Journal of King Saud University – Computer and Information Sciences 34 (2022) 10079–10093

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Generating BPMN diagram from textual requirements

Sholiq Sholiq [a,b], Riyanarto Sarno [a,*], Endang Siti Astuti [c]

[a] Department of Informatics, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo-Surabaya 60111, Surabaya, Indonesia
[b] Department of Information Systems, Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo-Surabaya 60111, Surabaya, Indonesia
[c] Department of Business Administration, Brawijaya University, Malang, Indonesia

ARTICLE INFO

Article history:
Received 23 May 2022
Revised 8 September 2022
Accepted 10 October 2022
Available online 20 October 2022

Keywords:
BPMN diagram
Natural language
Textual requirement

ABSTRACT

An interesting challenge in software requirements engineering is converting textual requirements to Business Process Model and Notation (BPMN) diagrams. In this study, the BPMN diagram is used as an intermediate representation before measuring the functional software size from Natural Language (NL) input. The methods currently used for converting NL input to BPMN diagrams are not able to generate complete BPMN diagrams, nor can they handle complex and compound-complex sentences in the NL input.

This study proposes conversion from textual requirements to a BPMN diagram for improving the weaknesses of existing methods. The proposed method has two stages: 1) analyzing the textual requirements using natural language processing and 2) generating the BPMN diagram. The output of the first stage is fact types as the basis for generating the BPMN diagram in the second phase. The BPMN diagram is generated using a set of informal mapping rules that were created in this study.

The proposed method was applied to ten textual requirements of an enterprise application, which involved simple, compound, complex, and compound-complex sentences. The experiments resulted in a suitable BPMN diagram with higher accuracy than obtained by other methods.

1. Introduction

Information about software size is crucial for several interested parties in software development projects. Project managers need this information to plan resources, cost, and personnel requirements and to monitor and control projects during execution. Systems analysts need the software size to determine the scope of the software to be developed. Software testers need this information to estimate software defects or defect density of the software unit (Cosmic, 2019). The method first used to measure software size was Source Line of Code (SLOC) (Nguyen et al., 2007). Later, a measurement was developed based on the software requirements called Functional Size Measurement (FSM) (Ochodek, 2016). FSM is based on functional requirements known as Functional User Requirements (FUR). The FUR is still in an unstructured form at the early stages of software development and is only later finalized by business analysis into a Software Requirements Specification (SRS). The SRS has the form of a structured statement or a Use Case Diagram (UCD) (Ahmed and Prasad, 2016). In some organizations, the SRS is written in a standard formal document called the SRS document. Generally, the SRS is expressed in qualitative textual or natural language form (Ahmed and Prasad, 2016; Calisaya, 2016; Dalpiaz et al., 2018).

FSM can be done by creating a Business Process Model and Notation (BPMN) diagram, as in (Monsalve et al., 2011; Marín et al., 2014; Khlif et al., 2017). The complexity of the BPMN diagram can also be measured (Yaqin et al., 2020), making it possible to measure the FSM and its complexity. Initially, BPMN diagrams were used to model organizational business processes, but over time they also started being used for requirement elicitation (Monsalve et al., 2012; Przybyłek, 2014). Ref. (Vega-Márquez et al., 2019) reported no difference between BPMN diagrams and UCDs in the comprehensive understanding of the requirement specifications stated in these diagrams. However, BPMN diagrams are better than UCDs for expressing sequential activities, so BPMN diagrams are more suitable to use as an intermediate representation to measure the functional size.

* Corresponding author.
E-mail address: riyanarto@if.its.ac.id (R. Sarno).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.10.007
1319-1578/

Several studies have proposed conversion from textual input to BPMN diagrams. Refs. (Friedrich et al., 2011; Gonçalves et al., 2011) used the Cooperative Requirements Engineering With Scenarios (CREWS) method as an intermediate representation before obtaining the BPMN diagram. Both studies used Natural Language (NL) input and produced BPMN output. Unfortunately, they did not consider all sentence structures (no complex sentences) as NL input, and the method proposed in (Friedrich et al., 2011) requires sequential NL input, so it does not allow for iterative processing. At the same time, (Gonçalves et al., 2011) did not test the effectiveness of the proposed method, as it was only tested with two case studies. While (Epure et al., 2015) used semi-structured text input, it focused on finding activities in NL input, not on generating the BPMN diagram. Ref. (Ferreira et al., 2017) extracted process model elements from the NL input and not from a complete BPMN diagram. A similar study was conducted by Honkisz et al. (2018), who used a spreadsheet-based description as an intermediate presentation to produce the BPMN diagram. An advantage of this approach is that it is straightforward to obtain the BPMN diagram, but it has three main weaknesses. Firstly, it can only handle limited BPMN diagrams (only tasks and gateways) and cannot deal with intermediate events, pools, lanes, datastores, and data objects. Secondly, no validation was done of the activity extraction process. Thirdly, it was not tested using complex and compound sentence structures in the NL input.

The present paper proposes conversion from textual requirements to BPMN diagrams while overcoming two weaknesses of the previous studies. Firstly, the proposed method can handle NL with various sentence structures, i.e., simple, compound, complex, and compound-complex sentences. Secondly, it can produce BPMN diagrams based on textual input involving all BPMN elements, including activities, intermediate events, pools, lanes, datastores, and object data.

The remainder of this paper is organized as follows. Section 2 contains related work and reviews several previous studies similar to this study. Section 3 describes the proposed method, and Section 4 describes and discusses the obtained results. Section 5 contains the conclusion of this study.

2. Related work

The authors of (Ramzan et al., 2014; Bajwa et al., 2011) reported that they had succeeded in automatically translating NL into Semantics of Business Vocabulary and Rules (SBVR). This approach uses a set of rules to transform NL into SBVR. The method from (Bajwa and Choudhary, 2012) converts NL to UML diagrams, converting the specifications in the SBVR to class diagrams. The test results showed that this approach was more accurate than existing conversion tools. Bajwa et al. (2012) used a slightly different approach, transforming NL into Object Constraint Language, a formal specification language used to define constraints for UML modelling. The method proposed by Btoush and Hammad (2015) uses a combined approach of Natural Language Processing (NLP) and rules to transform NL into entity-relationship diagrams. The extracted part-of-speech (POS) results are mapped onto entities, attributes, and relations to produce ER diagrams with a rule set. Thus, all three studies used rule sets to convert NL to either SBVR or UML diagrams and data models, but they did not test the proposed methods with NL input that included complex and compound sentences.

The studies used SBVR input to be transformed into several UML diagrams and data models. The method proposed in Ref. (Afreen et al., 2011) uses informal mapping rules to convert SBVR to a UML class diagram. The conversion process is carried out automatically using an Eclipse plug-in called SBVR2UML, with an average precision of 94.87 %. Essebaa and Chantit (2018) reported that they created an Eclipse plug-in application to convert SBVR to UML use case diagrams using a set of transformation rules. Bonais et al. (2016) used a formal transformation approach to convert SBVR to UML class diagrams. Iqbal and Bajwa (2016) automatically converted the requirement specifications in the SBVR into an activity diagram. Their method obtains the requirement specifications in the SBVR as input and then parses the input and selects UML objects such as fact types, nouns, verbs, and others. It produces a visual presentation in the form of activity diagrams. Data model generation (Bajwa et al., 2012) is reported in (Bajwa et al., 2016), where an Express data model is generated from SBVR by applying rules. Meanwhile, Ref. (Tangkawarow et al., 2021) focuses on introducing inclusive and exclusive gateways in the SBVR to generate a BPMN diagram, and Ref. (Quishpi et al., 2020) focuses on extracting textual descriptions into process annotations that describe key process elements such as actions, events, agents/patients, roles, and control flow relationships.

As for the transformation from NL input to the process model, Ref. (van der Aa et al., 2018) points out some of its opportunities and challenges. The opportunity is because of the capability and technology of NLP that allows it to be used to practice Business Process Models at different levels, while challenges can arise at all levels, namely multi-process, process models, and process instance management, including conversion from textual descriptions to process models (van der Aa et al., 2018). More specifically, these Refs (Mendling et al., 2019; Mendling et al., 2014) show 25 challenges related to NLP with process models.

The previous studies that are similar to our research are summarized in Table 1. All studies used NL input and produced BPMN output. Refs. (Friedrich et al., 2011; Gonçalves et al., 2011) used Cooperative Requirements Engineering With Scenarios (CREWS) as an intermediate representation before generating the BPMN diagram. Both studies did not involve complex sentences in the NL input, while (Friedrich et al., 2011) can only accept sequential NL. The conversion method from a textual description to a BPMN process model in (Friedrich et al., 2011) is automated using the web-based application in this Ref. (Delicado et al., 2017). In Ref. (Gonçalves et al., 2011), the effectiveness of the proposed approach was not evaluated since it was only tested with two case studies. In contrast, Ref. (Sawant et al., 2014) used a textual use case as input. Ref. (Epure et al., 2015) processed semi-structured text using NLP, focusing on semantic verbs for process mining and rule bases for activity relations, not on the creation of the BPMN diagram. Ref. (Ferreira et al., 2017) did not produce a complete BPMN diagram but integrated NLP processes and mapping rules to extract processes from text input. The study conducted by Sonbol and Ghneim (2022) used a concept map of words to generate the BPMN diagram.

