Automated Summarization of Service Workflows to Facilitate Discovery and Composition

Panagiotis Kotsikoris, Theodore Chaikalis[a], Apostolos Ampatzoglou[b] and Alexander Chatzigeorgiou[c]
Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece

Keywords: BPMN, Business Processes, Service Discovery, Workflow Composition, Cloud Software Development.

Abstract: The last decade marked undeniably the leading role of web services and the establishment of service-oriented architectures. Indeed, it is nowadays hard to find a contemporary software application that does not use at least one third-party web service. The main driver for this paradigm shift, lies in the benefits that decoupled, cloud-based services bring to software development, operation and maintenance as well as at the seamless deployment, integration and scalability features those modern public clouds provide. Furthermore, the widespread adoption of services has led to the consequent demand for a structured and accessible method for automatic service categorization, documentation, and identification, so that all available web services can be easily identified and used from possible clients. In the realm of web services, service compositions known as workflows provide a natural way to automate existing business processes and bridge the gap between technical and non-technical stakeholders. This work proposes an automatic documentation generator for business processes which facilitates service discovery, based on automatic summarization of business processes created through Business Process Model and Notation (BPMN)

1 INTRODUCTION

Rendering software functionality available on the Internet through standardized messaging protocols has become the primary method for developing enterprise software systems. Web services are defined as self-contained, modular and distributed applications that can be deployed and invoked over the network. More specifically, RESTful web services are scalable and light-weight and are commonly used to create APIs for web-based applications (Richardson & Ruby, 2007). In the REST architectural style, both data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs). Services can be composed to form workflows abstracting existing business processes.

Business Process Model and Notation (BPMN) is a widely established industry standard, capable of graphically modelling business flows, whose execution provides value to business stakeholders. Through well-defined and intuitive semantics, BPMN provides powerful capabilities that cover numerous different business scenarios. Each process is divided in multiple atomic and strictly defined steps which occur between start and end nodes. For the visual representation, BPMN uses nodes for modelling actions and edges for connecting actions and thus denoting the direction of the data and control flow. The resulting process graph reflects also the chronological order of execution.

BPMN has been initially released on 2007 and the current official release is version 2.0.2 that has been published in January 2014 (Geiger et al., 2016; OMG, 2007). The officially stated goal of the standard is to provide a method for visualizing in a strict and formal, yet intuitive and approachable way business processes so that they can be developed from both technical and non-technical stakeholders.

[a] https://orcid.org/0000-0003-0501-3671
[b] https://orcid.org/0000-0002-5764-7302
[c] https://orcid.org/0000-0002-5381-8418

Kotsikoris, P., Chaikalis, T., Ampatzoglou, A. and Chatzigeorgiou, A.
Automated Summarization of Service Workflows to Facilitate Discovery and Composition.
DOI: 10.5220/0011036200003176
In *Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2022)*, pages 317-326
ISBN: 978-989-758-568-5; ISSN: 2184-4895

While BPMN offers inherently a visual representation of the modeled business process, it is often convenient to have a brief, textual summary of the underlying workflow. Apart from labeling purposes, such summaries can be highly valuable for identifying existing workflows in registries (through keyword search). Furthermore, for platforms aiming to assist web service developers in reusing similar service compositions, textual summaries can be used to enable service searching, identification and reuse.

In the SmartCLIDE H2020 project[1] we aim at developing a smart cloud-native development environment that will support creators of cloud services in the discovery, creation, composition, testing, and deployment of full-stack data-centered services and applications in the cloud. Among other features, SmartCLIDE facilitates workflow development by recommending nodes, after seeking similar workflows to the one that is being developed. Workflow similarity is assessed based on the similarity of their corresponding textual descriptions. To this end, we have leveraged the power of Natural Language Processing (NLP) techniques to automatically extract a concise textual summary of an existing workflow expressed in BPMN.

The rest of the paper is organized as follows. Section 2 discusses related work on service composition and NLP. In Section 3 we elaborate on the adopted methodological approach and in Section 4 we present details about the implemented tool. Section 5 discussed results from the evaluation of the tool in an industrial context. Finally, we conclude in Section 6.

2 RELATED WORK

As described above, numerous technologies have been combined (web applications, BPMN, NLP etc.), to achieve the greater goal of workflow summarization. Each one of these technologies forms an entire field of research and practice in the science of information technology and has evolved largely over time.

Discovering appropriate services and components on the network (intranet or internet), in order to enhance and extend an application under development, is one of the holy grails in today's software development. Leveraging the power of existing services can help us escape from the hardcoded model used in monolithic application development which leads to dysfunctional and hard to maintain collections of software modules. Messina et al. (2016) describe a dynamic and flexible model to keep track of all the components using a service registry focusing on consistency during the discovery process while having each part isolated from the outside world.

