. Actions with higher priority levels are tested more intensely according to two custom coverage criteria. Evaluation is performed through a comparison of the technique with more naive approaches. However, this technique requires manual input from process and business analysts when determining the priority of each action and is not suitable for comprehensive testing.

In the approach publicized by Yuan *et al.* (2008), executable test cases are generated from UML Activity Diagrams and BPEL specifications. These are first transformed into an abstract test case, which specifies details such as the test data and the behavior to be tested, traversing the activity diagram using a DFS. The abstract test cases are generated using node and transition coverage criteria. These test cases are then converted into executable code in the Testing and Test Control Notation Version 3 (TTCN-3) testing language. This approach is not evaluated.

3.6.4 BPMN formal verification. Studies that denote formal verification of BPMN as the main goal are: (Corradini *et al.*, 2021; Dechsupa *et al.*, 2018, 2019, 2021; Durán *et al.*, 2018; Kheldoun *et al.*, 2017; Kog *et al.*, 2012; Lam, 2010; Meghzili *et al.*, 2020; Mendoza *et al.*, 2010; Nazaruka *et al.*, 2016; Szpyrka *et al.*, 2017; Wong and Gibbons, 2011; Yamasathien and Vatanawood, 2014).

BProVe is a framework for formal verification of BPMN models (Corradini *et al.*, 2021). It uses an implementation of BPMN syntax and semantics in Maude (Corradini *et al.*, 2018), combining Linear Temporal Logic (LTL) model-checking, using Maude's LTL Logical Model Checker (LMC) and statistical model-checking using MultiVeStA, to verify that a given process model satisfies specific properties, which may be specified by the user. The authors experimentally evaluate the feasibility and scalability of this approach extensively.

This approach, along with all others based on formal verification and logic, cannot be used directly for testing purposes, serving typically as a way to complement testing.

Durán *et al.* (2018) use the rewriting SMT (Satisfiability Modulo Theories) framework, typically used to "model and analyze reachability properties of infinite-state open systems" (Durán *et al.*, 2018), to formally verify BPMN models. Each data field is represented as a logical variable in a satisfiability solver. No evaluation is presented.

Kheldoun *et al.* (2017) present a technique for transforming a BPMN model into a different kind of high-level Petri net, called Recursive Extended Concurrent Algebraic Term Net (RECATNet), typically used to model dynamic systems with abstraction and recursion capabilities. RECATNet semantics can be expressed in rewriting logic, allowing properties

expressed as LTL formulas to be checked using Maude. This approach supports specific BPMN mechanisms typically not supported by other Petri net-based techniques, such as exceptions, cancellations and multi-instance sub-processes. The effectiveness of this technique is demonstrated through case studies.

Kog *et al.* (2012) suggest an approach for verifying construction industry processes using a transformation from BPMN to low-level Petri net. However, the details regarding the transformation and the verification are omitted.

Lam (2010) showcases a mapping from BPMN to the New Symbolic Model Verifier (NuSMV) language is showcased, expressed in terms of formal mathematical rules which establish a correspondence between BPMN elements and NuSMV code. This mapping includes a large subset of BPMN elements, only leaving out elements with a higher level of complexity, such as OR-gateways and compensation handlers. A case study is included.

Meghzili *et al.* (2020) present a BPMN-to-CPN transformation technique which makes use of the GRaphs for Object-Oriented VErification (GROOVE) graph transformation tool. A graph grammar allows this transformation to be executed automatically, covering complex BPMN mechanisms such as OR-gateways, multi-instance activities, sub-processes and message flows. The correctness of the transformation is verified using the GROOVE LTL model checker. The GROOVE representation of the CPN is then converted to an XML notation, where desired properties can be verified using CPN Tools. This method is illustrated using a case study.

Mendoza *et al.* (2010) transform BPMN models into the Communicating Sequential Processes + Time (CSP + T) process algebra, allowing the imposition of time-related constraints that enable the transformation of BPMN models with Timer events. This technique is a Formal Compositional Verification Approach (FCVA) based on the composition of all parallel components. Properties can be specified and verified using the Clocked Computation Tree Logic (CCTL) language. No evaluation is performed.

Similarly, Wong and Gibbons (2011) propose a representation of BPMN semantics using Communicating Sequential Processes (CSP) and extends it to enable the modeling of timing constraints. This bidirectional transformation is implemented using Haskell. A specification language, called PL, is also presented, which can be used to specify behavioral properties that are then translated to LTL. Examples are provided in the study but without evaluation.

Nazaruka *et al.* (2016) present a method of transforming a BPMN model into a Topological Functioning Model (TFM), a directed graph which depicts causal relations between vertices that represent functional characteristics of a system. The resulting TFM is then used to verify properties related to the problem domain. However, the authors do not explain how to perform the actual verification. An example is shown, but no evaluation.

Szpyrka *et al.* (2017) demonstrate a mapping from BPMN to the Alvis language, typically used to model concurrent systems, which can be used for visualization and model-checking. The Alvis XML file is then converted to Haskell. Users can add extra Haskell code if they wish to do so. Finally, this code can then be compiled using the GHC (Glasgow Haskell Compiler) compiler, which outputs a Labeled Transition System (LTS) graph. This structure can be used to verify if specific properties specified as μ-calculus formulas are satisfied using the CADP (Construction and Analysis of Distributed Processes) tool. The main advantage of this approach compared to the previous ones mentioned is the visual similarity between BPMN models and Alvis models. However, there is no evaluation.

Yamasathien and Vatanawood (2014) present a mapping from BPMN to the Process Meta Language (PROMELA) language. Simple BPMN constructs, such as exclusive choices and parallel splits, are identified in the model and translated to PROMELA code. Properties can then be verified using the SPIN (Simple Promela Interpreter) model checker. Only elementary BPMN elements and constructs are supported, data fields are not considered and no evaluation is presented.