The approach most similar to the present research is the one from Honkisz et al. (2018), using a spreadsheet-based description as an intermediate presentation to produce the BPMN diagram. Its main advantage is that it is straightforward to obtain the BPMN diagram, but it also has some limitations. As Point 6 in Table 1 notes, the limitations of this technique are: (i) it is only able to handle the standard structure of a limited BPMN diagram (only tasks and gateways) and is not able to deal with intermediate events, pools, lanes, datastores, and data objects; (ii) the activity extraction process was not validated; (iii) the method was not tested with NL input containing various sentence structures, specifically complex and compound sentences.

As in the other previous studies, the resulting BPMN diagram cannot handle several BPMN elements, such as intermediate events, pools, lanes, datastores, and data objects. Most previous studies have not been tested with NL input that contains complex and compound sentences. Our study applied several integrated NLP methods, SBVR definitions, and spreadsheets to improve the limitations of these studies.

Table 1
Summary of studies similar to this study.

No	Authors, year, paper	Input	Method	Output	Intermediate Representation	Limitations
1	Friedrich et al. (2011)	Sequential NL	Semantic analysis	BPMN	CREWS	- Text input does not involve complex sentences - Text input must be sequential
2	Gonçalves et al. (2011)	NL	NLP	BPMN	CREWS	- Text input does not involve complex sentences - The effectiveness of the proposed method was not evaluated because only two case studies were used
3	Sawant et al. (2014)	Textual use case	Rule base	BPMN	GATE Annotations	Requires more structured input in the form of a textual use case
4	Epure et al. (2015)	Semi-structured text	Rule set	BPMN	Structured table	Focuses more on the miner's activities and their relationship with the miner's activities, not on other BPMN elements
5	Ferreira et al. (2017)	NL	Mapping rules	Process model elements	–	Does not form a complete BPMN diagram
6	Honkisz et al. (2018)	NL	NLP + spreadsheet-based description	BPMN	Spreadsheet	- Only able to handle standard BPMN structure (limited task and gateway only), no intermediate events, pools, lanes, datastore, and data object - No validation of the activity extract process - No test for several sentence types in NL for simple, compound, and complex sentences
7	(Sonbol and Ghneim, 2022)	NL	NLP + concept map + mapping rules	BPMN	Concept map	- The method used is too complicated - It cannot handle input containing complex sentences and compound-complex
8	The proposed method	NL	NLP/SBVR definition + rules set + spreadsheet-based description	BPMN	Spreadsheet	–

Abbreviations: NL = Natural Language, NLP = Natural Language Processing, BPMN = Business Process Model and Notation, CREWS = Cooperative Requirements Engineering with Scenarios, GATE = General Architecture for Text Engineering.

3. Proposed method

This section describes the proposed method for generating BPMN diagrams with input from textual requirements. The textual requirements are processed using NLP to get the fact types. Then, the fact types are mapped onto BPMN elements, which are then used to create the BPMN diagram via a spreadsheet-based description. The proposed method consists of two phases: 1) analysis of textual requirements with natural language processing and 2) BPMN generation. Each stage is described in detail in the following sub-sections.

3.1. Analysis of textual requirements with NLP

This phase consists of four processes, namely preprocessing, syntax analysis, semantic analysis, and extracting fact types. Some of the processes and their details are:

3.1.1. Pre-processing

Preprocessing begins with splitting, i.e., separating the textual requirements consisting of multiple sentences into separate sentences, which are then stored in an array list. Next follows tokenization, i.e., breaking each sentence into separate words (tokens). For example, the sentence 'An officer selects a menu to accept a new member registration request.' is tokenized to '[An] [officer] [selects] [a] [menu] [to] [accept] [a] [new] [member] [registration] [request][.]'.

3.1.2. Syntax analysis

The following process is syntax analysis, consisting of two processes: part-of-speech (POS) tagging and generating parse, or dependency trees. We use Stanza (version 1.3.0) (Qi et al., 2020) for POS tagging, identifying each token as a noun, verb, preposition, etc. For example, for the sentence 'An officer selects a menu to accept a new member registration request.' the token tags are {An/DT officer/NN selects/VBZ a/DT menu/NN to/TO accept/VB a/DT new/JJ member/NN registration/NN request/NN.}.

After this, dependency parsing is performed using Stanza, creating a word tree structure to represent the structure of the textual input. This tree structure can be used to identify the relationships among the various syntactic structures of the textual input. As an example, the dependency parsing for the sentence 'An officer selects a menu to accept a new member registration request.' is shown in Fig. 1.

3.1.3. Semantic analysis

Next is Semantic Role Labeling (SRL), which automatically labels each token in a sentence. We follow (Bajwa, 2012; Bajwa et al., 2011), where the roles are based on SBVR elements, including noun/general concept, individual noun concept, verb concept attribute, quantification, and others (Omg, 2019).

3.1.4. Fact type extraction

Before identifying the fact types using the proposed rules, we briefly review sentence structures and conjunctions (and disjunctions):

- Sentence structures
  There are four sentence structures: simple, compound, complex, and complex-compound (Elizabeth O'Brien). A simple sentence contains only one independent clause, i.e., a group of words containing a subject, a predicate, and expressing a complete thought. An example of a simple sentence is: 'An officer selects a menu to accept a new member registration request.'.

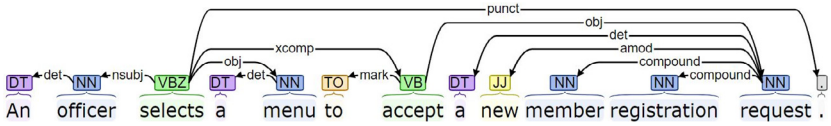

Fig. 1. The dependency parsing of the input: 'An officer selects a menu to accept a new member registration request.'.

Table 2
Set of rules for extracting fact types from a textual requirement.

	Set of rules to extract fact types
1	Separate a textual requirement input into its constituent sentences
2	Break up a compound or complex sentence into as many simple sentences as there are clauses in the sentence.
3	For passive voice, swap the subject and object; the subject of the passive voice becomes the object, and vice versa, the passive object becomes the subject.
4	For simple sentences:
	a) The subject consists of several nouns separated by commas or/and conjunctions (and, or, but, however, so, however, although). Break the sentence into several new sentences, as many as there are nouns in the subject.
	b) For predicates consisting of several verbs separated by commas or/and conjunctions, break the sentence into several new sentences, as many as there are verbs in the predicate.
5	Identify the fact types of subject and predicate pairs.
	a) Pairs of subject + predicate (SP) are identified as Unary Fact Type (UFT) or Characteristic (C).
	b) Pairs consisting of subject + predicate + object (SPO) are identified as Binary Fact Type (BFT). If the verb of the predicate has an adverb of degree or an adverb of manner, then the BFT may be one of the three following types:
	1. Strong BFT: if the adverb means very much, very loudly, completely, or fully, such as: absolutely, completely, excessively, fully, entirely, highly, totally, intensely, heavily, hardly, thickly, deeply, broadly, richly, fatly, highly, long, loudly, widely, dearly, expensively, etc.
	2. Weak BFT: if the adverb means partially, not much, rarely, only lightly, or slowly, such as: partly, almost, nearly, slowly, lightly, easily, thinly, shallowly, narrowly, cheaply, poorly, bluntly, etc.
	3. BFT: no adverb as either weak BFT or strong BFT.

A compound sentence has at least two independent clauses, which are connected by coordinating conjunctions such as 'for', 'and', 'nor', 'but', 'or', 'yet', and 'so'; these are called glue words, phrases, or clauses. An example of a compound sentence is: 'An officer selects a menu to accept a new member registration request, or he rejects it.' Meanwhile, if a sentence has at least one independent clause and a subordinate clause, it is called a complex sentence. Subordinate clauses, which contain 'after', 'although', 'as', 'because', 'before', 'even though', 'if', 'since', 'though', 'unless', 'until', 'when', 'whenever', 'whereas', 'wherever', 'while' or the relative pronouns 'who', 'which', or 'that', are groups of words with a subject and a predicate that do not express a complete thought. An example of a complex sentence is: 'After an officer selects a menu to accept a new member registration request, he rejects it.'. Finally, a compound-complex sentence combines a compound and a complex sentence. An example of a compound-complex sentence is: 'After an officer selects a menu to accept a new member registration request, he rejects it, and then he closes application.'.

- Conjunctions and disjunctions
  In English, conjunctions are usually stated with words such as 'and', 'yet', 'but', 'so', 'however', 'moreover', 'even though', and 'although' (Bajwa et al., 2012). In simple sentences, conjunctions connect two nouns or two verbs. Conjunctions can also connect clauses in compound sentences or complex sentences. Disjunctions can be either inclusive or exclusive. An inclusive disjunction is expressed with the words 'or', 'unless', or 'and/or' (Bajwa et al., 2012), while an exclusive disjunction is used to express its implication with the word 'if', for example, 'if (grade >= 65), then status = passed'.
- Identifying fact types
  The final step of the textual requirements analysis with NLP consists of obtaining the fact types. The combination of a noun concept and a verb concept is a fact type. In SBVR 1.5, a fact type is formed by verb phrases with one or more verb concept roles. Textual requirements can include unary fact types (or characteristics), binary fact types, and fact types with three or more roles (Omg, 2019).

