Mining BPMN Processes on GitHub for Tool Validation and Development

Thomas S. Heinze[1(✉)], Viktor Stefanko[2], and Wolfram Amme[2]

[1] Institute of Data Science, German Aerospace Center (DLR), Jena, Germany
thomas.heinze@dlr.de
[2] Institute of Computer Science, Friedrich Schiller University Jena, Jena, Germany
{viktor.stefanko,wolfram.amme}@uni-jena.de

Abstract. Today, business process designers can choose from an increasing number of analysis tools to check their process model with respect to defects or flaws, before, e.g., deploying the model in a process engine. Answering questions about the tools' effectiveness though is difficult, as their validation often lacks empirical evidence. In particular, for a modeling language like BPMN, where the process is the product, tools are validated by means of case studies or even artificial process examples. We here advocate instead an approach to systematically mine software repositories on `GitHub.com` for a large corpus of BPMN business process models and discuss how it can be used for tool validation and guiding tool development, using the example of the linting tool BPMNspector.

1 Introduction

Mining software repositories, i.e., the systematic retrieval, processing and analysis of data about software artifacts and software development from software forges and repositories, has drawn considerable attention in recent years. On the one hand, the increasing popularity and usage of platforms like `GitLab.com`, `Bitbucket.org`, `SourceForge.net` or `GitHub.com` for collaborative software development provide a tremendous source of data, encompassing software from a rich and heterogeneous spectrum of domains. On the other hand, the data mining techniques available today have made it possible to start to seize this treasure, and thus allow to answer research questions and empirically validate hypotheses on the development and usage of IT and software systems based upon real-world data. While the research field is mainly focused on source code of conventional programming languages, mining software repositories can as well help to understand more about the use of other artifacts in software development [12].

In particular the area of modeling languages, such as the *Unified Modeling Language (UML)* or the *Business Process Model and Notation* (BPMN) [4], can benefit from a data-driven approach like mining software repositories. There is a common lack of larger datasets with real-world models, which hinders empirical research in this area [20,28,30]. Retrieving systematically a corpus of models by mining software repositories promises to overcome this lack. For instance,

S. Nurcan et al. (Eds.): BPMDS 2020/EMMSAD 2020, LNBIP 387, pp. 193–208, 2020.
https://doi.org/10.1007/978-3-030-49418-6_13

research questions on how modeling languages are used in practice can be investigated based on the mined corpus, in order to distinguish the more frequently used and important parts of a language from unimportant parts and thus guide language and tool development. Analyzing the different modeling styles in the corpus allows for identifying best practices and guidelines to help model designers. Furthermore, best practices and tools proposed by academic research or by industry can be validated in a more realistic manner. Currently, there are often case studies, including only a small and homogeneous set of models, or artificial examples used when evaluating new tools and methods, with consequences for an evaluation's validity. Instead using a large set of real-world models as retrieved by mining software repositories allows for increasing validity. In this respect, the prior work on the creation of the *Lindholmen dataset* with UML models mined from software repositories on `GitHub.com` was an inspiration for this paper [12,28].

Empirical research is limited by the access to primary sources. Especially in case of modeling languages used for business processes, like BPMN, where the model is usually the product and thus subject to strict nondisclosure restrictions [30], this can pose an insurmountable challenge. Previous empirical studies on business process modeling and BPMN were using methods like experiments, surveys or case studies, each implying limitations to their generalizability [23]. In an experiment, a certain aspect, e.g., a modeling practice, is typically researched by differentiating two groups based on the aspect. The need to provide a large population and to strictly control the experiment's environment for all other variables, however, restricts the applicability of this method to narrow research problems. Surveys allow for more general problems, but are subject to bias, introduced, e.g., by the selection of survey participants or by inaccurate responses from the participants. Case studies are frequently used and in particular allow for insights into real-world practices and constraints. Compared to experiments, there is though no control of environment and influencing variables. Furthermore, reproducibility and comparability is usually not given. Due to the often homogeneous origin of business process models included in case studies, their findings also need to be validated by other research to increase generalizability [20,23]. Most of the empirical research focused on conceptual process models and omitted implemented and executable process models [23]. This also applies to community efforts to assemble collections of business process models like the *BPM Academic Initiative* [20], which mostly covers educational process models. Mining software repositories for business process models, as introduced in the following can be seen as another empirical approach, complementing established methods.