Assuming there is an existing set of available components, the next step is to combine them in a way to achieve the desired functionality. The composition of those parts is a concept where instead of dealing with the infrastructure and the location of the components, we focus on bringing the pieces together to achieve a greater goal. Wu et al. (2015) describe two basic ways to create compositions of services. The first one is a manual process where user decides which components will be combined and in which way, leading to more meaningful outcomes at the cost of significant effort and time, taking into consideration the existence of thousands of available options. The second way is an automated process where a list of components is evaluated using artificial intelligence approaches. Next, heuristic functions decide the best option to use. The latter approach yields less accurate compositions but offers substantially higher performance. Both options have their drawbacks, for example, for a user to evaluate a set of services, he first needs to execute them in a controlled environment, requiring a set of resources and tools. On the other side, the automated process also requires resources regarding the collection and the evaluation of the services in a parallel or even distributed way.

In order to generate an accurate definition of a flow, we first need to deconstruct its individual parts and classify them. In this process Natural Language Processing (NLP) comes in hand. Alan Turing back in 1950 was the first to conceive the idea that determining or not whether a computer is truly intelligent involves the generation of natural language as criterion of intelligence. Since then, multiple efforts were made to create chatbots or other conversational agents that can interact with the content in the same way that a person does. Lo et al. (2017) performed an analysis on various implementations to clarify how far we have come today and the potentials that lie ahead. Most of those implementations are naturally based on the English language. Fahad et. al. (2018) discuss the techniques used today using neural networks as a pillar for multi document summarization, spell checking, speech recognition etc.

[1] https://smartclide.eu/

The generation of natural language, which follows initial text processing, is the process in which an artefact or model is presented in text form as output. McBurney et al. (2016) suggest a five-step procedure in order to create a final text that can describe accurately a block of code and compare the results with the respective human generated descriptions. At first, the content that will be used is selected, which is a very crucial step that defines the quality of the result. After the content filtering comes the structuring, in which the sequence of the content is defined. The third step is the lexicalization, where more suitable words are selected when necessary. After that comes aggregation of similar phrases and sentences in order to remove any duplicate content. The final step is the realization, where the content to be exported is created taking into consideration the respective grammar rules. Since this is a standalone approach working automatically, it is effective to any input given in any language.

For the BPMN 2.0 specification, Geiger et al (2016) performed an analysis of the evolution of the standard, while Falcone et al. (2017) focus on the potential of the model that originates from its appealing graphical presentation and attempt to integrate it with other modelling and simulation systems.

3 BACKGROUND

3.1 BPMN

An example of a simplified business process is depicted in Figure 1 where a subset of the BPMN elements are demonstrated. The diagram models a process that gets triggered by a message which denotes that a working group is active. Next a scheduled activity triggers a repeating set of tasks every Friday at 6 PM: First a system task checks the status of the working group. Then a decision node determines the execution path depending on the result of whether the working group is active or not. It 'active=true', the execution continues to the task that sends a current issue list and then it goes back to waiting mode until the next Friday @ 6 PM. Otherwise the process terminates.

Both system tasks in this process (i.e. checking the working group's status and sending the current issue list) could be easily carried out by two respective Web Services that can be triggered through special configuration during the creation of the tasks.

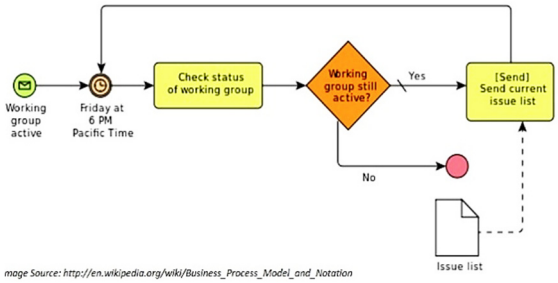

Figure 1: Example BPMN process.

3.2 Text Summarization

Text summarization refers to any automated process during which a program parses a set of sentences and generates a meaningful summary. The main goal of every summarization approach is to produce a precise and concise summary that reflects the basic context of the initial document and does not lack vital information, neither includes unnecessary or irrelevant details (Figure 2). Such approaches have been extensively used in automatic text classification, news article summarization and automatic title production.

As a technical problem, automatic summarization forms a challenging endeavour because unlike humans who can scan through an article and quickly grasp the context and the basic parts of the topic, machine algorithms must overcome several non-trivial problems for capturing the essence of a document.