A fact type with one role is known as a unary fact type, for example, 'the school is white'. A fact type that has a dual role is called a binary fact type, for example, 'An officer selects the menu', where 'An officer' and 'a menu' are two roles, and 'selects' is a verb related to these roles. A fact type that has three or more roles is called a three or more fact type and has at least three roles. For example, in the sentence 'An officer shows information to a student', the predicate 'shows...to' relates to three roles: the officer, the information, and the student.

- Rules to extract fact types
  The rules for extracting fact types from a textual requirement are given in Table 2:

To be more detailed and operational, the rules set in Table 2 are implemented using the flowchart in Fig. 2. In the flowchart, numbers are given in small circles that indicate the rule numbers in Table 2.

3.2. BPMN diagram generation

In this step, the BPMN diagram is obtained. There are three activities: mapping the fact types onto BPMN elements, making a spreadsheet-based description, and creating the BPMN diagram. The details are discussed briefly below.

3.2.1. Mapping the fact types onto BPMN elements

The rules for mapping the fact types onto BPMN elements are given in Table 3.

3.2.2. Generating the spreadsheet-based description

After obtaining the BPMN elements, the next step is generating the BPMN diagram. To construct the BPMN diagram with BPMN elements, we use the spreadsheet-based description proposed by Kluza and Wisniewski (2016), modified by adding a datastore/data object column. An example of a spreadsheet-based description expressed in tabular form is shown in Table 6. The formation of the spreadsheet-based description can be seen in Section 4.2.

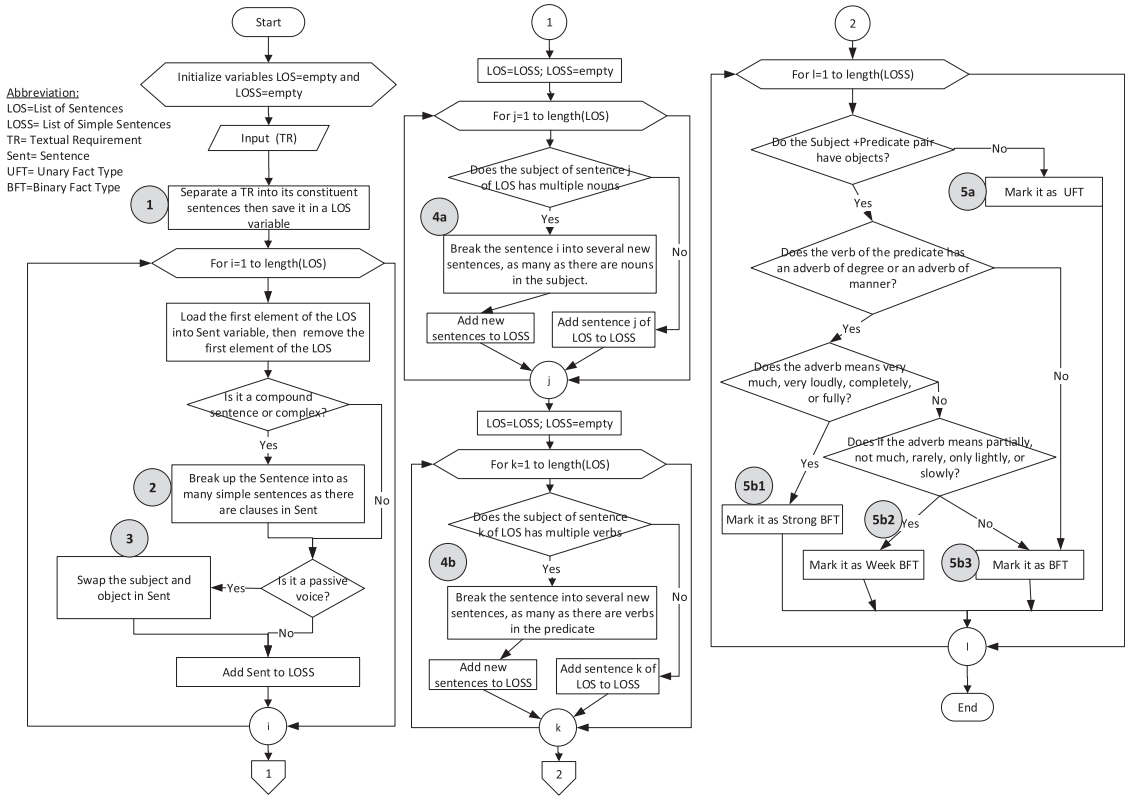

Fig. 2. The flowchart to extract fact types from textual requirements input.

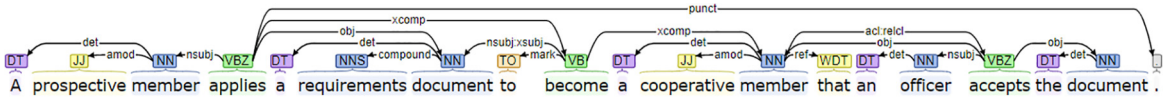

Fig. 3. The dependency parsing result for the first sentence in the case study 9.

3.2.3. Generating the BPMN diagram

In this step, the BPMN diagram is generated based on the spreadsheet-based description. Fig. 4 shows the BPMN diagram of the spreadsheet-based description in Table 6. Likewise, the process of constructing a BPMN diagram from a spreadsheet-based description is given in Section 4.2.

4. Result and discussion

This section is divided into three parts: textual requirements input format, examples of getting a BPMN diagram for a case study, and the test results for several case studies. Each is discussed successively in the following sub-sections.

4.1. The input format of textual requirements

We need to provide rules for the textual requirements input format to get the desired result. In this case, we used the following directions:

1 The textual requirements input consists of several sentences, where each sentence must include at least one subject and one predicate.
2 Each sentence in the textual requirements can be either simple, compound, complex, or compound-complex.
3 If we compose serial verbs as predicates or consecutive nouns as subjects, we use a comma (,) without the conjunction 'and' or the disjunction 'or'. If it is not a sequential process, we can use conjunction (or a disjunction). For example, 'An officer creates a new member and adds a new member to the database member.' The text has the conjunction 'and'; if it is intended

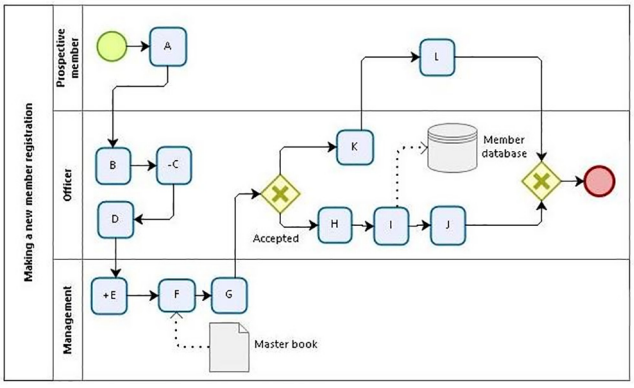

Fig. 4. The BPMN diagram generated for case study 9.

Table 3
Informal mapping from fact types onto BPMN diagram elements.

No	Element of NL	Logical operation	BPMN element	Notation
1	Title of requirement	–	Pool	
2	Subject of FT	–	Lane	
3	Weak BFT/BFT/Strong BFT	–	Weak activity/ Activity/ Strong activity	
4	UFT or C, unless UFT is in an if-then condition	–	Event	
5	FT in implication 'if…then' or 'if…then…else/otherwise.'	Implication	XOR gateway	
6	FT and FT (other conjunctions: 'yet', 'but', 'so', 'however', 'moreover', 'even though', and 'although')	Conjunction	AND gateway	
7	FT or FT (other disjunctions: 'unless' or 'and/or')	Disjunction	OR gateway	
8	FT or FT but not both	Exclusive disjunction	XOR gateway	
9	Not both FT and FT	Nand formulation	XOR gateway	
10	Neither FT nor FT	Nor formulation	XOR gateway	
11	FT whether or not FT	Whether-or-not formulation	XOR gateway	
12	The noun (or noun phrase) of either the second object phrase or adverb phrase of the concept verb in FT, which contains the **keyword verbs** (listed below), is mapped onto an inflow/ outflow datastore/object. If the object phrase includes the word 'database', it becomes a datastore. Otherwise, it becomes a data object. The keyword verbs for inflow: **write, add, keep, record, register, save, store, hold, update, edit, or file**. The keyword verbs for outflow: **read, review, check, verify, get, obtain, scan, retrieve, query, filter, or take**.	–	Inflow datastore Inflow data object Outflow datastore Outflow data object	

Abbreviations: NL = Natural Language, NLP = Natural Language Processing, BPMN = Business Process Model and Notation, BFT = Binary Fact Type, UFT = Unary Fact Type, C = Characteristic, FT = Fact Type.

for consecutive verbs, then the conjunction 'and' is removed, and the sentence becomes 'An officer creates a new member, adds a new member to the member database.'

4 Use the keyword 'go-to' followed by a < number > for jumping or looping. At the same time, the destination of the jump or loop is marked with an asterisk (*) followed by < number >. For example, for the looping 'go-to 2′, the destination is '*2′.