In this paper, we present our efforts to create a corpus of BPMN process models by mining software repositories hosted on `GitHub.com`. Due to the sheer amount of repositories and the bottleneck caused by the *GitHub API*'s rate limit, we limited our search to a random subset of 6,163,217 repositories, or 10% of all repositories on `GitHub.com` in November 2018. As a result, we were able to identify 1,251 repositories with at least one potential BPMN artifact

and overall 21,306 potential artifacts. We thereby discovered a wide range of file formats, indicating the various uses of BPMN, e.g., modeling conceptual process models or implementing executable processes. Further narrowing our search to business process models in the BPMN 2.0 XML serialization format and removing duplicates, we eventually gained a corpus of 8,904 distinct BPMN process models, conjoined with corresponding repository metadata. Based on this data, we studied descriptive questions on the usage of BPMN on `GitHub.com`. We also ran the analysis tool *BPMNspector* [10] for all the collected business process models to exemplify the use of the corpus for tool validation. Notably, the tool reported at least one violation of BPMN's syntax and semantic rules for 7,365 business process models or 83% of our corpus, which indicates the need for linting tools like *BPMNspector*. In summary, our contributions are:

- To the authors' knowledge, we provide the first corpus of BPMN, comprising 8,904 unique business process models in the BPMN 2.0 serialization format, which is retrieved by mining software repositories on `GitHub.com`.
- In doing so, we confirm the feasibility of the mining software repositories approach for BPMN, which can therefore also be used by others to complement their empirical research on business process models.
- We confirm results on the frequency of violations of BPMN's syntax and semantic rules, which have already been reported for a case study in [10], and thus demonstrate the usefulness of linting tools like *BPMNspector*.

The rest of the paper is structured as follows: We first provide a brief overview of BPMN and available analysis tools in Sect. 2. Our methodology, i.e., the systematic retrieval and analysis of BPMN process models on `GitHub.com` is described in Sect. 3. In Sect. 4, we present our findings with respect to the usage of BPMN on `GitHub.com` and the effectiveness of the analysis tool *BPMNspector*. Section 5 contains a discussion of related work. We conclude the paper in Sect. 6.

2 BPMN and Static Process Analysis

In the following, we shortly sketch the use of the BPMN process modeling language for business processes and provide a brief overview of the several static analysis tools, which are available for helping process designers in identifying modeling flaws and errors prior to process deployment and execution.

The *Business Process Model and Notation (BPMN)* [4] defines a industrial standard for the IT support of business processes and business process management. The language in its current version 2.0 provides a notation for defining the central artifact of a business process lifecycle [8], i.e., the process model, supporting process modeling on a conceptual level as conduced by domain experts as well as the fully-automated deployment and execution of implemented business process models in process engines like *Activiti*[1], *Camunda*[2], or *jBPM*[3]. To this

[1] https://www.activiti.org.
[2] https://camunda.com.
[3] https://www.jbpm.org.


```
<definitions ...
    xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL">
  <process id="process">
    <startEvent id="start">
      <outgoing>flow1</outgoing>
    </startEvent>
    <sequenceFlow id="flow1" sourceRef="start"
        targetRef="task" />
    <userTask id="task" name="theTask">
      <incoming>flow1</incoming>
      <outgoing>flow2</outgoing>
    </userTask>
    <endEvent id="end">
      <incoming>flow2</incoming>
    </endEvent>
    <sequenceFlow id="flow2" sourceRef="task"
        targetRef="end" />
  </process>
</definitions>
```

Fig. 1. Graphical modeling notation and process interchange format of BPMN.

end, the BPMN standard not only includes a graphical modeling notation, but also a machine-processable serialization format for process model interchange and the natural language definition of an execution semantics. In Fig. 1, a simple sample BPMN process model, consisting of a single human task, is shown in its graphical modeling notation as well as in the XML-based interchange format.

Process modeling with BPMN is known to be error-prone, in particular when it comes to executable business processes [9,10,22]. This is due to deliberate decisions on the language's design, e.g., unstructured vs structured process modeling and implied control flow errors, and to the complexity of applications and underlying technologies [29]. To illustrate the latter, consider an executable process package deployed on a process engine like *Camunda*, which not only includes the BPMN process model, but also XML schema files, Groovy or JavaScript code snippets, Java classes, and various configuration scripts. Accordingly, there exists a large body of work on process analysis tools, both from industry and academic research, to help process designers to detect modeling errors as early as possible.

Proposed tools span the whole spectrum of static analysis, i.e., automated rule-based inspection of process models for finding modeling errors or flaws. Linting tools, like *bpmnlint*[4] or *BPMNspector* [10] can be used to check business process models for suspicious and non-portable modeling styles, mostly on the syntactical level. More elaborate tools allow for checking conformance to best practices, e.g., *Signavio*[5], or identifying control and data flow anomalies, e.g., deadlocks [31], processing of undefined data [29], and support even more specific analysis problems like data leak detection [13,16]. Eventually, full-fledged model checking and verification tools can prove the compliance of a process model to certain desirable properties, e.g., proper termination known as soundness [9], by mapping the process model to a formalism like Petri nets [7,14]. Recapitulating

[4] https://github.com/bpmn-io/bpmnlint.
[5] https://www.signavio.com.

the tool evaluations in the literature, we observe the frequent use of case studies, where the most thorough evaluation in [9] comprises 735 process models.

