From Natural Language Text to Visual Models: A survey of Issues and Approaches

Cristina-Claudia OSMAN, Paula-Georgiana ZĂLHAN
Business Informatics Research Center,
Faculty of Economics and Business Administration,
Babeş-Bolyai University, Cluj-Napoca, Romania
cristina.osman@econ.ubbcluj.ro, paula.zalhan@econ.ubbcluj.ro

Over the last 20 years, research groups focused on automating the process of extracting valuable information from Natural Language text in order to discover data and process models. In this context, several tools and approaches have been proposed. The overall objective of this survey is to examine existing literature works that transform textual specifications into visual models. This paper aims to give a comprehensive account of the existing tools meant to discover data and process models from natural language text. Our analysis focuses on approaches of these tools in the model extraction process and highlight issues of each proposed approach. In the case of object oriented software modelling of data models extraction we analyze the degree of automation, efficiency and completeness of the transformation process. Regarding process models extraction, the study is not limited only to business process discovery, but it also provides case studies from several fields such as medical or archaeological. Even if not all the tools developed are clearly depicting a Natural Language Processing technique, a review of each approach is presented.
Keywords: *Natural Language Processing, Data Model, Business Process Model*

1 Introduction

Organizations focus on automate their processes in order to improve efficiency, reduce costs, and/or reduce human beings' errors in an easy and rapid manner. Business process management (BPM) methods provide a solution to this issue. In this context, information systems like CRM, ERP, SCM, etc. have known an increasing demand. The main problem consists of the length of the business process specifications. If new regulations appear, these specifications must be adapted. Manually extraction of visual models is time consuming. During time, a series of solutions were proposed. The literature shows a crowd of approaches that extracts data models [1], [2], [3], and process models [4], [5] from Natural Language (NL) text. This paper aims to analyze the Natural Language Processing (NLP) techniques and tools used in order to provide different types of visual representations.

On the last few years, approaches based on NLP have been developed in order to automate process conversion from NL text. NLP plays an important role in NL text analysis as NLP tries to understand speech and text as humans beings would do. Colloquialism, abbreviations or typos make this task a challenging one. NLP has the origins in 1950s when Alan Turing proposed the Turing test [6], by introducing the imitation game. Since then, the literature shows a plethora of NLP tools [7], [8], [9], [10] using several machine learning techniques, our focus being on those applied on data models and process models discovery from NL text, starting with the first language parser [11] to the actual ones like NLTk [7], [8], ANTLR[1], etc.

Linguistic analysis is closely tied to NLP. Liddy [12] highlights 7 levels of linguistic analysis: a) *Phonetic* or *Phonological level*: how words are pronounced, b) *Morphological level*: prefixes, suffixes and roots analysis, c) *Lexical level*: word level analysis including lexical meaning and Part-Of-Speech (POS)

[1] ANTLR, http://www.antlr.org/

DOI: 10.12948/issn14531305/20.4.2016.01

analysis, d) *Syntactic level*: grammatical analysis of words in a sentence, e) *Semantic level*: determining the possible meanings of sentences, f) *Discourse level*: interpreting structure and meaning for texts larger than a sentence, g) *Pragmatic level*: understanding the purpose of a language.

Some of the problems approached by NLP are: POS tagging, parsing, Named Entity Recognition (NER), chunking, Semantic Role Labeling (SRL). Anaphora resolution [13] refers to the interpretation of the link between the anaphor and its antecedents.

The remainder of the paper is organized as follows: Section 2 briefly outlines the NLP domain, describing the levels of linguistic analysis and the main NLP approaches of analyzing NL requirements. Section 3 focuses on data and process models extraction from NL text. This section makes an introduction to Object Oriented Analysis and Business Process Modeling and analyses the existing tools that discover data and process models from text. Subsequently, Section 4 summarizes the results of this work and the conclusions are drawn in Section 5.

2 NLP

A detailed review on NLP is given in [14] and in [15]. Jones [14] divides the history of NLP into four phases: the first starts at the beginning of the 1940s and lasts to the late 1960s, the second begins from the end of 60s and lasts to the end of 70s, the third is represented by late 80s, where the fourth phase starts in the late of 90s. Next, we will detail each phase as they were defined in [14] and [15]. First phase treated machine translation issues, while the second focused on artificial intelligence. The third phase can be called grammatico-logical phase, which is followed by the lexical phase. A fifth phase is proposed in [16] where formal theories and statistical data are combined. Since this study was published first in 1994 and then it was re-organized in 2001 we can add the sixth phase: from 2000 until present where NLP techniques are combined in order to contribute to visual models extraction.

Software requirements are usually written in NL which is asymmetric and irregular [17]. The major challenge in software design is the ability to analyze textual requirements outlined by the clients in order to extract valuable information used as input for next step's transformation in the process of data modelling. In recent years, several studies have proposed the use of linguistic instruments to help this requirements analysis [19], because of two main reasons:

a) the progress made in NLP which is an area of research and development that explores how computers can be used to analyze and represent NL texts at different levels of linguistic analysis for the purpose of achieving knowledge on how human beings understand and use language [18]
b) the need to provide the developers of software systems with support in the early phases of requirements definition and conceptual modelling [19]

2.1 Linguistic Analysis

In past years, numerous research projects focused on exploring how NLP can be used to carry out a linguistic analysis of requirements documents in order to produce conceptual models of them. The levels of linguistic analysis are presented in the following sections.