5 To read data from a datastore/object, use the following keyword verbs: **read, check, verify, review, get, obtain, scan, retrieve, query, filter,** or **take**. Meanwhile, to store data into a datastore/object, use the following keyword verbs: **write, add, keep, record, register, save, store, hold, update, edit, or file**. For example, 'The officer adds a new member to the member database.'

4.2. Example of obtaining the BPMN diagram with a case study

As an example, we use the cooperative application's case study 'Making a new member registration' (case study 9 in Appendix A) to get a BPMN diagram from textual requirements. The case study demonstrates simple, compound, and complex sentences in the textual requirements input. It also exhibits multiple verbs (series

Table 4
Results of SRL and fact type extraction for the case study.

SBVR concept	Count	Detail
Noun/individual concept	12	Member, requirement, document, officer, management, processing, master, book, status, result, database, and card
Verb concept	17	Apply, become, accept, have, be, check, send, review, base, assign, is, create, add, print, return, complete, and receive
Characteristic (C)	1	C1: The status is accepted
Binary fact type (BFT)		BFT1: A prospective member applies a requirements document
	7	BFT2: A officer accepts the document
		Week BFT3: The officer has lightly checked document
		BFT4: The officer sends the document
		Strong BFT5: Management fully reviews the document
		BFT6: Management assigns a status to a document
		BFT7: Management sends the document back to the officer
		BFT8: The officer creates a new member
		BFT9: The officer adds a new member
		BFT10: The officer prints the cooperative card
		BFT11: The officer returns the requirements document
		BFT12: The prospective member receives the requirements document

Abbreviations: SBVR = Semantics of Business Vocabulary and Rules, BFT = Binary Fact Type, C = Characteristic.

of verbs), passive voices, transitive and intransitive verbs, adverb of manners, and keyword verbs for datastore. The textual requirements are given as follows:

'A prospective member applies a requirements document to become a cooperative member that an officer accepts the document. Once the document has been lightly checked by the officer, the officer sends the document to the management for further processing. Management fully reviews the document based on the cooperative master book. Management assigns a status to a document based on the review results, so management sends the document back to the officer. If the status is accepted, then the officer creates a new member, adds a new member to the member database, prints the cooperative card; otherwise, the officer returns the requirements document to the prospective member to complete the requirements, then the prospective member receives the requirements document to be completed.'.

The textual requirement above consists of five sentences. The first is a complex sentence, and the second is a compound sentence, while the third is a simple sentence containing the verb 'review', which has the adverb 'fully', and the fourth is a compound sentence connected by the conjunction 'so'. Finally, the last sentence is a complex sentence with if-then-else, which also includes a sequential verb and a key verb for storage in the database.

Table 5
Extraction process in the set of rules applied in the case study.

No	Process	Short description	Example of input	Output
1	Rule 1	Rule 1 breaks the NL input sentence in case study 9 into 5 separate sentences.	The textual requirements for case study 9 (see Appendix A)	Five sentences that make up the textual requirements
2	Rule 2	In this example, the input is a complex sentence consisting of two clauses connected by a 'that' connector.	**Sentence 1**: A prospective member applies a requirements document to become a cooperative member that an officer accepts the document.	- A prospective member applies a requirements document to become a cooperative member. - An officer accepts the document.
3	Rule 2	An example of input is a compound sentence with two clauses separated by a 'so' connector.	**Sentence 4:** Management assigns a status to document based on the review results, so management sends document back to officer.	- Management assigns a status to document based on the review results. - Management sends document back to officer.
3	Rule 3	In this example, first clause of the input sentence is a passive voice. Using Rule 3, the passive voice is reversed into an active sentence.	**Sentence 2:** Once the document has been lightly checked by the officer, the officer sends the document to the management for further processing.	- The officer has lightly checked the document - The officer sends the document
4	Rule 4	This example demonstrates Rule 4b which breaks the input sentence into as many new sentences as the successive verbs in the predicate.	**A clause in Sentence 5**: The officer creates a new member, adds a new member to the member database, prints the cooperative card.	- The officer creates a new member - The officer adds a new member to the member database - The officer prints the cooperative card
5	Rule 5	This example demonstrates Rule 5a which identifies the input as Characteristic	**Clause 1 in Sentence 5:** if the status is accepted	'The status is accepted' as Characteristic
6	Rule 5	This example demonstrates Rule 5b3 as a rule to identify common fact types.	**Clause 2 in Sentence 5:** The officer creates a new member	'The officer creates a new member' as Binary Fact Type (BFT)
7	Rule 5	Example for the Rule 5b1 that identifies the presence of adverb of manners for Strong BFT.	**Sentence 3:** Management fully reviews the document based on the cooperative master book	'Management fully reviews the document based on the cooperative master book' as Strong BFT
8	Rule 5	Instead, this is an example to identify the presence of Weak BFT with rule 5b2. The input has gone through the process of reversing from passive voice into active sentence using Rule 3.	**Clause 1 in Sentence 2:** The officer has lightly checked the document	'The officer has lightly checked the document' as Weak BFT

Table 6
Spreadsheet-based description for the case study of 'Making a new member registration'.

Order	Activity/event (label)	Condition	Who	Datastore/data object	Terminated
0	Start	–	–	–	–
1	Applies a requirements document (A)	–	Prospective member	–	–
2	Accepts the document (B)	–	Officer	–	–
3	Lightly checked document (-C)	–	Officer	–	–
4	Sends the document (D)	–	Officer	–	–
5	Fully reviews the document (+E)	–	Officer	Mater book/ datastore/ outflow	–
6	Assigns a status to document (F)	–	Management	–	–
7	Sends document back to officers (G)	–	Management	–	–
8a1	Creates a new member (H)	The status is accepted	Officer	–	–
8a2	Adds a new member (I)	The status is accepted	Officer	Member database/ datastore/ inflow	–
8a3	Prints the cooperative card (J)	The status is accepted	Officer	–	–
8b1	Returns the requirements document (K)	Otherwise	Officer	–	–
8b2	Receives the requirements document (L)	–	Prospective member	–	–
9	End	–	–	–	Yes

Table 7
The identified fact types from the ten textual requirements.

No	Requirement	Description	Fact types identified
1	Making a cash transaction	NL input demonstrates simple sentences.	- BFT1 = An officer submits evidence of a transaction (A) - BFT2 = An accountant records the cash transaction (B)
2	Withdraw mandatory savings	NL input consists of two sentences. The second is a compound sentence related to the word 'so' and has two clauses, so there are three clauses in the NL input.	- BFT1 = Management wants to withdraw the mandatory savings (A) - BFT2 = An officer displays the mandatory savings (B) - BFT3 = The officer processes the withdrawal of the mandatory savings (C)
3	Resign a member	There are three sentences. The third demonstrates sequential verbs or compound predicate with four verbs, and the keyword verb for datastore.	- BFT1 = Management asks a member (A) - BFT2 = An officer accepts management's request (B) - BFT3 = The officer reads information (C) - BFT4 = The officer updates the member's status (D) - BFT5 = The officer informs the management (E)
4	Making a cash outflow report	There are two sentences in NL input. The first demonstrates a complex sentence consisting of adjective clauses. The first also contains a passive clause and the keyword verb for datastore. The second demonstrates a complex sentence with if-then-otherwise and has a verb sequence.	- BFT1 = Management orders a cash-out report (A) - BFT2 = Management gives the year and month (B) - BFT3 = An accountant filters out cash transactions (C) - C1 = The query is not empty - BFT4 = The accountant prints a cash-out report (D) - BFT5 = The accountant gives it (E) - C2 = There is no cash-out (F)
5	Display a member	NL input consists of four sentences. The third has the keyword verb for datastore. The fourth is a complex sentence containing if-then-else clauses and sequence clauses.	- BFT1 = A member needs to display detailed information (A) - BFT2 = An officer accepts member requests (B) - BFT3 = The officer reads a member's information (C) - C1 = The member exists - BFT4 = The officer reads the work unit data (D) - BFT5 = The officer provides the detailed member information (E) - BFT6 = The officer confirms no information (F)
6	Making a bill report	NL input consists of three sentences. The third is a complex sentence with an if-then-else clause, conjunction 'and', and an intransitive sentence.	- BFT1: A member needs a monthly billing statement (A) - BFT2 = An officer reads the member's billing data (B) - C1 = The monthly bill exists. - BFT3 = The officer prints the member's bill (C) - BFT4 = The officer informs the member (D) - C2 = The bill does not exist (E)
7	Changing a work unit	The first is a compound sentence consisting of two clauses, while the second and third are simple sentences with the keyword verb for datastore.	- BFT1: A member submits changes to the work unit (A) - BFT2 = An officer accepts the member's requests (B) - BFT3 = The officer reads the member and work unit information (C) - BFT4 = The officer updates the member information (D)
8	Realizing a member loan	NL input has five sentences. The third has a sequential verb. The fourth has the conjunction 'and' on the object. The fifth has a sequential verb and the keyword verb for datastore.	- BFT1: Management wants to realize member loans (A) - BFT2 = An officer receives orders for member loan realization (B) - BFT3 = The officer opens the member loan realization menu (C) - BFT4 = The officer sets the loan type (D) - BFT5 = The officer enters the member ID (E) - BFT6 = The officer sets the instalment amount, principal instalment, and repayment period (F) - BFT7 = The officer realizes the member loans (G) - BFT8 = The officer saves the loan realization information (H)
9	Making a new member registration	The first sentence is complex with adjective clause, while the second has a passive clause and an adverb 'lightly'. The third is a simple sentence containing the adverb 'fully', and the fourth is a compound sentence. The fifth is a complex sentence containing clauses (if-then-else/otherwise), and a clause that has consecutive verbs.	- BFT1: A prospective member applies a requirements document (A) - BFT2: A officer accepts the document (B)