3 Mining Software Repositories for BPMN

In this section, we introduce the approach of mining software repositories and discuss the several ways for retrieving software artifacts and metadata from software repositories hosted on `GitHub.com`. Based upon this, we present our implemented mining process for creating a corpus of BPMN processes models.

Systematically mining software repositories can be seen as a traditional data mining task, including steps for defining a research objective, selecting and extracting data, data preparation and cleansing, data analysis and eventually interpreting the analysis results. The most important data sources for repository mining are public software forges, where in particular `GitHub.com` plays a prominent role due to the sheer number of its hosted repositories. There are several ways for extracting data from software repositories on `GitHub.com`:

GitHub API. `GitHub.com` provides the public REST-based *GitHub API v3*[6], which allows for extracting repository metadata, e.g., specific commits, number of repository contributors or main programming language, as well as information on the repository structure and the repository contents. The API can be accessed without or with authentication, supporting up to 60 or 5.000 queries per hour, respectively. Repository mining using the API is therefore often implemented using authenticated access with a smaller or larger set of different user credentials [11,12]. Note that there also exists the *GitHub API v4*[7], which allows for queries based on the GraphQL language. There apply similar rate limits.

GHTorrent. The *GHTorrent*[8] project provides an alternative way for extracting repository data from `GitHub.com`, thereby avoiding any rate limits. It basically consists of two databases, one for mirroring the stream of events (push events, pull requests, etc.) for software repositories on `GitHub.com` and one for repository metadata, populated by interlinking and analyzing the events' contents [11]. The resulting databases can be accessed via web services or by using the provided dumps to set up local database instances. However, as only repository metadata and events are included, neither the repository structure nor the repository contents can be directly accessed. As a project similar in nature, *GHArchive*[9] provides a database with the repositories raw event data only.

Google BigQuery. In recent years, several datasets containing data on `GitHub.com` software repositories are provided via Google's *BigQuery*[10] service,

[6] https://developer.github.com/v3.
[7] https://developer.github.com/v4.
[8] http://ghtorrent.org.
[9] http://gharchive.org.
[10] https://cloud.google.com/bigquery.

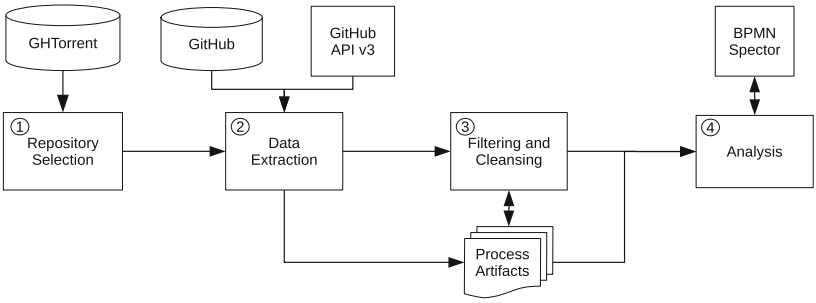

Fig. 2. Schematic illustration of the mining process.

including *GHTorrent* and *GHArchive*. There is as well a dedicated dataset[11] for `GitHub.com` including repository contents for a significant fraction of its repositories, though not all. Using the BigQuery service allows for querying massive datasets like the above-mentioned with SQL queries in a scalable manner. However, as in case of *GitHub API v3*, rate limits apply. Notably, 1TB per month can be accessed freely, which is instantly consumed when querying on tables with repository contents.

Web Scraping and Git. Since `GitHub.com` implements a web interface for managing and accessing hosted software repositories, conventional web scraping can also be used for extracting repository data out of the websites' HTML code. Eventually, knowing the URL of a software repository allows for cloning the repository using the standard git tooling. However, as cloning a repository is time-consuming due to downloading its complete history and all its contents, this may not scale in the presence of thousands or even millions of repositories.

Our approach of mining software repositories for BPMN 2.0 process models uses a combination of the methods discussed above, inspired by previous work on the creation of the *Lindholmen dataset* with UML models from `GitHub.com` [12, 28]. The approach consists of four steps, which we conducted in the beginning of 2019, see also Fig. 2: (1) Get a list of repositories hosted on `GitHub.com` and select a proper subset thereof, (2) Find and extract potential BPMN process model artifacts as well as associated metadata, (3) Examine the artifacts to identify BPMN 2.0 process models and clean up the resulting data, (4) Analyze the resulting set of process models for answering our research questions:

Repository Selection. In the first step, we used the *GHTorrent* database to get a list of all software repositories on `GitHub.com`. To this end, a local copy of the MySQL database was setup downloading the most recent database dump (*mysql-2018-11-01*, 85GB). Querying table *projects* allowed for randomly

[11] https://cloud.google.com/bigquery/public-data (table *github_repos.contents*).

selecting a set of 6,163,217 repositories, which comprises 10% of all non-forked and non-deleted repositories on `GitHub.com` in November 2018.