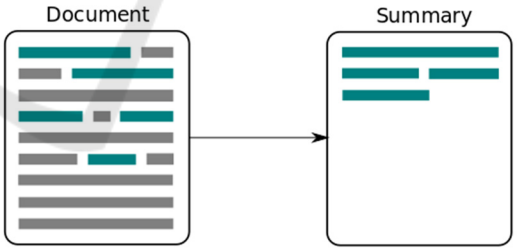

Figure 2: Text Summarization.

Three approaches are mainly used for automatic text summarization (Widyassari et al., 2020).

- Extraction Based summarization, where content only from the original document is being used. An initial evaluation pass determines the parts that should be kept and afterwards these parts are merged for the final summary.
- Abstraction Based summarization, which attempts to mimic the human approach. Initially, Natural Language Understanding techniques are being used for identifying the meaning of the document and

afterwards the final summary is produced by rephrasing versions of the basic parts.
- Aided summarization is a semi-automatic approach which demands intervention by a human who will pick the text parts that will form the final summary, as well as the method that this should be performed. This approach is used in occasions where high accuracy is needed.

The first one, extraction-based approach is the one that initially gained high popularity due to its simplicity and the abundance of available information. However, with the improvements in AI and computing power a huge part of the implementations has been move to the abstraction-based approach (Widyassari et al., 2020), which can lead to more meaningful summaries.

4 TOOL IMPLEMENTATION

The proposed tool forms part of a broader toolset which aims at semi-automatic service discovery and composition by using service descriptions that assist users in selecting the service that suits their needs in the most effective way. To this end, descriptions of available service flows should automatically be extracted by parsing BPMN diagrams. The summative descriptions can be used as short textual service documentation about the purpose and the mechanics of each service composition. This process produces a documented registry containing all services/workflows that can be identified and reused.

The selection process takes place in two steps. Initially users defined a set of keyword-based criteria for a desired service (service composition) and the system responds to this search by returning a list of compositions that are the best possible fit the given criteria. In the second step users read through returned results and select the one that is the most suitable to their demands.

The description of each service composition is obtained by analysing the internal structure, the underlying flow with all possible execution paths and the naming of all BPMN elements (nodes). The documentation is created by parsing the BPMN representation files (XML structure) and yields a textual description output using Natural Language Processing and Natural Language Generation techniques.

4.1 System Architecture

In this chapter a detailed analysis of all system components is presented along with a description of the steps that produce the service documentation.

The tool accepts one or more BPMN files as input, processes them and outputs the produced documentation in the form of text files in English language (Figure 3). It becomes evident that for the tool to operate correctly, the system should process only BPMN processes described in English.

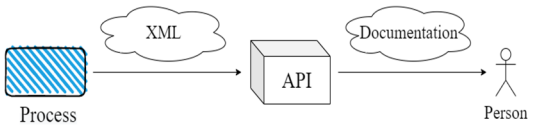

Figure 3: General analysis concept.

We opted for the Web as our deployment platform to provide a multi-channel API that can be invoked both by human users as well as from third-party web services as shown in Figure 4. The proposed tool is available both through a headless API and in the form of a human reachable Website.

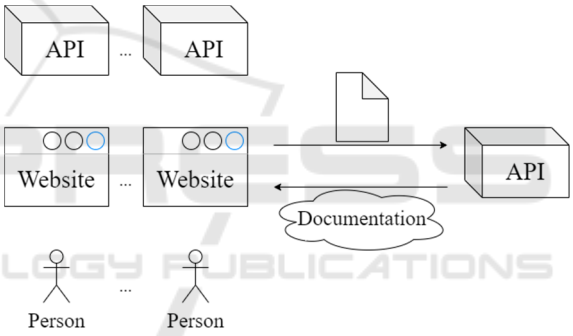

Figure 4: Application is available through both as a headless API and as human reachable Website.

The backend has been developed using the Java Spring Boot framework (*Spring Boot*, 2021) and the public interface is exposed through a REST API.

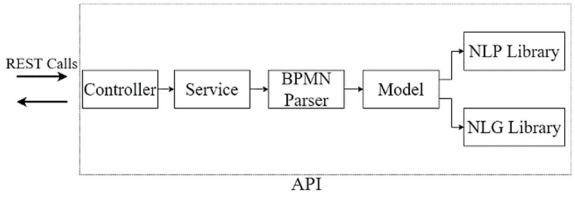

Figure 5: Internal application Architecture.

The internal structure of the backend module is depicted in Figure 5. The Controller module acts as the gateway of the system as it handles external requests, performs an initial input validation, and returns responses to clients.

The Service component is responsible for carrying out the actual business logic for the request. It contains the source code that analyses a business process and produces its documentation. Internally it invokes the BPMN parser, which goes through all XML nodes in the underlying XML file, models each node as a Java Object, invokes the Natural Language Processing Subsystem to semantically analyse the retrieved information and finally generates output with the Natural Language Generator. More details about the core functionality of extracting a textual summary are provided in the next subsection.