Table 7 (*continued*)

No	Requirement	Description	Fact types identified
			- Week BFT3: The officer has lightly checked document (-C) - BFT4: The officer sends the document (D) - Strong BFT5: Management fully reviews the document (+E) - BFT6: Management assigns a status to a document (F) - BFT7: Management sends the document back to the officer (G) - C1: The status is accepted - BFT8: The officer creates a new member (H) - BFT9: The officer adds a new member (I) - BFT10: The officer prints the cooperative card (J) - BFT11: The officer returns the requirements document (K) - BFT12: The prospective member receives the requirements document (L)
10	Selecting a member loan request	This case study is the most complex. There are several remarkable things in this case study, including: The NL input has a looping process from the third sentence to the first. The third sentence is compound complex. The third and fourth sentences have the keyword verb 'check', while the fourth has the conjunction 'and'. The fifth is a complex sentence with an if-then-else with the if-condition having the disjunction 'or'.	- BFT1: A member submits a loan request (A) - BFT2: The credit section accepts the member's loan requests (B) - BFT3: The credit section checks the credit requirements documents (C) - C1: The credit requirements document is incomplete. - BFT4: The credit section returns it (D) - BFT5: The credit section checks the mandatory deposit (E) - BFT6: The credit section checks the current loan (F) - *C2*: The mandatory deposit has not been paid off. - C3: The current loan exists. - C4: The submission status is rejected (G) - C5: The submission status is okay (H) - BFT7: The credit section sends the submission document (I) - BFT8: The officer receives the document (J)

Abbreviations: BFT = Binary Fact Type, C = Characteristic.

The first process analyzes the textual requirements, including splitting, tokenization, POS tagging, generating dependency parsing, SRL, and fact type extraction. The textual requirements of the input outlined in the five sentences and the tokenization for the first sentence is '[A] [prospective] [member] [applies] [a] [requirements] [document] [to] [become] [a] [cooperative] [member] [that] [an] [officer] [accepts] [the] [document][.]'. Then, the POS tagging result for the first sentence is '{A/DT prospective/JJ member/NN applies/VBZ a/DT requirements/NN document/NN to/TO become/VB a/DT cooperative/JJ member/NN that/WDT an/DT officer/NN accepts/VBZ the/DT document/NN./}', while Fig. 3 shows the dependency parsing result.

The following process is SRL, which in this study means extracting the fact types and obtaining noun/individual concepts and verb concepts as the primary components of the fact types. The results of SRL and the fact types extracted from the input text using set of rules (Table 2 or Fig. 2) are shown in Table 4. Twelve binary fact types (BFT1 to BFT12) and one characteristic (C1) were identified in the case study.

For a more detailed explanation of the process in the set of rules in Table 2 or Fig. 2 starting from Rule 1 to Rule 5, it is given in Table 5.

Next is the generation of the BPMN diagram, which consists of mapping the fact types obtained onto BPMN elements, creating a spreadsheet-based description, and finally creating the BPMN diagram. When we substitute the fact types that have been acquired into the input text, we get the following: **'BFT1 that BFT2. Once Weak BFT3, BFT4. Strong BFT5. BFT6, so BFT7. If C1, then BFT8, BFT9, BFT10; otherwise, BFT11, then BFT12.'.** Mapping the fact types onto BPMN elements using.

Table 3 as a reference, BFT1, BFT2, BFT4, and BFT6 to BFT12 becomes Activities, Weak BFT3 becomes Weak activity, Strong BFT5 becomes Strong activity, C1 becomes an event, and If. . .Then. . .Otherwise becomes an XOR branch.

After mapping the fact types onto BPMN elements, the next step is to create the spreadsheet-based description, which is given in Table 6. The start and end events are assigned '0′ and 'last' in the spreadsheet-based description. Orders 1 to 7 are activities A to G obtained from mapping BFT1 to BFT7, while orders 8a1, 8a2, and 8a3 are activities H, I, and J (mapping from BFT8 to BFT10), which are in IF with conditions C1. Orders 8b1 and 8b2 are activities K and L (from BFT11 and BFT12), which are in Otherwise.

Furthermore, based on Table 6, the BPMN diagram in Fig. 4 is generated. The process starts by creating a pool with the name 'Making a new member registration' according to the name of the functional process. Because in the Who column, there are 'Prospective member ', 'Officer', and 'Management', so there are three lanes, namely 'Prospective member', 'Officer', 'Management'. According to the Who column, activities or events are posted in their respective lanes. An activity can have datastores/data objects. For example, activity + E has a 'Master book/datastore/outflow'. Therefore, it has the datastore 'Master book' with the outflow direction. Order 8 has a branch with 8a and 8b, where the branch is an XOR gateway (with the condition in the Condition column). Order 8a consists of 8a1, 8a2, and 8a3 with the sequence of activities H, I, and J, and 8b consists of 8b1 and

8b2 with sequential activities K and L until the Terminated column is filled with 'end'.

4.3. Test results using several case studies

This section displays the test results of the ten case studies of the textual requirements (Appendix A). The test results are given in Table 7, and the BPMN diagram generated is shown in Appendix B. Each textual requirement demonstrates a specific case, as described in the Description column.

Each case demonstrates a different textual input, ranging from simple sentences (Case 1) to compound complex sentences (Case 10). Also, it shows sequential processes (Case 1 to Case 3, Case 7 to Case 8), a gateway process (Case 4 to Case 6, Case 9), and an iterative process (Case 10). The process of reading and updating from/to the datastore is demonstrated in several cases, namely Case 3 to Case 10. Meanwhile, Case 9, apart from demonstrating the gateway process and reading data from the datastores, also shows Strong and Weak activities.

4.4. Evaluation and discussion

The previous section tested the proposed method with ten textual requirements of a cooperative application. In this section, we examine the proposed method with two comparisons. First, we compare the results of this study with previous studies, and then we test the results of the proposed method with the results of manual processing by an expert. Each is discussed in a separate subsection.

Table 8
Comparison test results with similar previous studies.

No of req	Demonstrate	The proposed method	Honkisz et al. (2018)	Ferreira et al. (2017)	Epure et al. (2015)	Sonbol and Ghneim (2022)
1	Simple sentence	√	√	√	√	√
2	- Simple sentence	√	√	√	√	√
	- Compound sentence	√	√	√	√	√
3	- Simple sentence	√	√	√	√	√
	- Sequential verbs	√	×	√	√	×
	- Keyword verb for datastore	√	×	×	×	×
4	- Complex sentence with adjective clauses	√	×	×	×	√
	- Passive voice	√	×	√	√	×
	- Keyword verb for datastore	√	×	×	×	×
	- Complex sentence with if-then-otherwise	√	√	√	√	√
	- Sequential verbs	√	×	√	√	×
	- Intransitive clause	√	×	×	×	×
5	- Simple sentence	√	√	√	√	√
	- Keyword verb for datastore	√	×	×	×	×
	- Complex sentence with if-then-else	√	√	√	√	√
	- Intransitive clause	√	×	×	×	×
	- Sequential clauses	√	√	√	√	√
6	- Simple sentence	√	√	√	√	√
	- Keyword verb for datastore	√	×	×	×	×
	- Complex sentence with if-then-otherwise	√	√	√	√	√
	- The conjunction 'and'	√	√	√	√	√
	- Intransitive clause	√	×	×	×	×
7	- Compound sentence	√	√	√	√	√
	- Keyword verb for datastore	√	×	×	×	×
	- Simple sentence	√	√	√	√	√
8	- Simple sentence	√	√	√	√	√
	- The conjunction 'and' on object	√	√	√	√	√
	- Sequential verbs	√	×	√	√	×
	- Keyword verb for datastore	√	×	×	×	×
9	- Complex sentence with adjective clause	√	×	×	×	√
	- Complex sentence with if-then-else	√	√	√	√	√
	- Simple sentence	√	√	√	√	√
	- Passive voice	√	×	√	√	×
	- Adverb of manners	√	×	×	×	×
	- Keyword verb for data object	√	×	×	×	×
	- Keyword verb for datastore	√	×	×	×	×
	- Intransitive clause	√	×	×	×	×
	- Sequential verbs	√	×	√	√	×
10	- Looping process	√	×	×	×	√
	- Simple sentence	√	√	√	√	√
	- Compound-complex sentence	√	×	×	×	√
	- Complex sentence with adjective clause	√	×	×	×	√
	- Complex sentences with if-then-else	√	√	√	√	√
	- Keyword verb for data object	√	×	×	×	×
	- Keyword verb for datastore	√	×	×	×	×
	- The conjunction 'and'	√	√	√	√	√
	- The disjunction 'or'	√	√	√	√	√
	- Intransitive clause (1)	√	×	×	×	×

Abbreviation: Req = Requirements.