Data Extraction. All 6,163,217 repositories have been examined for BPMN process model artifacts, using three steps for each repository. Similar to [12], the default branch of the repository (`master` in most cases, though not always) and its latest commit is identified first, using up to three queries to the *GitHub API*. Afterwards, the repository structure is accessed for this commit, in order to generate the list of files for the repository, again querying the *GitHub API*. We reused and adapted the Python scripts[12] from [12,28] for implementing this step. Note that alone this step would require more than 100 days for our repository subset using the credentials of a single user due to the rate limit imposed by the *GitHub API*. In order to increase throughput, we therefore conducted data extraction using several user credentials, which were donated, in parallel. However, this step was still the bottleneck of our approach and lasted 31 days.

Based on the resulting lists, we then scanned for potential BPMN process model artifacts. Having tried several heuristics, we opted for simply including all files with the term `"bpmn"` in their name or file extension. As a result, we found 1,251 repositories with at least one potential artifact and overall 21,306 potential artifacts. Each of the identified repositories was then cloned locally with the standard git tooling. Metadata was extracted from the downloaded repositories using *Code Maat*[13]. Information about the repository, its metadata and the identified artifacts were stored in a relational database for later analysis.

Filtering and Cleansing. We expect to find a wide range of file formats, including images with graphical process models and source code of various programming languages as included in process packages deployed on process engines. Nonetheless, BPMN defines a standard XML-based interchange format, and most tools support to import and export using this format. For our corpus of BPMN process models, we focused on this format and filtered for XML files containing a schema reference matching the BPMN 2.0 serialization standard [4], i.e., `http://www.omg.org/spec/BPMN/20100524/MODEL`. Furthermore, we looked for duplicates using the tooling *Duplicate Files Finder*[14] and only kept distinct files in the resulting corpus of 8,904 BPMN process models.

Analysis. In the final step, we analyzed the retrieved metadata and the identified BPMN process models. The former analysis was mainly implemented by querying a relational database. For the latter analysis, we processed the models and ran the tool *BPMNspector*[15] [10] for them. The tool's reports were fed back into the database and afterwards analyzed and aggregated using SQL queries.

[12] https://github.com/LibreSoftTeam/2016-uml-miner.
[13] https://github.com/adamtornhill/code-maat.
[14] http://doubles.sourceforge.net.
[15] https://github.com/uniba-dsg/BPMNspector.

4 Results and Discussion

The results of our mining process are presented and discussed in this section. In general, the generated corpus of BPMN process models allows for addressing a multitude of research questions and for validating various hypotheses. Due to space constraints, we here focus on the following research questions:

Research Question 1: *Are there projects which use BPMN on* `GitHub.com`*?*
This first questions is used to study the feasibility of the mining repositories approach. Obviously, if there are no BPMN models on `GitHub.com`, the approach can not be used for empirical studies on BPMN. An answer to this question also helps in understanding BPMN's state of practice on `GitHub.com`.

Research Question 2: *How diverse is the corpus of BPMN process models?*
As mentioned in Sect. 1, the generalizability of an empirical study is influenced by the availability of heterogeneous and comprehensive data. We here do not address this question directly. Instead, we analyze indicators as the number of original models, the geographical origin of repositories, the models' age, size, and frequency of changes. Deeper analysis is prospect to future work.

Research Question 3: *How common are violations of BPMN's syntax and semantic rules as identified by BPMNspector?*
With this question, we want to understand, if there is the need for tools like *BPMNspector* with respect to the compliance of process models to the BPMN standard. An answer to this question could guide the development of process modeling tools, e.g., through the integration of respective linting tools, or help in classifying certain rules of the BPMN language as obsolete.

The corpus of BPMN process models and more information, including the list of identified repositories and their metadata, is available online [15][16]. The scripts used to implement the mining process can be obtained from the same source.

4.1 Usage of BPMN on `GitHub.com`

We have found 21,306 potential BPMN process model artifacts, included in 1,251 repositories on `GitHub.com`, which represents a 0.02% share of the overall 6,163,217 analyzed software repositories. Filtering for file formats, we identified serialized BPMN 2.0 process models to be the largest fraction among all the artifacts, counting for more than two thirds as shown in the following table:

File format	XML (BPMN 2.0)	XML (other)	Images (*.png, *.jpg, etc.)	Other (*.jar, *js, etc.)
Number of artifacts	16,907 (79.3%)	384 (1.8%)	1,635 (7.7%)	2,380 (11.2%)

When just considering the serialized BPMN 2.0 process models, the identified artifacts where distributed over 928 software repositories.

[16] https://github.com/ViktorStefanko/BPMN_Crawler.