Standing out from the rest of the techniques seen thus far, an automatic transformation technique is presented by Dechsupa *et al.* (2018, 2019, 2021) for transforming BPMN models

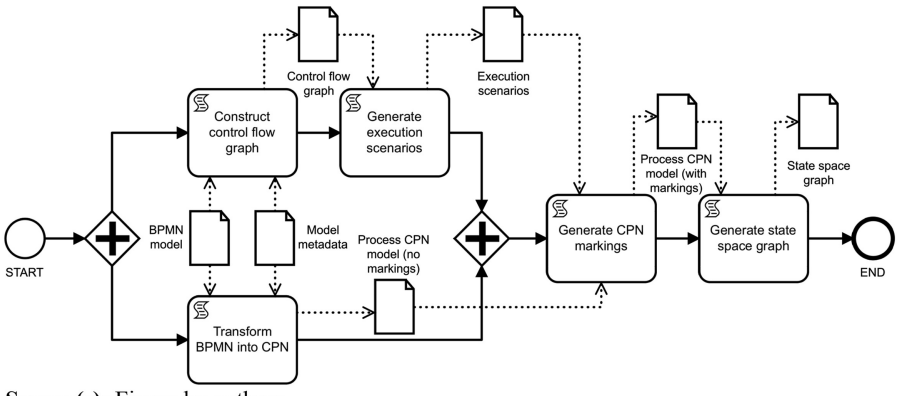
into CPNs. Firstly, the BPMN model is partitioned and a set of rules establish a mapping between BPMN and CPN constructs (Dechsupa *et al.*, 2018). It is important to note that this mapping encompasses many BPMN elements that other approaches do not have a mapping for, such as OR-gateways, errors and multi-instance activities. Then, the same authors use an extension of this mapping for formal verification purposes using state-space analysis (Dechsupa *et al.*, 2019). Finally, the authors present a complete automated framework for BPMN model verification, combining the BPMN-to-CPN transformation presented in their previous work with a BPMN-to-CFG transformation (Dechsupa *et al.*, 2021). The CFG is used to determine all possible execution paths; each execution path is used to create the necessary CPN markings automatically; finally, these are used to generate a complete state-space graph. Refer to Figure 9 for a simple visualization of this portion of the framework as a BPMN diagram.

The global state-space graph is ultimately used in a model checker to check desirable properties expressed by the user in Clocked Computation Tree Logic (CCTL). This approach was amply tested through several experiments and case studies. Despite the hindrance posed by the need for business process modelers and engineers to learn a new logic formula language to make use of this technique and despite the absence of a specific method for test case generation, this approach proved to distinguish itself from others due to the potential for combination with approaches from other studies (cf. Jahan *et al.*, 2016) in order to reach a full, complete solution to the problem at hand which combines test case generation and formal verification while also encompassing almost all BPMN elements.

3.6.5 Other. The following studies cover techniques which have distinct main goals, but were still deemed relevant for this review (Braghetto *et al.*, 2011; Böhmer and Rinderle-Ma, 2016; Dijkman and van Gorp, 2010; Rachdi *et al.*, 2016).

Braghetto *et al.* (2011) present an algorithm to convert BPMN models to Stochastic Automata Network (SAN) models, a formalism that enables performance evaluation of parallel and distributed systems. The algorithm functions based on a set of rules that map small BPMN constructs to SAN constructs. After conversion, a different set of procedures is used to reduce and simplify the SAN model obtained. The authors illustrate how this algorithm can be used to analyze execution times and resource utilization rates when executing multiple instances of the process in parallel.

In the publication authored by Böhmer and Rinderle-Ma (2016), a genetic algorithm is presented for automatic business process test case selection based on a set of custom



Source(s): Figure by authors
Adapted from Dechsupa *et al.* (2021, p. 624)

Figure 9.
BPMN verification approach

coverage metrics. These metrics are supported by historical data, such as node execution frequency and previously identified faults, an innovative approach not found in any other study. A random set of test suite configurations is generated and a genetic algorithm is applied to maximize a given fitness function. This approach was evaluated through a comparison to random and adaptive greedy selection techniques. However, the state explosion problem (a frequent motivator of test case selection) is not as common in the field of BPT when compared to, for instance, traditional software testing. Business process models often have significantly fewer possible execution paths in comparison, which mitigates the effects of potential exponential growth of test case suite size and testing time (Böhmer and Rinderle-Ma, 2015). This is particularly true for the case of BPMN models: visual clarity is critical, so models typically have a low number of elements so that they are easily visualizable. This means that techniques focused on reducing and selecting test cases may not bring much value to the current research.

Dijkman and van Gorp (2010) use graph rewrite rules to mimic BPMN semantics. This technique is used in the context of a tool used for verification of workflow engine conformance to the official BPMN specification. This verification tool determines possible execution paths, along with the possible values for the required data fields and compares the behavior of the workflow engine with the resulting graph transformations to assess the correctness of the engine. However, the implementation of the tool itself is left for future work.

Finally, Rachdi *et al.* (2016) present a formal method for verification of business rules which represent specific properties related to resources (data), tasks (activities and events), agents (pools and lanes) and time (assigned activity duration). A language is showcased, named Business Rule Language (BRL), to formally define business rules. A DFS-like algorithm is applied to a BPMN file to traverse the model and verify a set of business rules, which makes use of a trace mechanism to deal with loops. However, no evaluation is performed.

3.7 Discussion

The selected studies include several different approaches to business process model verification and testing, as well as possible ways to use process model verification techniques to complement testing. Plenty of test generation techniques were also presented, differing, for instance, in terms of coverage criteria and data awareness.

The 32 papers were grouped according to the type of testing approach and the language used to represent the processes. A taxonomy made up of five disjoint groups was created, which can be found in Table 2 alongside the documents contained in each one.

This taxonomy covers four distinct types of approaches, corresponding to Groups 1–4.

(1) *Group 1:* The variety of techniques presented for BPMN model testing allows a potential user to choose the most appropriate approach considering the desired coverage criteria, BPMN elements to support and awareness of test data. The existence of approaches that do not make use of intermediate models (see de Moura *et al.*, 2017; Paiva *et al.*, 2018; Schneid *et al.*, 2021) is adequate when dealing with rapidly changing process models to avoid having intermediate representations of the business process that do not reflect the latest changes. Test cases can be represented in numerous formats and automatic execution of the tests is also possible. The only major downside to this type of approach is that most techniques that utilize it only support a limited subset of all BPMN elements, given the difficulty associated with determining possible paths when dealing with models with complex elements.