4.4.1. Comparison with previous studies

A comparison between the proposed method and some previous similar studies is given in Table 8. The comparison was applied to the ten test cases of the requirements shown in Appendix A. Each case study is broken down into one or more demonstrations (see the column 'Demonstrate' in Table 8). Then for each demonstration, we put a checkmark (√) if the demonstration was successful, otherwise it is marked x. To get a clearer understanding, Table 9 summarizes the comparison with the previous studies.

We compared the results of the present study with previous similar studies (Epure et al., 2015; Ferreira et al., 2017; Honkisz et al., 2018; Tangkawarow et al., 2021) to find out the advantages of our method (see Table 8 and Table 9). Table 9 summarizes the previous studies, showing the input constraints of textual requirements in the ten case studies. We tested the ability of the different methods to handle simple sentences, compound sentences, sequence clauses, the conjunction 'and', and the conjunction 'or', which all studies stated they could address. Meanwhile, the four previous studies cannot handle several constraints (intransitive sentences, keyword verb for datastore and data objects, and adverb manner). As for the other constraints, they were as presented in Table 9.

4.4.2. Comparison with the manual process by expert

To evaluate the proposed method's results, we measured how close its results were to the results of manual processing carried out by a human expert. Because the BPMN diagrams created by experts can produce multiple representations of an object, different human experts can produce different representations. However, we ascertained that the expert in this study was proficient in BPMN modelling and English. We limited our testing to ten case studies from the cooperative application used in the previous section. Table 10 shows the results of the numbers of BPMN diagram elements (activities, intermediate events, gateways, datastores/-data objects, and pools/lanes) obtained by the Proposed Method (PM), the Manual Expert (ME), and the Difference between both (D).

$$MRE_i = \frac{Difference_i}{ManualExpert_i} \cdot 100\% \quad (1)$$

$$MMRE = \frac{1}{N}\sum_{1}^{N} MRE_i \quad (2)$$

Table 9
Summary of comparison test results with similar previous studies.

No	Demonstrate	The proposed method	Honkisz et al. (2018)	Ferreira et al. (2017)	Epure et al. (2015)	Sonbol and Ghneim (2022)
1	Simple sentence	√	√	√	√	√
2	Compound sentence	√	√	√	√	√
3	Complex sentence with if-then-else	√	√	√	√	√
4	Sequential clauses	√	√	√	√	√
5	The conjunction 'and'	√	√	√	√	√
6	The conjunction 'and' on object	√	√	√	√	√
7	The disjunction 'or'	√	√	√	√	√
8	Sequential verbs	√	×	√	√	×
9	Passive voice	√	×	√	√	×
10	Complex sentence with adjective clause	√	×	×	×	√
11	Compound-complex sentence	√	×	×	×	√
12	Looping process	√	×	×	×	√
13	Keyword verb for datastore	√	×	×	×	×
14	Intransitive clause	√	×	×	×	×
15	Keyword verb for data object	√	×	×	×	×
16	Adverb of manners	√	×	×	×	×
∑Supported items		16	7	9	9	10

Table 10
Results of the evaluation of the proposed method with a manual process by expert.

No of Req	Activities			Intermediate events			Gateways			Datastores/objects			Pools/lanes			∑			MRE (%)
	PM	ME	D	PM	ME	D	PM	ME	D	PM	ME	D	PM	ME	D	PM	ME	D	
1	2	2	0	0	0	0	0	0	0	1	1	0	3	3	0	6	6	0	0.00
2	3	3	0	0	0	0	0	0	0	0	2	2	3	3	0	6	8	2	25.00
3	5	5	0	0	0	0	0	0	0	2	2	0	3	3	0	10	11	0	0.00
4	5	4	1	1	1	0	2	2	0	1	1	0	3	3	0	12	11	1	9.09
5	6	5	1	0	0	0	2	2	0	2	2	0	3	3	0	13	12	1	8.33
6	4	4	0	1	1	0	2	2	0	1	3	2	3	3	0	11	13	2	15.38
7	4	5	1	0	0	0	0	0	0	3	3	0	3	3	0	10	11	1	9.09
8	8	8	0	0	0	0	0	0	0	1	1	0	3	3	0	12	12	0	0.00
9	12	12	0	0	0	0	2	2	0	2	3	1	4	4	0	20	21	1	4.76
10	8	8	0	2	2	0	5	5	0	3	3	0	4	4	0	22	22	0	0.00
															∑	122	126	8	
															MMRE				7.17

Abbreviations: Req = Requirements, Dev = Deviation, PM = Proposed Method, ME = Manual by Expert, MRE = Magnitude of Relative Error, MMRE = Mean of Magnitude of Relative Error (MMRE)

Abbreviations: Req = Requirements, Dev = Deviation, PM = Proposed Method, ME = Manual by Expert, MRE = Magnitude of Relative Error, MMRE = Mean of Magnitude of Relative Error (MMRE).

In this study, the validation metric used was accuracy, which was determined by measuring the error rate of the method or model used. Practitioners apply the Mean Magnitude of Relative Error (MMRE) in software estimation to determine the error rate (Nassif et al., 2011). First, the Magnitude of Relative Error (MRE) was determined for each case study using Equation (1) and then the MMRE was calculated from all observations using Equation (2). We got an MMRE of 7.17 %, which means the accuracy of the proposed method for the ten case studies was 92.83 %. Compared with the level of accuracy of the study conducted by Ferreira et al. (2017), which obtained an accuracy of 91.92 %, this study performed slightly better. Meanwhile, the proposed method's accuracy was higher than that of the method developed by Sonbol and Ghneim (2022), which was 81 %.

5. Conclusion, limitations, and future works

We have proposed a new method for generating BPMN diagrams from textual requirements input. The resulting BPMN diagram is intended for further processing to measure the functional software size. This is an interesting challenge because textual requirements are easier to obtain at the beginning of the software development project. This means that project managers, or anyone else interested, can get a functional software size estimation earlier. Thus, the project manager can earlier make an estimate of the resources needed to execute the project.

The proposed method has two stages: 1) analyzing the textual requirements using natural language processing and 2) generating the BPMN diagram. In the first phase, the analysis of the textual requirements is done to extract the fact types. Previously, the textual requirement inputs were split into separate sentences, then tokenized, and POS tagged using Stanza (version 1.3.0) to identify tokens such as nouns, verbs, prepositions, determiners, etc. Then dependency parsing is done using Stanza to get the relations between the words in the sentence. Finally, the fact types are extracted by applying the rules set created in this study. The fact types obtained are used as input for the second phase, i.e., generating the BPMN diagram. This phase begins with mapping the fact types onto elements of the BPMN diagram, followed by the formation of a spreadsheet-based description as a bridge for the generation of the BPMN diagram.

The proposed method was applied to ten case studies of textual requirements in a cooperative application. We compared the results of this study with those of previous similar studies to see the advantages of this study over the previous studies. The proposed method can handle complex sentences, compound-complex sentences, transitive sentences, input/output flow in/out datastore or data object, and adverbs of manner contained in the textual input. Compared with the BPMN diagram generated manually by an expert, we obtained an accuracy rate of 92.83 %, which was higher than the accuracy achieved in the previous studies.

However, there are some limitations to this study, namely 1) as mentioned by van der Aa et al. (2018) that the rules have not been able to distinguish the semantic/pragmatic level, for example, 'the production department' in a sentence, and 'a member of the production department' in other sentence are actually referring to the same actor in the process, but a system consider them two different entities. This study has not implemented it, so further research is needed to develop an algorithm to overcome these weaknesses. 2) Sometimes, the conjunction has multiple meanings, indicating a sequential or parallel process. For example, the sentence 'the student opens the book and reads it' is a sentence that has the conjunction 'and', which means sequential, while the sentence 'the student is reading a book and listening to music' has the conjunction 'and' which means parallel. Also, this study has not addressed these weaknesses, so further studies require in-depth semantic analysis to overcome these weaknesses. 3) This study applies 10 case studies to explain the generation of textual requirements into the BPMN Diagram. For point 3), replication studies need to be carried out by adding case studies of the textual requirements of several software applications.

Declaration of Competing Interest

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

Appendix A

Table A1

Table A11
The ten textual requirements of case studies.