While we apparently used a very simple heuristic for identifying BPMN process models and therefore may have missed many BPMN models hosted on `GitHub.com`, we nevertheless found a substantial number of artifacts and also of serialized BPMN 2.0 files. The resulting number of BPMN process models clearly exceeds the numbers used in case studies (compare with Sect. 2), but is smaller than the numbers reported for UML models (21,316 in [12] and 93,596 in [28]). The share of 0.02% of repositories with at least one potential BPMN process model artifact is also smaller than the share of 2.8% reported for UML in [12]. The results can be well explained by UML being a family of general-purpose modeling languages, while BPMN is a domain-specific modeling language. Note that UML is also older than BPMN. The reports on UML also present a larger fraction of images among the identified UML models (51.7% in [12] and 61.8% in [28]), which may be reasoned by their more permissive heuristic to consider files with terms like `"diagram"` or `"design"` in their name as UML models.

Answer to Research Question 1: There have been 1,251 repositories with at least one potential BPMN process model artifact, containing 21,306 potential artifacts and 16,907 serialized BPMN 2.0 models.

4.2 Properties of Identified BPMN Artifacts

The next question concerns the diversity of identified BPMN process models. A first hint for an answer is given by the different types of files. As already noted above, found file formats range from XML, over image formats, to source code of programming languages. Assuming that images are used for conceptual models and source code may indicate executable process models, the formats reflect the different uses of BPMN along the business process lifecycle.

Age. We also looked at the age of the identified potential BPMN process model artifacts, i.e., the time passed since their last modification in a repository. Unsurprisingly, most artifacts are recent. More than each third artifact was modified in the last year at the time of conducting the study in the beginning of 2019:

Age in years	< 1	1	2	3	4	> 4	n.a.
Number of artifacts	7,656 (36.0%)	5,154 (24.2%)	3,291 (15.4%)	2,344 (11.0%)	712 (3.3%)	2,079 (9.8%)	70 (0.3%)

We though sporadically found artifacts older than 8 years, which thus do not reference the BPMN 2.0 standard. The results can be well explained by the exponential growth of the number of software repositories on `GitHub.com`.

Updates. The number of updates, i.e., commits implying changes on a given artifact, excluding artifact creation, was also analyzed. We found that 16,285 or over two third of the potential BPMN process model artifacts are never updated. For the remaining artifacts, the number of updates is low, such that only 7.6% of the artifacts are updated more than once, as shown in the following table:

Number of updates	0	1	2	3	> 3	n.a.
Number of artifacts	16,285 (76.4%)	3,378 (15.9%)	620 (2.9%)	572 (2.7%)	427 (2.0%)	24 (0.1%)

BPMN process models thus seem to be rather static contents of software repositories on `GitHub.com`. These results are in line with the findings for UML in [12], where they found that only 26% of the UML models are updated at least once.

Geographical Location. Information on the geographical distribution of potential BPMN process model artifacts was investigated using location information for contributors to a software repository, if available. Unfortunately, we were only able to retrieve the locations of 627 contributors for 395 repositories, or 31.6% of all analyzed 1,251 repositories. Furthermore, a contributor of a repository may not necessary contribute to a process model. However, our findings, as shown below, at least indicate that the artifacts originate from several regions:

Location	China	Germany	USA	Switzerland	France	Other
Number of contributors	92 (14.7%)	90 (14.4%)	90 (14.4%)	24 (3.8%)	22 (3.5%)	309 (49.2%)

Duplicates. Furthermore, we analyzed whether the identified potential BPMN process model artifacts are distinct or if there are any duplicates. We found surprisingly many duplicates, almost one half of all artifacts are duplicates of other artifacts, either in the same or in another repository, lowering the number of distinct artifacts to 10,707 or 50.3%. In the following table, we show how often duplicates occur among the identified BPMN process model artifacts:

Number of occurences	1	2	3	4	5	6	7	8	> 8
Number of artifacts	8,030	852	571	193	110	219	272	342	118

As can be seen, all duplicates can be traced back to 2,677 unique artifacts or 12.6% of all artifacts. In the majority of cases, there are no more than 5 duplicates of a unique artifact, though we also found artifacts with up to 89 duplicates. Just considering serialized BPMN 2.0 process models, we made the same observation, with 8,904 distinct process models and 8,003 duplicates thereof. Why there are so many duplicates is an open question. Possible reasons may be found in the reuse of process models, which are part of platforms like the *Camunda* process engine, in several software repositories, or repositories that are manually derived from other software repositories avoiding `GitHub.com`'s forking mechanism [12].