4.2 Analysis Engine

In this section a more in-depth analysis about the automatic business process documentation engine is presented. The overall flow which comprises three sequential phases is depicted in Figure 6. Initially the process is being parsed for the determination of all nodes in the workflow. Then each node is visited and documented, starting from the first node of the process, the Start Event. Finally, all results are collected and formatted to formulate the final description.

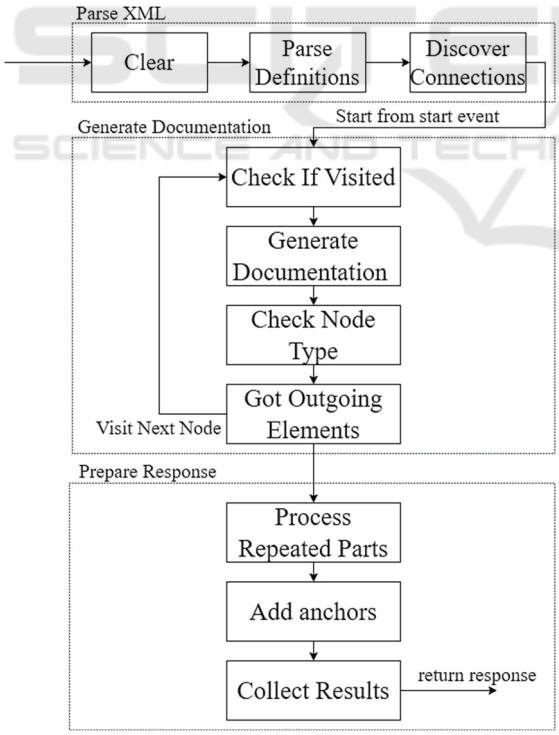

Figure 6: Detailed analysis of processing flow.

Phase A - BPMN Parsing: This phase starts from the cleaning task which removes redundant or unnecessary parts of the process that would introduce noise to the NLP algorithms later. The basic part of this phase is the modelling of BPMN nodes into Java Objects and the identification of connections among them. This task ensures the validity of the given process and facilitates the understanding of the process flow.

Phase B – Documentation Generation: This phase accepts the previously created objects with their connections and creates a description for each one of them. These descriptions are the parts that will comprise the final documentation. The core algorithm contains a recursive procedure which visits all elements in specific order beginning from the Start Event of the process.

Phase C – Response Preparation: After visiting all nodes, all parts are aggregated to give the final result. In the occurrence of back cycles in the flow, special anchors in the form of markers are inserted to avoid redundant text duplication. A second optimization that is applied in this step is the removal of duplicated phrases or sentences. Finally, the complete process documentation is returned.

4.2.1 Documentation Process

The description of a process is created from the concatenation of sentences produced from individual parts. Apart from specific descriptions for each part of the process, a general introductory sentence offering a general overview of the process context, is inserted at the begging of the documentation.

Table 1: Process characteristics.

Adjective	Description
Scheduled	Process that contains a timer event that controls the firing time and date
Automatic	No human interaction involved
Single approval	One step of approval only
2-level approval	Two approval steps
3-level approval	Three approval steps
Multiple-level approval	More than three approval steps
Short (single-task)	Process contains only one task
Short (2-tasks)	Process contains 2 tasks
Short (3-tasks)	Process contains 3 tasks
Flat (no branching)	Process does not contain a decision gateway
Multi branched	Contains one or more decision gateways.
Alternative ending	Contains multiple ending nodes
Repetitive	A cyclic flow exists

For the general description of the process structure, the analysis engine searches for a set of specific characteristics which map to adjectives that are composed to form the desired phrase. Table 1 presents the characteristics along with a short description.

The general description contains also a phrase describing the process type which is deducted from the analysis of the Start Node and the BPMN file itself. Initially the algorithm attempts the retrieval of the description of the Start Node, however if description does not exist or is not usable (no noun), the algorithm utilizes the name of the BPMN file that contains the process. The reason for selecting these 2 elements is because BPMN process developers describe them in a way that differentiates one process from the others.

5 USE CASES AND EVALUATION

This section presents several use cases with results of the application of the proposed approach on synthetic examples. Next the results from the empirical evaluation of the results are presented. To validate the quality of the extracted summaries we resorted to industry professionals having experience in creating/understanding process workflows as part of their job.

5.1 Use Cases

Figure 7 depicts a simple book reservation process with one human task and one system/notification task.

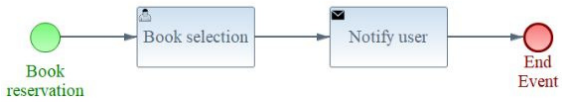

Figure 7: Simple book reservation use case.