No	Requirement	Textual input	∑sent
1	Making a cash transaction	An officer submits evidence of a transaction. An accountant records the cash transaction to a cash database, including transaction type, date, proof number, value for money, and description.	2
2	Withdraw mandatory savings	Management wants to withdraw the mandatory savings by providing information about the members. An officer displays the mandatory savings based on the member ID, so the officer processes the withdrawal of the mandatory savings.	2
3	Resign a member	Management asks a member to be removed from the cooperative membership. An officer accepts the management's request. The officer reads information about members from the member database, updates the member's status to resign to the member database, then informs the management about the member's status.	3
4	Making a cash outflow report	Management orders a cash-out report which of the year and month is given by management, that an accountant filters out cash transactions from the cash database. If the query is not empty, the accountant prints a cash-out report, gives it to management; otherwise, there is no cash-out.	2
5	Display a member	A member needs to display detailed information about a member by providing a member ID. An officer accepts member requests. The officer reads a member's information from the member database. If the member exists, then the officer reads the work unit data from the member database, the officer provides the detailed member information to the member, else the officer confirms no information.	4
6	Making a bill report	A member needs a monthly billing statement by providing his member ID and billing month. An officer reads a member's billing data from the billing database based on member ID and month. If the monthly bill exists, the officer prints the member's bill and informs the member, otherwise the bill does not exist.	3
7	Changing a work unit	A member submits changes to the work unit by filling out a form, so an officer accepts member's requests. The officer reads member and the works unit information from the member and the works unit databases. The officer updates member information based on the member database.	3
8	Realizing a member loan	Management wants to realize member loans. An officer receives orders for member loan realization. The officer opens the member loan realization menu, sets the loan type, enters the member ID. The officer sets the instalment amount, principal instalment, and repayment period. The officer realizes member loans, saves loan realization information to the loan database.	5

Table A11 (*continued*)

No	Requirement	Textual input	∑sent
9	Making a new member registration	A prospective member applies a requirements document to become a cooperative member that an officer accepts the document. Once the document has been lightly checked by the officer, the officer sends the document to the management for further processing. Management fully reviews the document based on the cooperative master book. Management assigns a status to a document based on the review results, so management sends the document back to the officer. If the status is accepted, then the officer creates a new member, adds a new member to the member database, prints the cooperative card, otherwise the officer returns the requirements document to the prospective member to complete the requirements, then the prospective member receives the requirements document to be completed.	5
10	Selecting a member loan request	*1 A member submits a loan request by attaching a credit requirements document. The credit section accepts member's loan requests. The credit section checks the credit requirements document which if the credit requirements document is incomplete, so the credit section returns it to the member to complete the credit required document, go-to 1. The credit section checks the mandatory deposit from the deposit database and checks the current loan from the loan database. If the mandatory deposit has not been paid off or the current loan exists, then the submission status is rejected, else the submission status is okay. The credit section sends the submission document to the officer for further processing. The officer receives the document.	7

Arbitrations: BFT = Binary Fact Type, C = Characteristic, Sent = Sentence.

Appendix B

Figs. B1–B9

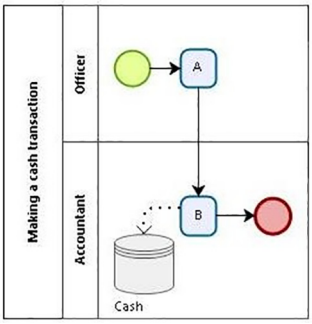

Fig. B1. The BPMN diagram for the case study 1.

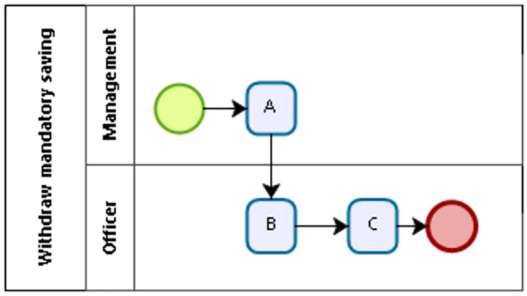

Fig. B2. The BPMN diagram for the case study 2.

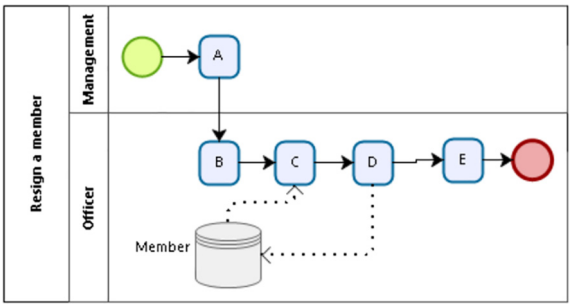

Fig. B3. The BPMN diagram for the case study 3.

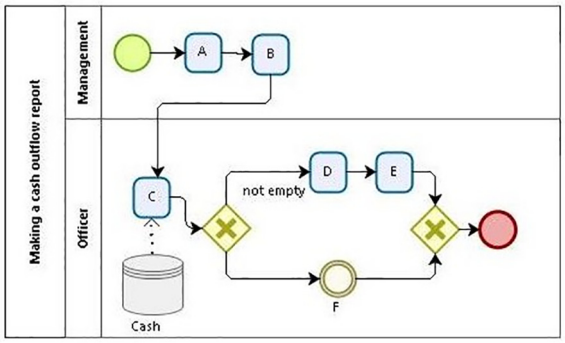

Fig. B4. The BPMN diagram for the case study 4.

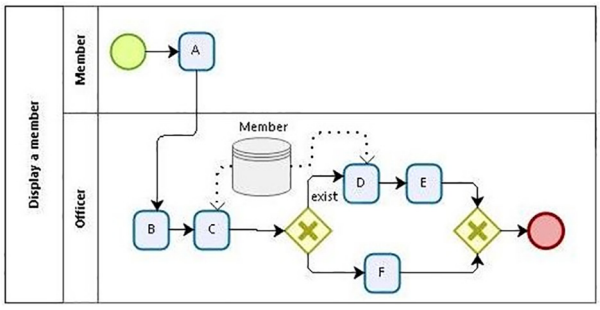

Fig. B5. The BPMN diagram for the case study 5.

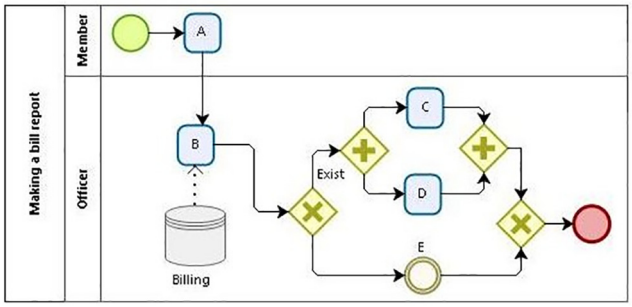

Fig. B6. The BPMN diagram for the case study 6.

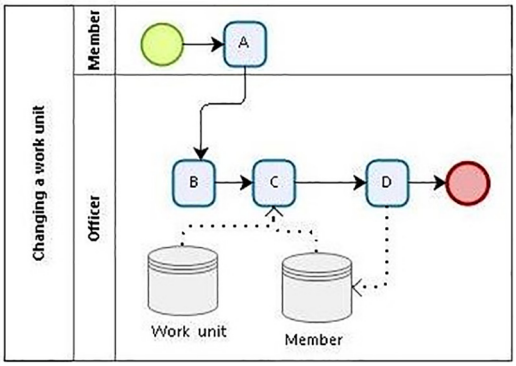

Fig. B7. The BPMN diagram for the case study 7.

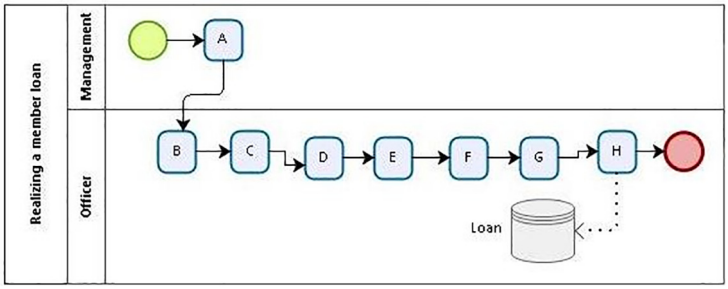

Fig. B8. The BPMN diagram for the case study 8.

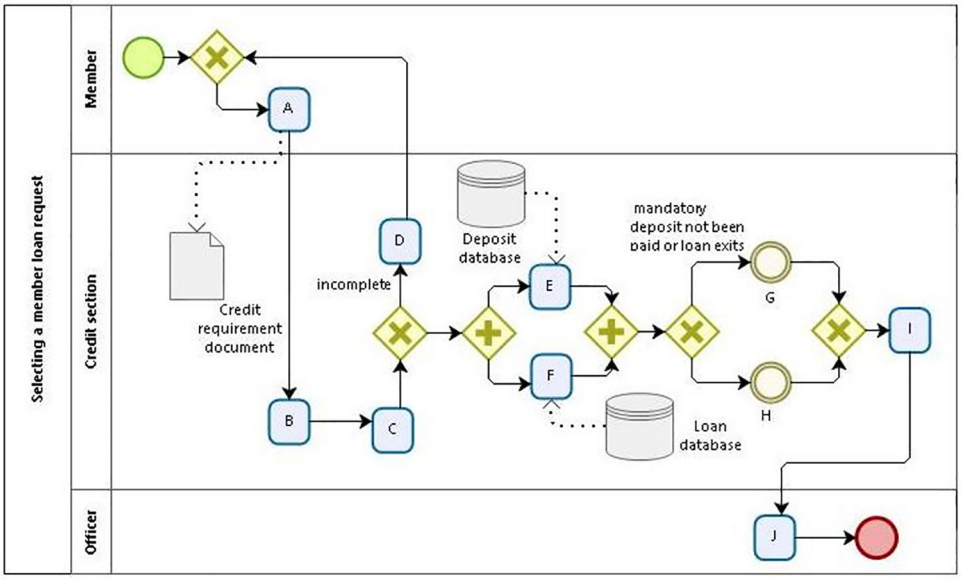

Fig. B9. The BPMN diagram for the case study 10.

References

Afreen, H., Bajwa, I.S., Bordbar, B., 2011. SBVR2UML: A Challenging Transformation. In: Proc. - 2011 9th Int. Conf. Front. Inf. Technol. FIT 2011, pp. 33–38. https://doi.org/10.1109/FIT.2011.14.