Size. Analyzing process model size was conducted for the 8,904 distinct process models in the BPMN 2.0 serialization format. The XML-based format includes the XML node `<process>`, which defines a process' logical structure [4]. To get a simple measure for the size of a model, we thus simply counted the number of children elements for the `<process>` node. As can be seen in the following table, the corpus of BPMN 2.0 process models includes a range of different model sizes:

Number of nodes	$1-10$	$11-20$	$21-50$	$51-100$	>100	n.a.
Number of artifacts	1,881 (21.1%)	2,714 (30.5%)	2,346 (26.3%)	978 (11.0%)	878 (9.9%)	107 (1.2%)

While half of the process models are small and contain no more than 20 XML element nodes, we find a substantial share of medium-sized models. There are also larger models in the corpus. Notably, 57 process models contain more than 1,000 nodes. The largest model even contains 4,096 nodes. In [20], they report similar results for BPMN process models collected by the *BPM Academic Initiative*. The average number of nodes there is 16 and the largest model contains 156 nodes. Note that the numbers can though not be directly compared, since we count element nodes in the XML serialization format of a process, which is larger than the sum of its activities and gateways reported in [20].

Answer to Research Question 2: While the majority of identified artifacts are static and recent contents of a software repository, we also found artifacts which are older than 8 years, updated more than once and come from different geographical regions. At least half of the potential BPMN process model artifacts and half of the serialized process models are distinct, yielding a corpus of 8,904 unique BPMN 2.0 process models. The corpus contains models of various sizes.

4.3 *BPMNspector* Analysis Results

The third research question concerns the frequency of violations against BPMN's syntax and semantic rules, as reported when applying the linter *BPMNspector* to the 8,904 BPMN 2.0 process models. *BPMNspector* has been designed to check models for compliance with the BPMN standard [4]. Checks include BPMN's schema validation, referential integrity, and more sophisticated constraints [10]. Violations against the rules imposed by the standard, on the one hand, impede the portability of process models between different tools and vendors and, on the other hand, can cause unexpected behavior when process models are executed [21].

Results of the analysis revealed issues for almost all the process models. While we were not able to run the analysis for 57 process models, we identified only 1,471 process models or 16.5% of all 8,904 models to be compliant with respect to the standard. All other 7,376 models, a share of 82,8%, were analyzed to have at least one violation of rules of the BPMN standard. Overall, *BPMNspector* reported 150,168 rule violations for our corpus of BPMN 2.0 process models. We also took a closer look into the reported issues and found a large fraction to be violations of rules Ext.023, Ext.101 and Ext.107, as can be seen in the following tables, showing the Top-5 rule violations with respect to the number of models with at least one violation and the absolute number of violations:

Rule	Ext.101	Ext.023	Ext.107	Ext.150	Ext.151
Models with violation	5,299 (59.5%)	5,206 (58.5%)	5,112 (57.4%)	1,846 (20.7%)	1,796 (20.2%)

Rule	Ext.023	XsdCheck	Ext.107	Ext.092	Ext.101
Absolute Number	58,015 (38.6%)	43,516 (29.0%)	7,398 (4.9%)	6,788 (4.5%)	6,699 (4.5%)

Rules Ext.023, Ext.101, Ext.107 refer to the non-compliant definition of sequence flows in the BPMN 2.0 serialization format[17]. According to the standard, a sequence flow must be redundantly defined using a node `<sequenceFlow>` and nodes `<incoming>` and `<outgoing>` for the flow's source and target nodes, respectively (also compare with Fig. 1). If one is missing, *BPMNspector* reports an issue. The other shown rules denote violations of BPMN's XML schema (XsdCheck), missing or ambiguous sources of data associations (Ext.092), or missing incoming sequence flow (Ext.150) and missing outgoing sequence flow (Ext.151). We additionally repaired violations of rules Ext.023, Ext.101, Ext.107 in the process models using a tool[18] provided alongside with *BPMNspector*. As a result, the number of models with at least one rule violation shrank to 4,106, still constituting a share of 46.1%, and the absolute number of violations to 64,652.

The authors of *BPMNspector* found similar results when evaluating their tool using a case study of 66 BPMN process models. Though, they reported only a share of 42 models or 63.6% to be non-compliant to the standard's rules. They also identified the largest fraction of violations referring to the wrong use of sequence flows. The high frequency of rule violations can be reasoned by missing tool support, as discussed in [21]. However, as most process engines tolerate the violations reported by *BPMNspector*, the results may also indicate that at least some rules of the BPMN standard are obsolete [10]. In summary, our corpus of BPMN process models confirmed the results of the case study in [10] and therefore provided further empirical evidence for the usefulness of *BPMNspector*.

Answer to Research Question 3: Violations of BPMN's syntax and semantic rules as reported by *BPMNspector* are frequent in the corpus of BPMN process models, affecting 82,8% of the models.

5 Related Work

Most related to our work is the creation and research on the *Lindholmen dataset*[19], which was also the inspiration for our approach. The creators of the dataset describe the used mining software repositories approach [12], introduce the dataset [28], and report on insights gained about the use of UML on `GitHub.com` by analyzing the dataset, e.g., in [5,18]. Their main research question was though on the usage of UML in conventional software development, while we were mainly interested in using our corpus to validate analysis tools for

[17] http://bpmnspector.org/ConstraintList_EXT.html.
[18] https://github.com/matthiasgeiger/BPMNspector-fixSeqFlow.
[19] http://oss.models-db.com/.