The automatic documentation of the process is as follows:

This is a single approval, short (2-task), flat process about book reservation.
A user decides about the book selection. A send task is used to notify user.
Then the process ends.

Bold Highlighted Text in Italics designates the general description of the process from which we can deduct that a user task exists (this enables the characterization of the process as one with "approval step"), that is a flat process (without alternative paths) and that it contains two individual tasks and is about a book reservation. Next, the process steps are described in chronological order described in clear text with zero formatting as it is deduced from the order of nodes on the directed graph. Underlined phrases indicate the description about the process terminating node.

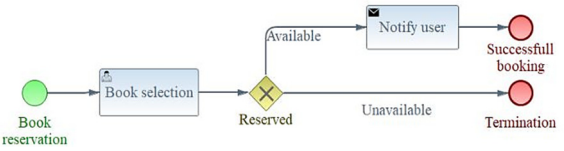

Figure 8: Example Business Process with decision branching.

An example of a business process which contains a decision node, which leads to two branches is depicted in Figure 8. After book selection, an availability check takes place and if the book is available, the process notifies the user, otherwise it immediately terminates.

The produced description follows. Note that the general description here lacks the characterization "flat" due to the branch existence.

This is a single approval, short (2-task) process about book reservation.
A user decides about the book selection. A decision is taken depending if reserved.
If available: A send task is used to notify user. Then the process ends.
If unavailable: Then the process ends.

We proceed with a few more complicated examples to highlight the strengths and weakness of the proposed approach.

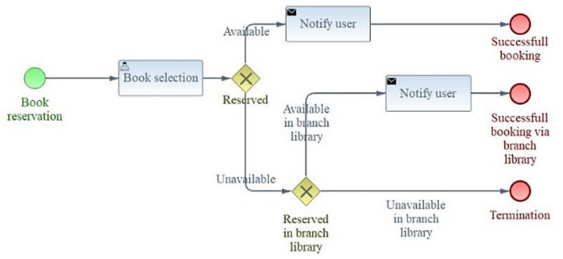

Figure 9: Process example with two decision branches.

Figure 9 presents a business process with two decision gateways, which create three decision branches. The produced description is as follows:

This is a single approval, short (3-task), alternate ending (3-ways) process about book reservation.
A user decides about the book selection. A decision is taken depending if reserved.
If available: A send task is used to notify user. Then the process ends.
If unavailable: A decision is taken depending if reserved in branch library.
If available in branch library: A send task is used to notify user. Then the process ends.
If unavailable in branch library: Then the process ends.

The process in Figure 10 contains an intermediate throw signal event which means that when this node is executed, a special signal is emitted to notify other interested (subscribed) processes.

Figure 10: Process with an intermediate Catch Event.

The automatic description for the process with intermediate signal throw as shown in Figure 10 is as follows:

This is an automatic (not involving any human task), short (2-task), flat process about book return.
A method is called to set book status. The process broadcasts a signal. A manual task is used to update book position.
Then the process ends.

The process in Figure 11 introduces the concept of End Message. This is a signal with a message payload that is emitted at the end of the process to notify any other interested processes (Message Start events or intermediate message-catch events). The same approach is being followed for all signal types (Message, Error, Cancel, Link etc).

The automatically generated description follows.

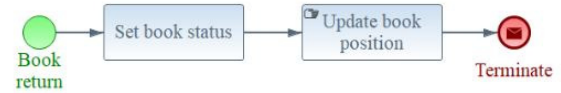

Figure 11: Example Process with End Message Event.

This is an automatic (not involving any human task), short (2-task), flat process about book return.
A method is called to set book status. A manual task is used to update book position. The process concludes messaging a participant.

Next, in Figure 12 a special process with a timer controlled segment is depicted. A timer event is a special type of node on which the execution flow blocks until the timer expression that is declared during node creation is fulfilled. The automatically generated description follows.

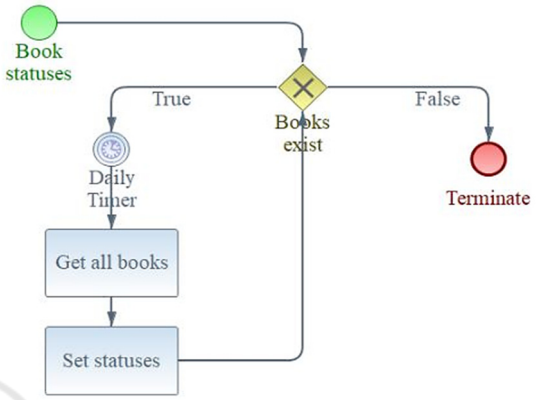

Figure 12: Example process with Timer Event.