(2) *Group 2:* Five documents using three distinct testing approaches were covered: metaheuristic algorithms, transformation to graph-based structures and mapping to UML Activity Diagram. Despite the language difference, the techniques presented for

test case generation from BPEL processes that make use of auxiliary structures can also be applied in the context of BPMN model testing, if used in combination with mappings from BPMN to these structures.

(3) *Group 3:* The documents analyzed that use other ways to represent business process models for testing do not contain any particularly novel or innovative testing techniques when compared to Group 1 and Group 2 documents.

(4) *Group 4:* Being the largest of the five groups, the field of BPMN formal verification seems considerably robust, allowing the verification of formal structural and behavioral properties through various techniques, most of them based on temporal logic and model-checking. The usage of older mathematical concepts and structures that have been studied extensively in the past, along with the work that has been done to support more and more elements of the BPMN standard, has greatly solidified this field of research.

Group 5 contains documents that did not fit into any of these four groups due to differences regarding the end goal of the studies, but were still classified as relevant concerning the topic being covered.

3.8 Findings

The aggregation and summarizing of all the information related to BPT present in these documents makes it possible to tackle the established research questions.

3.8.1 Types of business process testing.

RQ1. Which types of process model testing approaches are currently being used by process modelers and business analysts for BPT?

The techniques analyzed throughout the review prompted the creation of a simple classification scheme for types of functional BPT, as follows.

(1) *Black/gray-box testing:* used to check if the process meets the necessary functional requirements through comparison of obtained outputs to expected outputs (de Moura *et al.*, 2017).

(2) *Regression testing:* used to check if the process still behaves as expected for specific scenarios after the process is changed.

(3) *Integration testing:* used to test interactions between the underlying services throughout the execution of a process.

Table 3 shows the types of testing covered in the documents of Groups 1, 2 and 3.

Outside the first three groups, Braghetto *et al.* (2011) also cover (non-functional) *performance testing* applied to business processes.

RQ1.1. What are the practical applications of each type of business process model testing?

Each type of testing serves a different purpose and is meant to be used in distinct situations, as follows.

(1) *Black/gray-box testing:* this type of testing was solely seen being applied to processes represented in BPMN. Black-box and gray-box testing are helpful when first testing new processes after they are modeled, enabling the generation of test cases which allow for path-level testing to attest its conformity to functional requirements.

(2) *Regression testing:* this type of testing was also used solely with BPMN processes. It is helpful when dealing with already implemented processes that are under constant

Types of testing	Documents	# Docs
Black/gray-box testing	Buchs *et al.* (2009) de Moura *et al.* (2017) Seqerloo *et al.* (2019) Yotyawilai and Suwannasart (2014)	4
Regression testing	Makki *et al.* (2017) Schneid *et al.* (2021)	2
Integration testing	Blanco *et al.* (2009) Nahak *et al.* (2019) Ma *et al.* (2008) Jahan *et al.* (2016) Guangquan *et al.* (2007) Yuan *et al.* (2008)	6
N/A	Paiva *et al.* (2018) Bures *et al.* (2017)	2

Source(s): Table by authors

Table 3. Groups 1–3 document classification according to the type of functional BPT covered

change, allowing the automatic detection of undesired changes to the behavior of the process (Makki *et al.*, 2017).

(3) *Integration testing:* used primarily to test Web service compositions and choreographies, integration testing was frequently used when dealing with processes represented in the BPEL language. It enables the testing of the behavior of all web services orchestrated by a BPEL process (Jehan *et al.*, 2015) within an organization or between different organizations (Jahan *et al.*, 2016).

3.8.2 BPMN testing and verification approaches.

RQ2. Which approaches are currently in use by process modelers and business analysts for testing and verification of BPMN models?

The seven Group 1 documents contained distinct approaches for BPMN model testing. These approaches vary significantly between them regarding BPMN elements supported, coverage criteria, test data generation, auxiliary representations and overall objective. A summary of these differences can be seen in Table 4.

Reference	BPMN elements	Coverage[1]	Test data[2]	Auxiliary representation[3]
Buchs *et al.* (2009)	(unstated)	Custom		Algebraic Petri Net
Makki *et al.* (2017)	Comprehensive	N/A		–
de Moura *et al.* (2017)	Basic (w/OR)	Path		–
Paiva *et al.* (2018)	Limited	Path		–
Schneid *et al.* (2021)	Limited	Path		–
Seqerloo *et al.* (2019)	Basic	Path		State graph
Yotyawilai and Suwannasart (2014)	Basic	Path		Flow graph

Note(s): [1] Coverage criteria used to generate test cases
[2] Whether or not test data is generated for the test cases alongside the flow paths to follow
[3] Intermediate auxiliary representation that the original BPMN model is transformed into
Source(s): Table by authors

Table 4. Comparison of BPMN test case generation approaches

The decision of which technique or techniques to use may depend on several different criteria. For instance, organizations with significantly complex processes that contain many possible flow paths may opt for a testing method that utilizes a non-comprehensive coverage criterion when generating test cases to prevent testing times from rising to insurmountable levels. Organizations with dynamic, constantly-changing processes may want to choose a technique that does not make use of any auxiliary, intermediate representations to prevent discrepancies between the models in production and the models under test. Alternatively, organizations using process models that contain elements with high semantic complexity may opt to redesign their business processes or refactor their models to replace complex elements with simpler ones, enabling testing.

Group 4 documents covered ways to formally verify structural and behavioral properties of BPMN models. In particular, documents published from 2017 onward proved to be quite solid, tackling the problem based on similar logical foundations and encompassing a large number of BPMN elements. Table 5 shows the existing differences between the recently-published formal verification approaches. This table also presents the amount of theoretical background required to use such approaches, showing the difficulties associated with these techniques that a typical process modeler or business analyst may face.

It is worth mentioning that the techniques described in Group 2 also show significant potential for usage with BPMN models when combined with BPMN model transformations to specific auxiliary structures. One example of how this could be accomplished is by combining the CFG and CPN-based algorithms presented by Jahan *et al.* (2016) with mappings that transform BPMN into these two representations (see Dechsupa *et al.*, 2018 and Meghzili *et al.*, 2020, respectively).