BPMN process modeling. The *Lindholmen dataset* is considerably larger than our corpus, currently counting 93,596 UML models [28]. Note again, that UML is a family of general-purpose modeling languages while BPMN is one domain-specific modeling language. They also put more effort into the identification of graphical models, using machine learning in order to identify images which contain UML. Due to our research objective, we were more interested in BPMN's XML-based serialization format. The authors are not aware of any other work, which systematically mines software repositories to create a corpus of BPMN business process models. There though exists approaches in BPM research, which mine software repositories for other purposes. The authors in [3] argue to use repository metadata to mine software development processes. According to their approach, metadata can be for example used to create GANTT charts based on the contributors activities [2] or to classify repository contributors into roles, e.g., developers or testers [1].

As mentioned in Sect. 1, most empirical research on BPMN is based on experiments, surveys or case studies. Notable tool evaluations, going beyond the usual number of included process models, are [9] with 735 customized UML process models and [22] with 585 BPMN models. There have also been several community efforts to create open model collections [17]. For business process modeling, the *BPM Academic Initiative* provides a platform to create and share process models for academic teaching. The authors report on 1,903 different process models in [20], including BPMN, created by 4,500 users and spanning different model size and complexity. The recent number of models is 29,285, but data collection has stopped and the focus is on conceptual business process models [17]. Also note the need for executable process models and their difference to conceptual models [23]. A similar platform was presented by the name *RePROSitory* [6], currently including 174 business process models. Another initiative is the *BenchFlow* project [30], where process models were collected from industrial partners and used for process engine benchmarking. The authors claim to have collected 8,363 models, with a share of 64% of BPMN [30]. Unfortunately, the collection is not public.

Another line of empirical research focuses on the quality of business process models and modeling practices [24]. In [25], process metrics like modularity or complexity are used for predicting modeling errors based on the analysis of a collection of 2000 conceptual models. Experiments where used in [26,27] to investigate on modeling styles in order to better understand and support process designers. More recent work investigated on best practices for BPMN modeling by inspecting practices in industrial process models [22]. Analyzing similarities and differences of these analyses and our corpus is a prospect of future work.

6 Conclusion

In this paper, we describe our efforts to systematically extract a corpus of BPMN business process models from software repositories hosted on `GitHub.com`. Mining 10% of all repositories yielded 21,306 potential BPMN process model arti-

facts, originating from 1,251 repositories, which after further filtering and cleansing constituted a corpus of 8,904 distinct serialized BPMN 2.0 process models. The corpus can be used to answer various empirical research questions on the use of business process models. We here demonstrate, how to complement an existing case study for the linting tool *BPMNspector* with an evaluation on a much larger scale. Doing so, we can confirm results on the frequency of violations of the BPMN standard, thus showing the need for analysis tools like *BPMNspector*.

Threats to Validity. There are a number of threats that affect the validity of our approach. For a general discussion on the threats of mining software repositories, we refer the reader to [19]. Process modeling in software repositories may not resemble industrial practice and our results may thus not generalize beyond open software development and academia [19,23]. This is a common threat to external validity, which we also find for other studies, e.g. [20]. Analyzing the transferability of empirical results about software repositories and academia to an industrial context is an open research question. We therefore advocate the complimentary use of our approach with other empirical research methods. Furthermore, we just mined `GitHub.com` and did not consider other software forges. Since `GitHub.com` counts the largest number of hosted repositories, we though believe that our results apply for most open software development. Due to the heuristics used for the identification of BPMN models, we also have missed process models and thus may underestimate certain effects [12], e.g., model duplication or the frequency of graphical process models. We therefore only provide a descriptive analysis of the use of BPMN on `GitHub.com` and do not use our corpus for inferential statistics and prediction. Finally, due to `GitHub.com` being a dynamic environment, repositories may change or be removed over time.

References

1. Agrawal, K., Aschauer, M., Thonhofer, T., Bala, S., Rogge-Solti, A., Tomsich, N.: Resource classification from version control system logs. In: EDOC Workshops 2016, pp. 1–10. IEEE (2016)
2. Bala, S., Cabanillas, C., Mendling, J., Rogge-Solti, A., Polleres, A.: Mining project-oriented business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 425–440. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_28
3. Bala, S., Mendling, J.: Monitoring the software development process with process mining. In: Shishkov, B. (ed.) BMSD 2018. LNBIP, vol. 319, pp. 432–442. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94214-8_34
4. Business Process Model and Notation (BPMN), Version 2.0. Object Management Group (OMG) Standard (2011). https://www.omg.org/spec/BPMN/2.0/PDF
5. Chaudron, M.R.V., Fernandes-Saez, A., Hebig, R., Ho-Quang, T., Jolak, R.: Diversity in UML modeling explained: observations, classifications and theorizations. In: Tjoa, A.M., Bellatreche, L., Biffl, S., van Leeuwen, J., Wiedermann, J. (eds.) SOFSEM 2018. LNCS, vol. 10706, pp. 47–66. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73117-9_4

6. Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F.: RePROSitory: a repository platform for sharing business PROcess modelS. In: BPM PhD/Demos 2019, pp. 149–153. CEUR (2019)
7. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Techn. **50**(12), 1281–1294 (2008)
8. Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management, 2 edn. Springer, Heidelberg (2018)
9. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_19
10. Geiger, M., Neugebauer, P., Vorndran, A.: Automatic standard compliance assessment of BPMN 2.0 process models. In: ZEUS 2017, pp. 4–10. CEUR (2017)
11. Gousios, G.: The GHTorent dataset and tool suite. In: MSR 2013, pp. 233–236. IEEE (2013)
12. Hebig, R., Quang, T.H., Chaudron, M., Robles, G., Fernandez, M.A.: The quest for open source projects that use UML: mining GitHub. In: MODELS 2016, pp. 173–183. ACM (2016)
13. Heinze, T.S., Amme, W., Moser, S.: Process restructuring in the presence of message-dependent variables. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 121–132. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19394-1_13
14. Heinze, T.S., Amme, W., Moser, S.: Static analysis and process model transformation for an advanced business process to Petri net mapping. Softw.: Pract. Exp. **48**(1), 161–195 (2018)
15. Heinze, T.S., Stefanko, V., Amme, W.: Mining von BPMN-Prozessartefakten auf GitHub. In: KPS 2019, pp. 111–120 (2019). https://www.hb.dhbw-stuttgart.de/kps2019/kps2019_Tagungsband.pdf
16. Heinze, T.S., Türker, J.: Certified information flow analysis of service implementations. In: SOCA 2018, pp. 177–184. IEEE (2018)
17. Ho-Quang, T., Chaudron, M.R.V., Robles, G., Herwanto, G.B.: Towards an infrastructure for empirical research into software architecture: challenges and directions. In: ECASE@ICSE 2019, pp. 34–41. IEEE (2019)
18. Ho-Quang, T., Hebig, R., Robles, G., Chaudron, M.R.V., Fernandez, M.A.: Practices and perceptions of UML use in open source projects. In: ICSE-SEIP 2017, pp. 203–212. IEEE (2017)
19. Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D.E.: The promises and perils of mining GitHub. In: MSR 2014, pp. 92–101. ACM (2014)
20. Kunze, M., Luebbe, A., Weidlich, M., Weske, M.: Towards understanding process modeling – the case of the BPM academic initiative. In: Dijkman, R., Hofstetter, J., Koehler, J. (eds.) BPMN 2011. LNBIP, vol. 95, pp. 44–58. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25160-3_4
21. Lenhard, J., Ferme, V., Harrer, S., Geiger, M., Pautasso, C.: Lessons learned from evaluating workflow management systems. In: Braubach, L., et al. (eds.) ICSOC 2017. LNCS, vol. 10797, pp. 215–227. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91764-1_17
22. Leopold, H., Mendling, J., Günther, O.: Learning from quality issues of BPMN models from industry. IEEE Softw. **33**(4), 26–33 (2016)

23. Lübke, D., Pautasso, C.: Empirical research in executable process models. Empirical Studies on the Development of Executable Business Processes, pp. 3–12. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17666-2_1
24. Mendling, J.: Empirical studies in process model verification. In: Jensen, K., van der Aalst, W.M.P. (eds.) Transactions on Petri Nets and Other Models of Concurrency II. LNCS, vol. 5460, pp. 208–224. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00899-3_12
25. Mendling, J., Sánchez-González, L., García, F., Rosa, M.L.: Thresholds for error probability measures of business process models. J. Syst. Softw. **85**(5), 1188–1197 (2012)
26. Pinggera, J., et al.: Styles in business process modeling: an exploration and a model. Softw. Syst. Model. **14**(3), 1055–1080 (2013). https://doi.org/10.1007/s10270-013-0349-1
27. Pinggera, J., et al.: Tracing the process of process modeling with modeling phase diagrams. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011. LNBIP, vol. 99, pp. 370–382. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_36
28. Robles, G., Ho-Quang, T., Hebig, R., Chaudron, M., Fernandez, M.A.: An extensive dataset of UML models in GitHub. In: MSR 2017, pp. 519–522. IEEE (2017)
29. Schneid, K., Usener, C.A., Thöne, S., Kuchen, H., Tophinke, C.: Static analysis of BPMN-based process-driven applications. In: SAC 2019, pp. 66–74. ACM (2019)
30. Skouradaki, M., Roller, D., Leymann, F., Ferme, V., Pautasso, C.: On the road to benchmarking BPMN 2.0 workflow engines. In: ICPE 2015, pp. 301–304. ACM (2015)
31. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models through SESE decomposition. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 43–55. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74974-5_4