This is a scheduled, automatic (not involving any human task), short (2-task), repetitive process about book statuses.
[1] A decision is taken depending if books exist.
If true: A daily timer is used to repeat the following process. A method is called to get books. A method is called to set statuses. The same flow is repeated [1].
If false: Then the process ends.

As it is readily observed, special anchors have been used for the correct description of the flow circle in order to avoid redundant phrase repetitions.

Figure 13 depicts a process with multiple levels of approval and multiple endings. The corresponding automatic documentation follows.

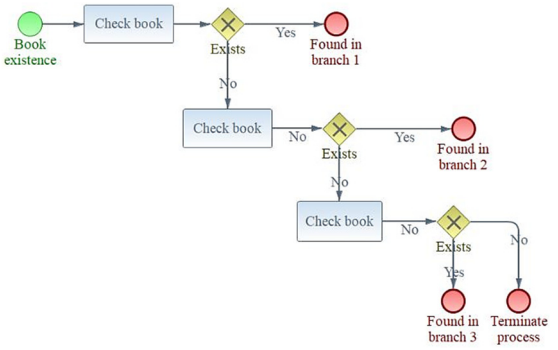

Figure 13: Example process with multiple approval levels and multiple endings.

This is an automatic (not involving any human task), short (3-task), alternate ending (4-ways) process about book existence.

```
A method is called to check book.
A decision is taken depending if exists.
   If yes: Then the process ends.
   If no: A method is called to check book.
A decision is taken depending if exists.
   If yes: Then the process ends.
   If no: A method is called to check book.
A decision is taken depending if exists.
   If yes: Then the process ends.
   If no: Then the process ends.
```

5.2 Evaluation

To evaluate the effectiveness of the proposed methodology in capturing the essence of the actual business process through the extracted summary, an online questionnaire-based study was conducted. The participants were nine industry professionals, all of them highly skilled on BPMN process creation and understanding.

Each participant was presented with 6 processes along with the corresponding automatically generated documentation and was asked to evaluate the extent to which the documentation captured the actual process and is meaningful. The responses were provided in Likert scale (Joshi et al., 2015) as depicted in Figure 14.

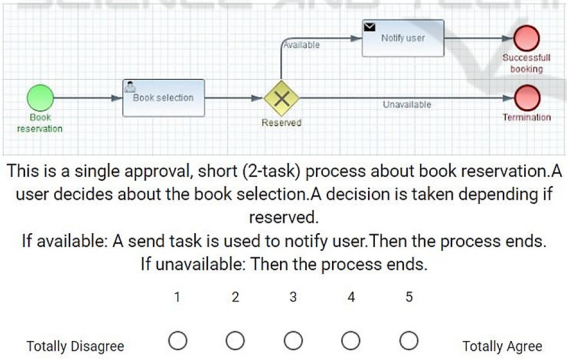

Figure 14: Indicative evaluation question.

The responses indicate a generally positive opinion about the extracted summaries. A pie chart with an overview of the evaluation results is presented in Figure 15. It should be noted that none of the participants voted with a value equal to 1. The majority of evaluations correspond to values 4 and 5 (reaching a total 80%), highlighting the expressiveness of the approach, at least for the used case studies.

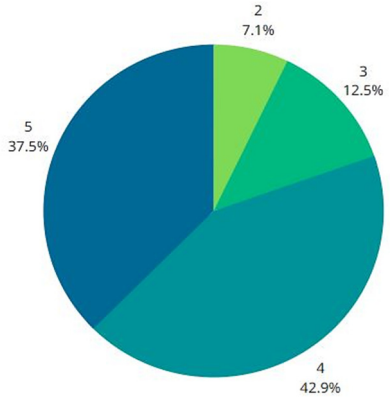

Figure 15: Distribution of evaluation responses.

The next section presents separate results for each case along with the distribution of the evaluations received, depicted as a bar chart. It is reasonable to expect variations in the user acceptance of the summaries, as some processes are more complex than others. The results confirm this hypothesis, since some of the most complex scenarios, such as case 6 (Fig. 21) and case 4 (Fig. 19) received fewer '5' scores.

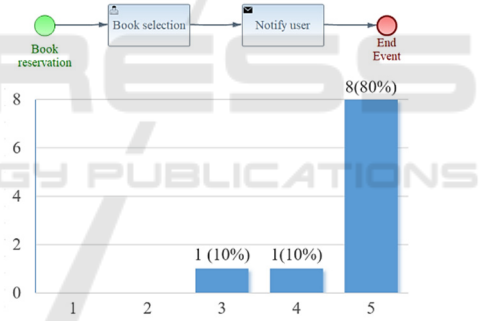

Figure 16: Evaluation case 1.