RQ2.1. What impact do said approaches have on testing efficiency?

Although none of the documents analyzed testing times in comparison to manual testing, plenty of documents featured an evaluation section which showed very positive results.

Focusing on Group 1, the evaluation results of Schneid *et al.*'s (2021) approach show that processing times for the tool stay under 100 ms for processes with over 300 flows and several loops, with 90% of test subjects answering "Agree" or "Strongly Agree" (levels 4 and 5 of a five-level Likert scale) to the question "Using the [prototype] saves a lot of time when creating

Reference	Logic system[1]	Languages and systems[2]	Auxiliary representations[3]
Corradini *et al.* (2018)	LTL[a]	Maude	–
Dechsupa *et al.* (2018, 2019, 2021)	CTL[b]	–	CPN[c], CFG[d]
Durán *et al.* (2018)	Rewriting logic	Maude	–
Kheldoun *et al.* (2017)	LTL[a]	Maude	RECATNet[e]
Meghzili *et al.* (2020)	LTL[a]	–	CPN[c]
Szpyrka *et al.* (2017)	μ-calculus	Alvis, Haskell	LTS[f] graph

Note(s): [1] System of logical rules serving as the basis for the solution
[2] Software languages and software systems used
[3] Intermediate auxiliary representations that the original BPMN model is transformed into
[a] Linear Temporal Logic
[a] Computation Tree Logic
[c] Colored Petri Net
[d] Control Flow Graph
[e] Recursive Extended Concurrent Algebraic Term Net
[f] Labeled Transition System
Source(s): Table by authors

Table 5. Comparison of BPMN formal verification approaches published during or after 2017

tests". Additionally, the regression detection mechanism presented by Makki *et al.* (2017) posed an average performance overhead to the workflow engine of just 3.9%.

Even without an explicit comparison, using these techniques is, evidently, a significant improvement to manual BPT, which can take multiple hours or days for a single process with several dozen flow paths.

Seqerloo *et al.* (2019) also show that even for processes with fewer than 20 possible flows, domain experts struggle to manually determine all possible paths, with path determination precision and recall averages across eight different processes sitting at 0.76 and 0.78, respectively. By contrast, the solution showcased in the same study had an average path determination precision and recall of 0.94 and 0.96, respectively.

In short, most of the solutions analyzed in this review have a significant positive impact on BPT efficiency while also improving overall accuracy.

RQ2.2. How can existing verification techniques complement BPMN testing?

The formal verification approaches analyzed in Group 4 can complement testing by being used in combination with testing techniques. By combining testing and verification, modelers and analysts can ensure the model is not only behaviorally correct, but also well-formed, not containing deadlocks, livelocks or infinite loops.

These techniques show the most potential when combined with other methods based on transition-based formalisms (see Dechsupa *et al.*, 2018, 2019, 2021). Using verification approaches based on intermediate transition-based structures in combination with test generation techniques that rely on the same formalisms enables the simultaneous use of model-checking and MBT.

Additionally, some of the verification techniques described may also be used directly for testing purposes. This could be accomplished through the generation of counter-examples based on negated properties (Böhmer and Rinderle-Ma, 2015). The way to accomplish this would depend on different aspects, such as the techniques used throughout the transformation, the language the model is transformed into and the logic system used to verify specific properties.

Even so, there are serious disadvantages regarding the difficulty in property formulation and the theoretical background required to use these techniques.

3.9 Testing framework

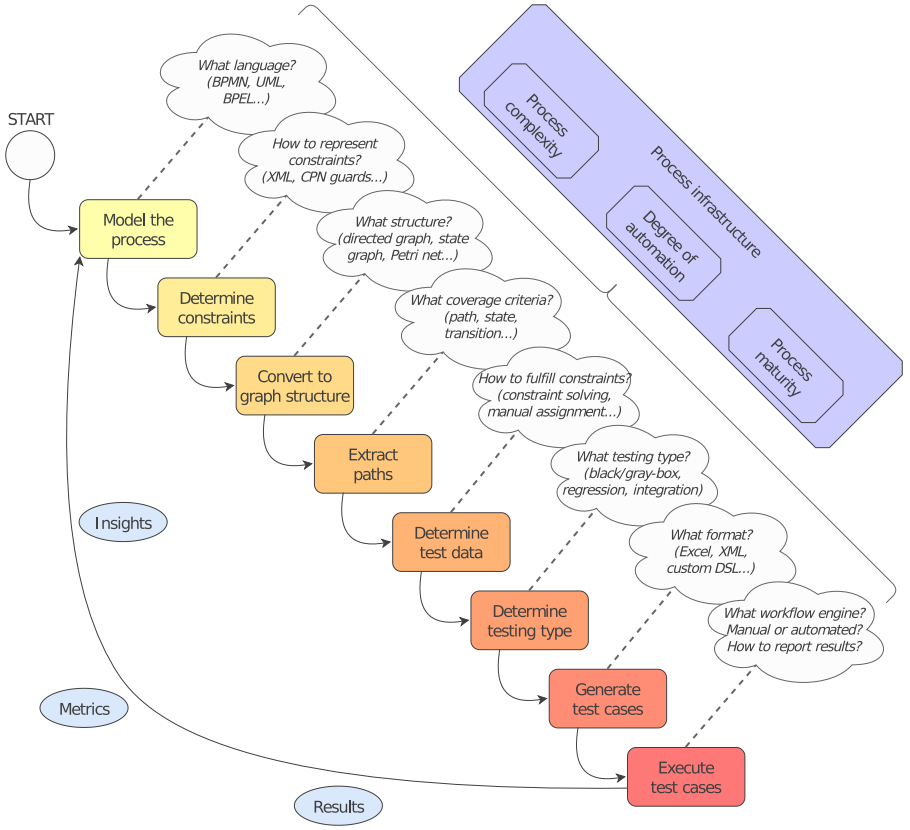
Based on the analyzed studies, a framework was developed for continuous BPT. This framework was derived from the common aspects of different BPT approaches studied throughout the literature review.