Ahmed, A., Prasad, B., 2016. Foundations of Software Engineering. CRC Press, Boca Raton, FL, USA.

Bajwa, I.S., Choudhary, M.A., 2012. From Natural Language Software Specifications to UML Class Models. Lect. Notes Bus. Inf. Process. vol. 102 LNBIP, 224–237. https://doi.org/10.1007/978-3-642-29958-2_15.

Bajwa, I.S., Lee, M.G., Bordbar, B., 2011. SBVR Business Rules Generation from Natural Language Specification. In: AAAI Spring Symp. - Tech. Rep., pp. 2–8.

Bajwa, I.S., Lee, M., Bordbar, B., 2012. Translating Natural Language Constraints to OCL. J. King Saud Univ. - Comput Inf. Sci. 24 (2), 117–128. https://doi.org/10.1016/j.jksuci.2011.12.003.

Bajwa, I.S., Sarwar, N., Naeem, M.A., 2016. Generating Express Data Models from SBVR. Proc. Pakistan Acad. Sci. Part A 53 (4A), 381–389.

Bajwa, I.S., 2012, "A Natural Language Processing Approach to Generate SBVR and OCL," The University of Birmingham.

Bonais, M., Nguyen, K., Pardede, E., Rahayu, W., 2016, "Automated Generation of Structural Design Models from SBVR Specification," vol. 11, pp. 51–87, doi: 10.3233/AO-160162.

Btoush, S.E., Hammad, M.M., 2015. Generating ER Diagrams from Requirement Specifications Based on Natural Language Processing. Int. J. Database Theory Appl. 8 (2), 61–70. https://doi.org/10.14257/ijdta.2015.8.2.07.

Calisaya, E.S., 2016. Analysis of Natural Language Scenarios. Universidad Nacional de San Agustin.
Cosmic, C., 2019, "The cosmic functional size measurement method version 4.0.2: introduction to the COSMIC method of measuring software."
Dalpiaz, F., Ferrari, A., Franch, X., Palomares, C., 2018. Natural language processing for requirements engineering: the best is yet to come. IEEE Softw. 35 (5), 115–119. https://doi.org/10.1109/MS.2018.3571242.
Delicado, L., Sànchez-Ferreres, J., Carmona, J., Padró, L., 2017, "NLP4BPM - Natural language processing tools for business process management," in: BPM Demo and Industrial Track 2017 Proceedings, 2017, vol. 1920, [Online]. Available: https://upcommons.upc.edu/handle/2117/121215.
Elizabeth O'Brien, "Sentence Structure," Grammar revolution. https://www.english-grammar-revolution.com/sentence-structure.html.
Epure, E.V., Martin-Rodilla, P., Hug, C., Deneckere, R., Salinesi, C., 2015, "Automatic Process Model Discovery from Textual Methodologies: An Archaeology Case Study," in: Proc. - Int. Conf. Res. Challenges Inf. Sci., vol. 2015-June, no. June, pp. 19–30, doi: 10.1109/RCIS.2015.7128860.
Essebaa, I., Chantit, S., 2018, "Tool Support to Automate Transformations from SBVR to UML Use Case Diagram," in: ENASE 2018 - Proc. 13th Int. Conf. Eval. Nov. Approaches to Softw. Eng., vol. 2018-March, no. Lim, pp. 525–532, doi: 10.5220/0006817705250532.
Renato C'esar Borges Ferreira, Lucin'eia Heloisa Thom, Marcelo Fantinato, 2017, "A Semi-Automatic Approach to Identify Business Process Elements in Natural Language Texts," in: Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017), vol. 3, no. Iceis, pp. 250–261, doi: 10.5220/0006305902500261.
Friedrich, F., Mendling, J., Puhlmann, F., 2011, "Process Model Generation from Natural Language Text," in: International Conference on Advanced Information Systems Engineering, pp. 482–496.
Gonçalves, J.C.de A.R., Santoro, F.M., Baião, F.A., 2011. Let Me Tell You A Story - On How to Build Process Models. J. Univers. Comput. Sci. 17 (2), 276–295.
Honkisz, K., Kluza, K., Wiśniewski, P., 2018, "A Concept for Generating Business Process Models from Natural Language Description," in: International Conference on Knowledge Science, Engineering and Management, vol. 11061, doi: 10.1007/978-3-319-99365-2_8.
Iqbal, U., Bajwa, I.S., 2016. Generating UML Activity Diagram from SBVR Rules. In: 2016 6th Int. Conf. Innov. Comput. Technol. INTECH 2016, pp. 216–219. https://doi.org/10.1109/INTECH.2016.7845094.
Khlif, W., Haoues, M., Sellami, A., Ben-Abdallah, H., 2017, "Analyzing functional changes in BPMN models using COSMIC," in: ICSOFT 2017 - Proc. 12th Int. Conf. Softw. Technol., no. January, pp. 265–274, doi: 10.5220/0006418902650274.
Kluza, K., Wisniewski, P., 2016, "Spreadsheet-Based Business Process Modeling," in: Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, FedCSIS 2016, vol. 8, pp. 1355–1358, doi: 10.15439/2016F376.
Marín, B., Quinteros, J., Portales, U.D., 2014, "A COSMIC Measurement Procedure for BPMN Diagrams," in: SEKE, pp. 408–411.
Mendling, J., Leopold, H., Thom, L.H., van der Aa, H., 2019, "Natural language processing with process models (NLP4RE report paper)," CEUR Workshop Proc., vol. 2376.
Mendling, J., Leopold, H., Pittke, F., 2014. 25 Challenges of Semantic Process Modeling [Online].Available: Int. J. Inf. Syst. Softw. Eng. Big Co. 1 (1), 78–94 http://www.uajournals.com/ijisebc/journal/1/6.pdf.
Monsalve, C., Abran, A., April, A., 2011. Measuring Software Functional Size from Business Process Models. Int. J. Soft. Eng. Knowl. Eng. 21 (03), 311–338. https://doi.org/10.1142/S0218194011005359.
Monsalve, C., April, A., Abran, A., 2012, "On the expressiveness of business process modeling notations for software requirements elicitation," in: IECON Proc. (Industrial Electron. Conf., pp. 3132–3137, doi: 10.1109/IECON.2012.6389398.
Nassif, A.B., Capretz, L.F., Ho, D., 2011, "Estimating software effort based on use case point model using sugeno fuzzy inference system," in: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, pp. 393–398, doi: 10.1109/ICTAI.2011.64.
Nguyen, V., Deeds-Rubin, S., Tan, T., Boehm, B., 2007, "A SLOC counting standard," in: Center for Systems and Software Engineering, University of Southern California.
Ochodek, M., 2016. Functional size approximation based on use-case names. Inf. Softw. Technol. 80, 73–88. https://doi.org/10.1016/j.infsof.2016.08.007.
OMG, "Semantics of Business Vocabulary and Business Rules Version 1.5," no. October, 2019.
Przybyłek, A., 2014, "A business-oriented approach to Requirements Elicitation," in: ENASE 2014 - Proc. 9th Int. Conf. Eval. Nov. Approaches to Softw. Eng., pp. 152–163, doi: 10.5220/0004887701520163.
Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D., 2020, "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages," arXiv Prepr. arXiv2003.07082, doi: 10.18653/v1/2020.acl-demos.14.
Quishpi, L., Carmona, J., Padró, L., 2020, "Extracting annotations from textual descriptions of processes," in: International Conference on Business Process Management, vol. 12168 LNCS, pp. 184–201, doi: 10.1007/978-3-030-58666-9_11.
Ramzan, S., Bajwa, I.S., Ul Haq, I., Naeem, M.A., 2014, "A Model Transformation from NL to SBVR," in: 2014 9th Int. Conf. Digit. Inf. Manag. ICDIM 2014, no. September, pp. 220–225, doi: 10.1109/ICDIM.2014.6991430.
Sawant, K.P., Roy, S., Sripathi, S., Plesse, F., Sajeev, A.S.M., 2014, "Deriving Requirements Model from Textual Use Cases," in: 36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings, pp. 235–244, doi: 10.1145/2591062.2591193.
Sonbol, R., Ghneim, N., 2022, "A Machine Translation like Approach to Generate Business Process Model from Textual Description," pp. 0–26, doi: 10.21203/rs.3.rs-1242866/v1.
Tangkawarow, I., Sarno, R., Siahaan, D., 2021. Modeling Business Rule Parallelism by Introducing Inclusive and Complex Gateways in Semantics of Business Vocabulary and Rules. Int. J. Intell. Eng. Syst. 14 (1), 281–295.
van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L., 2018, "Challenges and opportunities of applying natural language processing in business process management," in: COLING 2018 - 27th International Conference on Computational Linguistics, Proceedings, 2018, pp. 2791–2801, [Online]. Available: https://upcommons.upc.edu/handle/2117/121682.
Vega-Márquez, O.L., Chavarriaga, J., Linares-Vásquez, M., Sánchez, M., 2019. Requirements comprehension using BPMN: an empirical study. Empir. Stud. Dev. Exec. Bus. Process., 85–111 https://doi.org/10.1007/978-3-030-17666-2_5.
Yaqin, M., Sarno, R., Rochimah, S., 2020. Measuring Scalable Business Process Model Complexity Based on Basic Control Structure. Int. J. Intell. Eng. Syst. 13 (6), 52–65.