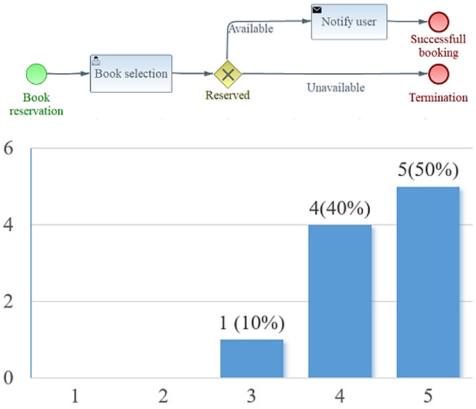

Figure 17: Evaluation case 2.

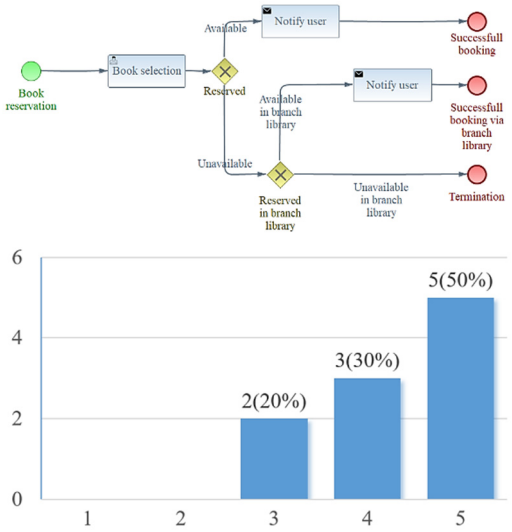

Figure 18: Evaluation case 3.

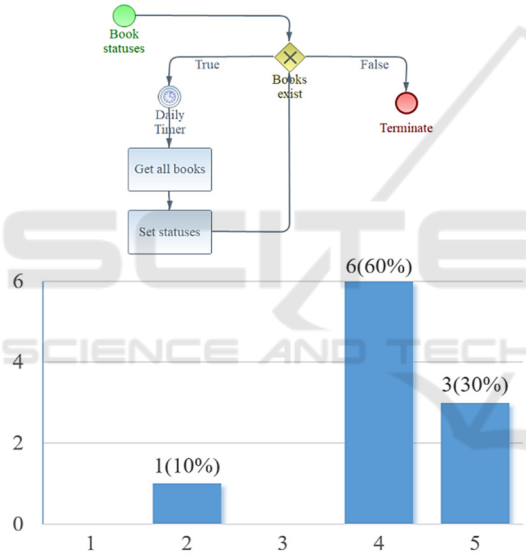

Figure 19: Evaluation case 4.

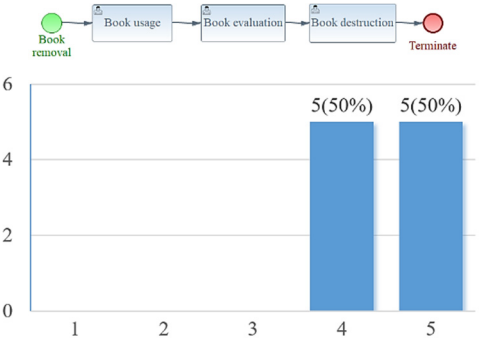

Figure 20: Evaluation case 5.

The evaluation results from this pilot evaluation on the 6 presented cases are positive and demonstrate the potential of the approach. Extracting a meaningful summary becomes more challenging for processes having multiple paths and more nodes. A possible explanation of this phenomenon might be the fact that longer processes produce longer documentation segments which inevitably are less cohesive and more difficult to express in natural language (even if a human attempts to extract a summary for a highly complex workflow the resulting text would seem less 'natural'). Through the free text responses that were allowed, the participants identified syntactic errors that should be avoided.

We remind that the goal of extracting a summary for a given workflow is not only the 'labeling' of a process with text but to enable the searching for similar workflows through similarity of their corresponding description. While this experiment is planned as future work we believe that the extracted summaries capture both key aspects of the process (such as decision nodes and timed events) and all necessary terms that characterize nodes. Thus, we are optimistic that performing similarity checks between the textual summary of BPMN process will allow the efficient identification of 'similar' workflows.

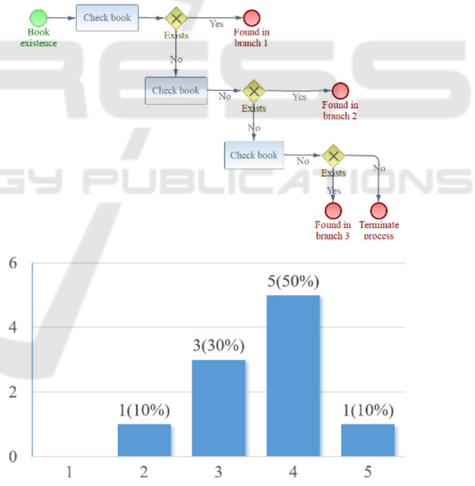

Figure 21: Evaluation case 6.