The Business Process Evaluation and Research Framework for Enhancement and Continuous Testing (bPERFECT) framework, at its core, consists of an iterative chain of high-level directives that aim to guide the development of future BPT approaches and serve as the foundation for future research in the field. A graphical representation of this framework can be found in Figure 10.

The framework describes the coarse-grained steps that typically go into creating or implementing a new BPT solution, the decisions that must be taken before commencing each step and some of the factors that influence said decisions. The framework also showcases how testing can be used to make changes, corrections and enhancements to the process.

The following bullets explain the steps included in the bPERFECT framework in greater detail, alongside illustrative examples of potential solutions for each step.

(1) *Model the process*: if not already modeled, create a model of the process to be tested using a process modeling language. Already existing models may be used or adapted as well. e.g.: model the process using BPMN.



Source(s): Figure by authors

Figure 10.
The bPERFECT framework

(2) *Determine constraints*: determine the constraints that dictate the flow of the process. e.g.: specify the constraints next to each XOR-split in the model.

(3) *Convert to graph structure*: convert the process to a graph-based structure that can be used to extract the possible process paths/flows. Different types of graphs provide different capabilities regarding essential aspects such as constraint representation and behavioral complexity. Alternatively, one may model the process directly as a graph, removing the need for this step. e.g.: apply the BPMN-to-flow-graph transformation presented by Seqerloo *et al.* (2019).

(4) *Extract paths*: apply a DFS-like algorithm to extract the process paths for which test cases are to be generated according to the specified coverage criteria. A test case selection algorithm may also be used to reduce the size of the test case suite for processes with an insurmountable amount of possible flows, e.g. apply a DFS to the flow graph to extract paths using path coverage criteria, as presented by Seqerloo *et al.* (2019).

(5) *Determine test data*: determine possible values for the variables associated with each path that fulfill the associated constraints. For simple constraints, manual

assignment of values is sufficient. A more sophisticated solution may be required for more complex constraints with distinct data types. e.g.: use a constraint solver to determine possible values for the test variables that fulfill all constraints for each path, as presented by Jahan *et al.* (2016) and suggested by Schneid *et al.* (2021).

(6) *Determine testing type*: choose the type of testing desired—black/gray-box, regression, or integration—depending on the motives behind testing. e.g.: opt to perform black/gray-box testing, as presented by Buchs *et al.* (2009), de Moura *et al.* (2017), Seqerloo *et al.* (2019), Yotyawilai and Suwannasart (2014).

(7) *Generate test cases*: generate the test cases, one per extracted path, considering each one's constraints and previously assigned variable values. e.g.: generate one table for each test case containing the process path to follow, required inputs and variable values and expected outputs, similarly to Yotyawilai and Suwannasart (2014).

(8) *Execute test cases*: execute the generated test cases. This step involves simulating the process using a workflow engine of choice, with varying degrees of automation. e.g.: generate executable code for the Camunda BPM engine, as presented by Schneid *et al.* (2021), based on the tabular test cases, to simulate each test case and compare obtained outputs to expected outputs.

Each decision taken throughout the testing procedure is influenced by a multitude of factors related to process infrastructure, such as.

(1) *Process complexity*: high process complexity impacts modeling efforts and often results in a very high number of flows, resulting in an inability to generate/execute test cases for all possible flows in a timely manner. Subdividing the process into several smaller models or using less comprehensive coverage criteria may facilitate testing.

(2) *Degree of automation*: the level of automation associated with each of the testing steps, as well as the level of automation associated with the business process itself, may be a key factor when making path extraction, test case generation and test case execution decisions.

(3) *Process maturity*: special precautions may have to be taken depending on how well defined the business processes are, how strictly they are followed and how robust they are when dealing with exceptions, edge cases, back-tracking and cancellation scenarios which may influence the testing procedure as a whole.

The results of test execution allow for functional assessment of the process and can be used to compute metrics and derive insights that can be used to modify the process, which leads back to the first step. This systematization of process assessment enables the implementation of continuous process re-engineering improvement practices using a test-centered approach.

3.10 Final remarks

As new regulatory norms and technological advancements lead to continuously growing business complexity (Paiva *et al.*, 2018), the need for a robust process infrastructure becomes increasingly present. As competition grows, ensuring high-quality outputs while maintaining internal efficiency is a critical priority for businesses, resulting in growing efforts from organizations to continuously evaluate and improve internal operations and processes (Siha and Saad, 2008).

The noticeable increase in published studies on assessing business processes in recent years reflects this growth in interest. Notably, regarding BPMN model testing specifically, 5

of the 7 Group 1 documents included in the present review were published within the last five years. Additionally, compared to the literature review on process model testing authored by Böhmer and Rinderle-Ma (2015), there was a significant boost in recent BPMN-centric process studies, which previously tended to focus mainly on the BPEL language. This tendency is expected to continue as more and more businesses opt to use more modern modeling languages over other alternatives.

However, for the most part, this increase in publications has not resulted in a faster rate of growth and innovation in the field. Namely, a common trend among the different testing solutions analyzed was a lack of shared knowledge, with each testing solution typically being developed in an ad hoc fashion. Additionally, different terms were often used to refer to equal or similar concepts, which may create confusion.

The contributions presented throughout this study aim to shift this tendency and assist in bridging existing gaps between BPT researchers and practitioners by establishing a solid but adaptable baseline that can be built upon in future work. An overview can be found in Figure 11. These can be summed up as four distinct outputs.