6 LIMITATIONS AND THREATS TO VALIDITY

The proposed approach and accompanying tool suffers from specific limitations which can be addressed by more advanced NLP techniques. First of all, the tool cannot parse workflows containing descriptions in languages other than English, but if such a need exists, the approach can be adapted accordingly. The tool will also yield inaccurate

summaries in case of missing verbs in BMPN elements or for structures which exhibit substantial complexity (e.g. in the case of tens of paths between the start and end node). While there is no trivial way to address such issues, human intervention to abstract entire blocks of BPMN elements by tagging them with an appropriate high-level description, might be promising in this direction.

The performed study to evaluate the effectiveness of the approach and tool is also subject to validity threats. The number of employed evaluation cases in limited, both in number and in terms of the underlying domain. As a result, any claims about the potential of the approach are subject to external validity threats and the results cannot be generalized to other domains. Furthermore, we acknowledge that using a simple rating to capture the correctness, meaningfulness and attractiveness of the extracted summary might be insufficient to assess the pros and cons of the approach. The relevant construct validity threat can be addressed by more systematic and larger-scale validations in the future. The low number of participants did not allow any systematic statistical analysis (e.g. to investigate inter-rater agreement) of the findings. We plan to advance the statistical conclusion validity during the case study on the ability of using summaries to identify similar processes.

7 CONCLUSIONS

Service-based software systems have become mainstream in various domains as the benefits of using and composing individual services towards reduced development time, better scalability and easier maintainability are well acknowledged and documented. Modeling real business processes as BPMN workflows where individual nodes correspond to invoked services has great potential to lower the entry barriers to system development. However, findings workflows which are similar to the targeted one, so as to reuse previous services is challenging.

To address this problem, as part of the SmartCLIDE H2020 project, we have developed an approach and accompanying tool that automatically extracts summaries from a BPMN process. By providing as input the process file a textual summary is extracting leveraging NLP techniques. A pilot evaluation with 9 industry professionals revealed a positive reception of the generated summaries. As a next step, we plan to evaluate the efficacy of textual summaries as a means for finding similar workflows.

ACKNOWLEDGEMENTS

Work reported in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871177 (project: SmartCLIDE).

REFERENCES

Fahad, S. A., & Yahya, A. E. (2018). Inflectional Review of Deep Learning on Natural Language Processing. *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*. https://doi.org/10.1109/ICSCEE.2018.8538416

Falcone, A., Garro, A., D'Ambrogio, A., & Giglio, A. (2017). Engineering systems by combining BPMN and HLA-based distributed simulation. *2017 IEEE International Systems Engineering Symposium (ISSE)*, 1–6. https://doi.org/10.1109/SysEng.2017.8088302

Geiger, M., Harrer, S., Lenhard, J., & Wirtz, G. (2016). On the Evolution of BPMN 2.0 Support and Implementation. *2016 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, 101–110. https://doi.org/10.1109/SOSE.2016.39

Joshi, A., Kale, S., Chandel, S., & Pal, D. K. (2015). Likert scale: Explored and explained. *British Journal of Applied Science & Technology*, *7*(4), 396.

McBurney, P. W., & McMillan, C. (2016). Automatic Source Code Summarization of Context for Java Methods. *IEEE Transactions on Software Engineering*, *42*(2), 103–119. https://doi.org/10.1109/TSE.2015.2465386

Messina, A., Rizzo, R., Storniolo, P., & Urso, A. (2016, June 26). *A Simplified Database Pattern for the Microservice Architecture*. https://doi.org/10.13140/RG.2.1.3529.3681

OMG. (2007). *BPMN Specification—Business Process Model and Notation*. https://www.bpmn.org/

Richardson, L., & Ruby, S. (2007). *RESTful Web Services* (1st ed.). O'Reilly Media.

Spring Boot. (2021). https://spring.io/projects/spring-boot

Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., Affandy, A., & Setiadi, D. (2020). *Review of automatic text summarization techniques & methods*. https://doi.org/10.1016/j.jksuci.2020.05.006

Wu, Z., Deng, S., & Wu, J. (2015). Chapter 7—Service Composition. In Z. Wu, S. Deng, & J. Wu (Eds.), *Service Computing* (pp. 177–227). Academic Press. https://doi.org/10.1016/B978-0-12-802330-3.00007-2