(1) BPT study synthesis/Creation of a BPT knowledge base,

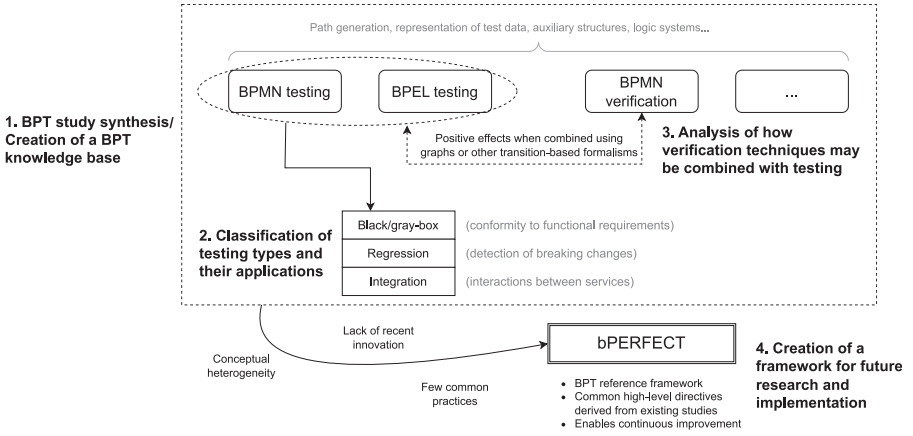
(2) Classification of testing types and their applications,

(3) Analysis of how verification techniques may be combined with testing and

(4) Creation of a framework for future research and implementation.

In the long run, these contributions aim to remove some of the currently existing barriers associated with the implementation of BPT techniques and practices in today's organizations and allow said organizations to continuously adapt and deal with ever-growing complexity while ensuring processes stay compliant and continue to bring value to the business. Effective and efficient development and implementation of such testing solutions will, ultimately, allow organizations to improve their processes, increase internal efficiency and reduce costs.

Three main research gaps have been identified, which may be further explored with the help of the bPERFECT framework.



Source(s): Figure by authors

Figure 11. Contributions overview

(1) There are no complete process model testing solutions that support the entire BPMN standard, which may lead to significant process model refactoring efforts and may even preclude testing as a whole in the case of very complex processes where refactoring may be unfeasible.

(2) There are no studies that tackle platform cross-compatibility and interoperability, limiting the potential practical applications of existing solutions.

(3) The field of formal verification of business processes is quite solidified, with most approaches being based on techniques, languages, structures and mathematical frameworks that have been extensively studied for dozens of years. However, a complete process assessment solution that seamlessly combines testing and verification still does not exist.

Despite the contributions mentioned, the review conducted posed some limitations. Namely, the selected databases may have limited the document pool's breadth. Relevant studies may have been left out due to them not being indexed in the collection used for the search or due to the inclusion criteria and query strings utilized.

Notwithstanding the limitations, this literature review still constitutes a strong baseline for organizations and researchers alike, being capable of serving as a knowledge base for BPMN model testing and BPT as a whole while also highlighting tendencies and research gaps.

4. Conclusion

This paper reviews the literature associated with testing and verification of business processes, with a particular emphasis on BPMN, motivated by a growing need to continually assess the structure and behavior of BPMN processes to ensure they stay stable, continue to meet business requirements and are compliant to regulatory norms.

Several BPT approaches were explored, which tackled the problem from different perspectives and with varying degrees of automation, shown to improve testing efficiency greatly. Different coverage criteria were used to create test sets. Additionally, several mathematical structures and frameworks were shown to facilitate verification significantly.

This review constitutes a vast body of information for BPMN model testing and verification, adequately achieving the established objective of aggregating and synthesizing existing knowledge regarding process model testing to guide future research and facilitate business process assessment in organizations.

Namely, the classification system proposed for BPT approaches reduces conceptual heterogeneity in process testing, alleviating confusion among practitioners and organizations. Additionally, the proposed bPERFECT framework seeks to guide future research in this field, constituting a solid starting point for further innovation and automation.

Ultimately, this literature review and the bPERFECT framework eliminate many obstacles that often impede adequate BPT, allowing organizations to improve their processes and produce higher-quality results, increase productivity and decrease expenses.

As the BPT field continues to advance, it is crucial that future research focuses on tackling the described research gaps – such as the lack of support for the entire BPMN standard or the poor workflow engine cross-compatibility – and addressing the limitations of existing approaches. Exploring these concerns is vital to making BPT more accessible to organizations, allowing them to use its cost-saving and productivity-boosting potential fully.

Note

1. Web of Science: https://www.webofscience.com/

References

Beerepoot, I., Di Ciccio, C., Reijers, H.A., Rinderle-Ma, S., Bandara, W., Burattin, A., Calvanese, D., Chen, T., Cohen, I., Depaire, B., Di Federico, G., Dumas, M., van Dun, C., Fehrer, T., Fischer, D.A., Gal, A., Indulska, M., Isahagian, V., Klinkmüller, C., Kratsch, W., Leopold, H., Van Looy, A., Lopez, H., Lukumbuzya, S., Mendling, J., Meyers, L., Moder, L., Montali, M., Muthusamy, V., Reichert, M., Rizk, Y., Rosemann, M., Röglinger, M., Sadiq, S., Seiger, R., Slaats, T., Simkus, M., Someh, I.A., Weber, B., Weber, I., Weske, M. and Zerbato, F. (2023), "The biggest business process management problems to solve before we die", *Computers in Industry*, Vol. 146, 103837, doi: 10.1016/j.compind.2022.103837.

Blanco, R., García-Fanjul, J. and Tuya, J. (2009), "A first approach to test case generation for BPEL compositions of web services using Scatter Search", *Second International Conference on Software Testing Verification and Validation, ICST 2009*, April 1-4, 2009, IEEE, Denver, CO, Workshops Proceedings, pp. 131-140, doi: 10.1109/ICSTW.2009.24.

Böhmer, K. and Rinderle-Ma, S. (2015), "A systematic literature review on process model testing: approaches, challenges, and research directions". doi: 10.48550/arXiv.1509.04076.

Böhmer, K. and Rinderle-Ma, S. (2016), "Automatic business process test case selection: coverage metrics, algorithms, and performance optimizations", *International Journal of Cooperative Information Systems*, Vol. 25 No. 4, pp. 1740002.1-1740002.36, doi: 10.1142/S0218843017400020.

Booth, A. (2006), "Clear and present questions: formulating questions for evidence based practice", *Library Hi Tech*, Vol. 24 No. 3, pp. 355-368, doi: 10.1108/07378830610692127.

Braghetto, K.R., Ferreira, J.E. and Vincent, J.-M. (2011), "Performance evaluation of business processes through a formal transformation to SAN", Proceedings, *Computer Performance Engineering: 8th European Performance Engineering Workshop, EPEW 2011*, October 12-13, 2011, Springer, Borrowdale, pp. 42-56, doi: 10.1007/978-3-642-24749-1_5.

Buchs, D., Lucio, L. and Chen, A. (2009), "Model checking techniques for test generation from business process models", *Reliable Software Technologies – Ada-Europe 2009: 14th Ada-Europe International Conference on Reliable Software Technologies*, Brest, June 8-12, 2009, Springer, Proceedings, pp. 59-74, doi: 10.1007/978-3-642-01924-1_5.

Bures, M., Cerny, T. and Klima, M. (2017), "Prioritized Process Test: more efficiency in testing of business processes and workflows", in *Information Science and Applications 2017: ICISA 2017*, Springer, Macau, China, pp. 585-593, doi: 10.1007/978-981-10-4154-9_67.

Clarivate (2021), "Web of science coverage details - resources for librarians", Archived Using Wayback Machine, October 27, 2021, available at: https://web.archive.org/web/20211027164550/https://clarivate.libguides.com/librarianresources/coverage (accessed 10 October 2022).

Corradini, F., Fornari, F., Polini, A., Re, B. and Tiezzi, F. (2018), "A formal approach to modeling and verification of business process collaborations", *Science of Computer Programming*, Vol. 166, pp. 35-70, doi: 10.1016/j.scico.2018.05.008.

Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F. and Vandin, A. (2021), "A formal approach for the analysis of BPMN collaboration models", *Journal of Systems and Software*, Vol. 180, 111007, doi: 10.1016/j.jss.2021.111007.

de Moura, J.L., Charão, A.S., Lima, J.C.D. and de Oliveira Stein, B. (2017), "Test case generation from BPMN models for automated testing of web-based BPM applications", *2017 17th International Conference on Computational Science and its Applications (ICCSA 2017)*, IEEE, Trieste, pp. 1-7, doi: 10.1109/ICCSA.2017.7999652.

Dechsupa, C., Vatanawood, W. and Thongtak, A. (2018), "Transformation of the BPMN design model into a colored Petri net using the partitioning approach", *IEEE Access*, Vol. 6, pp. 38421-38436, doi: 10.1109/ACCESS.2018.2853669.

Dechsupa, C., Vatanawood, W. and Thongtak, A. (2019), "Hierarchical verification for the BPMN design model using state space analysis", *IEEE Access*, Vol. 7, pp. 16795-16815, doi: 10.1109/ACCESS.2019.2892958.

Dechsupa, C., Vatanawood, W. and Thongtak, A. (2021), "An automated framework for BPMN model verification achieving branch coverage", *Engineering Journal-Thailand*, Vol. 25 No. 2, pp. 135-150, doi: 10.4186/ej.2021.25.2.135.

Dijkman, R. and van Gorp, P. (2010), "BPMN 2.0 execution semantics formalized as graph rewrite rules", Proceedings, *Business Process Modeling Notation: Second International Workshop, BPMN 2010*, Potsdam, October 13-14, 2010, Springer, pp. 16-30, doi: 10.1007/978-3-642-16298-5_4.

Dumas, M., Rosa, M.L., Mendling, J. and Reijers, H.A. (2018), *Fundamentals of Business Process Management*, Springer, Berlin.

Durán, F., Rocha, C. and Salaün, G. (2018), "Symbolic specification and verification of data-aware BPMN processes using Rewriting Modulo SMT", Proceedings, *Rewriting Logic and Its Applications: 12th International Workshop, WRLA 2018, Held as a Satellite Event of ETAPS*, Thessaloniki, June 14-15, 2018, Springer, pp. 76-97, doi: 10.1007/978-3-319-99840-4_5.

Guangquan, Z., Mei, R. and Jun, Z. (2007), "A business process of web services testing method based on UML2.0 Activity Diagram", *Workshop on Intelligent Information Technology Application (IITA 2007)*, IEEE, Zhangjiajie, pp. 59-65, doi: 10.1109/IITA.2007.83.

Guerreiro, S. (2020), "Conceptualizing on dynamically stable business processes operation: a literature review on existing concepts", *Business Process Management Journal*, Vol. 27 No. 1, pp. 24-54, doi: 10.1108/BPMJ-02-2020-0072.

Hutchinson, J., Rouncefield, M. and Whittle, J. (2011), "Model-driven engineering practices in industry", *'ICSE '11: Proceedings of the 33rd International Conference on Software Engineering'*, ACM, Waikiki, Honolulu, pp. 633-642, doi: 10.1145/1985793.1985882.

Ipate, F. and Banica, L. (2007), "W-method for hierarchical and communicating finite state machines", *2007 5th IEEE International Conference on Industrial Informatics*, IEEE, Vienna, pp. 891-896, doi: 10.1109/INDIN.2007.4384891.

Ipate, F. and Holcombe, M. (2008), "Testing data processing-oriented systems from Stream X-machine models", *Theoretical Computer Science*, Vol. 403 No. 2, pp. 176-191, doi: 10.1016/j.tcs.2008.02.045.

Jahan, H., Rao, S. and Liu, D. (2016), "Test case generation for BPEL-based web service composition using colored Petri nets", *2016 International Conference on Progress in Informatics and Computing (PIC 2016)*, IEEE, Shanghai, China, pp. 623-628, doi: 10.1109/PIC.2016.7949575.

Jehan, S., Pill, I. and Wotawa, F. (2015), "BPEL integration testing", Proceedings, *Fundamental Approaches to Software Engineering: 18th International Conference, FASE 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015*, London, April 11-18, 2015, Springer, pp. 69-83, doi: 10.1007/978-3-662-46675-9_5.

Kheldoun, A., Barkaoui, K. and Ioualalen, M. (2017), "Formal verification of complex business processes based on high-level Petri nets", *Information Sciences*, Vol. 385, pp. 39-54, doi: 10.1016/j.ins.2016.12.044.

Kitchenham, B. (2004), "Procedures for performing systematic reviews", Technical report, Keele University, Keele, Staffs, available at: https://www.inf.ufsc.br/%7Ealdo.vw/kitchenham.pdf

Kog, F., Scherer, R.J. and Dikbas, A. (2012), "Petri net based verification of BPMN represented configured construction processes", in *ECCPM 2012 - eWork and eBusiness in Architecture, Engineering and Construction*, CRC Press, Reykjavik, pp. 243-249.

Lam, V.S.W. (2010), "Formal analysis of BPMN models: a NuSMV-based approach", *International Journal of Software Engineering and Knowledge Engineering*, Vol. 20 No. 7, pp. 987-1023, doi: 10.1142/S0218194010005079.

Link Consulting (2016), "Edoclink white paper", available at: https://pdfhost.io/v/nsLgUFJTc_edoclink_white_paper_2016

Ma, C., Wu, J., Zhang, T., Zhang, Y. and Cai, X. (2008), "Testing BPEL with Stream X-machine", *2008 International Symposium on Information Science and Engineering (ISISE)*, IEEE, Shanghai, pp. 578-582, doi: 10.1109/ISISE.2008.201.

Makki, M., van Landuyt, D. and Joosen, W. (2017), "Automated regression testing of BPMN 2.0 processes", *ACM SIGPLAN Notices*, Vol. 52 No. 3, pp. 178-189, doi: 10.1145/2993236.2993257.

Meghzili, S., Chaoui, A., Strecker, M. and Kerkouche, E. (2020), "An approach for the transformation and verification of BPMN models to colored Petri nets models", *International Journal of Software Innovation*, Vol. 8 No. 1, pp. 17-49, doi: 10.4018/IJSI.2020010102.

Mendoza, L.E., Capel, M.I. and Pérez, M. (2010), "Compositional verification of business processes by model-checking", *Modelling, Simulation, Verification and Validation of Enterprise Information Systems, Proceedings of the 8th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS 2010, in conjunction with ICEIS 2010*, Funchal, Madeira, June 2010, SciTePress, pp. 60-69, doi: 10.1109/CLEI.2013.6670616.

Nahak, S.K., Mohapatra, D.P. and Patra, M.R. (2019), "A new test case generation for web service choreography testing by using metaheuristic algorithm", *Soft Computing in Data Analytics: Proceedings of International Conference on SCDA 2018*, Springer, Chilakapalem, pp. 817-824, doi: 10.1007/978-981-13-0514-6_77.

Nazaruka, E., Ovchinnikova, V., Alksnis, G. and Sukovskis, U. (2016), "Verification of BPMN model functional completeness by using the Topological Functioning Model", *ENASE 2016: Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, SciTePress, Rome, pp. 349-358, doi: 10.5220/0005930903490358.

Object Management Group (2010), "Business process model and notation (BPMN), version 2.0" available at: https://www.omg.org/spec/BPMN/2.0

Paiva, A.C.R., Flores, N.H., Faria, J.P. and Marques, J.M.G. (2018), "End-to-end automatic business process validation", *Procedia Computer Science*, Vol. 130, pp. 999-1004, doi: 10.1016/j.procs.2018.04.104.

Rachdi, A., En-Nouaary, A. and Dahchour, M. (2016), "Verification of common business rules in BPMN process models", *Networked Systems: 4th International Conference, NETYS 2016, Marrakech, Morocco, May 18-20, 2016, Revised Selected Papers*, Springer, Marrakech, pp. 334-339, doi: 10.1007/978-3-319-46140-3_27.

Schieferdecker, I. (2012), "Model-based testing", *IEEE Software*, Vol. 29 No. 1, pp. 14-18, doi: 10.1109/MS.2012.13.

Schneid, K., Stapper, L., Thöne, S. and Kuchen, H. (2021), "Automated regression tests: a no-code approach for BPMN-based Process-Driven Applications", *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE, Gold Coast, pp. 31-40, doi: 10.1109/EDOC52215.2021.00014.

Seqerloo, A.Y., Amiri, M.J., Parsa, S. and Koupaee, M. (2019), "Automatic test cases generation from business process models", *Requirements Engineering*, Vol. 24, pp. 119-132, doi: 10.1007/s00766-018-0304-3.

Siha, S.M. and Saad, G.H. (2008), "Business process improvement: empirical assessment and extensions", *Business Process Management Journal*, Vol. 14 No. 6, pp. 778-802, doi: 10.1108/14637150810915973.

Szpyrka, M., Nalepa, G.J. and Kluza, K. (2017), "From process models to concurrent systems in Alvis language", *Informatica*, Vol. 28 No. 3, pp. 525-545, doi: 10.15388/Informatica.2017.143.

Weidlich, M., Decker, G., Großkopf, A. and Weske, M. (2008), "BPEL to BPMN: the myth of a straightforward mapping", *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008*, Monterrey, November 9-14, 2008, Springer, Proceedings, Part I, pp. 265-282, doi: 10.1007/978-3-540-88871-0_19.

Wong, P.Y.H. and Gibbons, J. (2011), "Formalisations and applications of BPMN", *Science of Computer Programming*, Vol. 76 No. 8, pp. 633-650, doi: 10.1016/j.scico.2009.09.010.

Wynn, M.T., Verbeek, H.M.W., van der Aalst, W.M.P., ter Hofstede, A.H.M. and Edmond, D. (2009), "Business process verification – finally a reality!", *Business Process Management Journal*, Vol. 15 No. 1, pp. 74-92, doi: 10.1108/14637150910931479.

Yamasathien, S. and Vatanawood, W. (2014), "An approach to construct formal model of business process model from BPMN workflow patterns", *2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, IEEE, Bangkok, pp. 211-215, doi: 10.1109/DICTAP.2014.6821684.

Yotyawilai, P. and Suwannasart, T. (2014), "Design of a tool for generating test cases from BPMN", *2014 International Conference on Data and Software Engineering (ICODSE)*, IEEE, Bandung, pp. 1-6, doi: 10.1109/ICODSE.2014.7062692.

Yuan, Q., Wu, J., Liu, C. and Zhang, L. (2008), "A model driven approach toward business process test case generation", *2008 10th International Symposium on Web Site Evolution*, IEEE, Beijing, pp. 41-44, doi: 10.1109/WSE.2008.4655394.

Corresponding author
Tomás Lopes can be contacted at: tomas.lopes@tecnico.ulisboa.pt