2.1.1 The Lexical Level of Analysis

Lexical analysis, also called token generation, is the process of converting a sequence of characters into a sequence of tokens [20]. It is composed of the following processing steps: tokenization, sentence splitting and POS tagging [21]. Tokenization is the first step in lexical analysis and it is used to identify words and numbers in sentences. Sentence splitting identifies sentence boundaries of a given text. POS-tagging enables NLP systems to interpret the meaning of individual words. According to this step, each word from the NL text corresponds to a particular part of speech based on the context in which it occurs. The most probable POS tag is assigned to a word taking into consideration the relationship with adjacent and related words in the phrase, sentence or paragraph [22]. Hence, POS tagging identifies words as nouns, verbs, adjectives,

DOI: 10.12948/issn14531305/20.4.2016.01

etc. For example, a NNI tag signifies a singular noun, while VBB indicates the base form of a verb. Currently, the most promising tool used for lexical analysis is Stanford Parser [2].

2.1.2 The Morphological Level of Analysis
Morphological analysis deals with the componential nature of words. A word from the NL text is composed of morphemes. It is important to mention here that a morpheme is the smallest piece of word that carries a meaning [23]. A NLP system can understand the meaning of a word analyzing it into its constituent morphemes. For example, appending a suffix to a verb, moves the action of that verb in the past [24]. At this level of analysis, there are used some stemming algorithms to remove the morphological affixes (such as prefixes and suffixes) of each word from the NL text, in order to achieve the root form of the word.

2.1.3 The Syntactic Level of Analysis
The syntactic level focuses on analyzing the words in a sentence using a grammar in order to reveal the structural dependency relationships between words [24]. The grammar provides syntax rules about possible organization of words in sentences. The output of this level of linguistic processing is a parse tree that represents the syntactic structure of a given sentence.

2.1.4 The Semantic Level of Analysis
The semantic level of processing determines possible meanings of sentences examining the meaning of constituent words. This level includes semantic disambiguation of polysemous words identifying the appropriate meaning of a word looking at the rest of the sentence. Semantics recognizes that most of the words have more than one meaning, based on their dictionary and context meaning [23]. The semantic analysis focusses on the interactions among word-level meanings in the sentence in an analogous way to morphological disambiguation of words [24]. This level permits only one sense of the multiple senses of a word to be selected and included in the semantic representation of the sentence. For the semantic analysis, different dictionaries and knowledge bases are used. The most known example is the lexical database WordNet[3] which is used to disambiguate the lexical structures of the NL text and to find semantically similar terms.

2.1.5 The Discourse Level of Analysis
While syntax and semantic level deal with sentence-length units, the discourse level works with units of text longer than a sentence trying to understand the role of a piece of information that exists in a particular document [24]. This level of linguistic analysis examines the structure of a given NL text, making connections between component sentences in order to understand and represent meaning of the text as a whole.

2.1.6 The Pragmatic Level of Analysis
Pragmatic analysis is concerned with how the external world knowledge impacts the meaning of the NL text. This level of analysis depends on a body of knowledge that comes from outside the contents of the document in order to gain understanding about purpose and goal of the given text [23]. Pragmatic level of analysis eliminates ambiguities in requirements reinterpreting the text in order to find its actual meaning and the speaker's intention.

2.2 NLP Techniques
In NLP area, researchers have proposed several techniques for analyzing the NL requirements. These techniques are basically categorized into two leading approaches:

2.2.1 Rule Based Approach
Rule approach relies on hand-constructed rules acquired from linguistic experts [25] and encapsulates human knowledge to apply these rules to the NL text. There is the possibility to automatically learn these rules through the analysis of annotated corpora using machine learning methods. It is important to mention here that a corpus (plural "corpora") describes a set of documents that have been annotated by NLP experts with the correct values that need to be learned [26].

[2] Stanford Parser, http://nlp.stanford.edu.

[3] WordNet2.1, http://www.cogsci.princeton.eu/~wn/.

DOI: 10.12948/issn14531305/20.4.2016.01

There are some advantages to the rule-based approach. First, rules can be refined for accuracy by experts in order to detect spelling or grammar mistakes [27]. Second, is easy to incorporate domain knowledge into the linguistic knowledge and the linguistic knowledge acquired for one natural language processing system may be reused to build knowledge required for a similar task in another system [25]. However, the cost of this approach is high, since it requires great effort from human experts to analyze the large volume of data from textual documents of requirements.

2.2.2 Statistical Based Approach

Statistical-based approach requires large text corpora to develop statistical models in order to automatically learn the linguistic features identified in the textual data. This approach uses mathematical techniques, including stochastic, probabilistic and statistical methods to solve some of the problems that arise due to long sentences from the NL text [26].

These methods are data-driven and rely on quantitative methods to automatically discover relations between words from the sentences [28]. Some of the issues regarding long sentences are due to the fact that these sentences are highly ambiguous and the analysis using a grammar leads to numerous interpretations. Methods of disambiguation involve using Hidden Markov models (HMMs) where the NL text is assumed to be a Markov process with unobserved states. HMMs are especially known for their application in fields such as speech recognition, gesture recognition, POS tagging. For example, in POS tagging based on HMM, the hidden states represent the underlying part-of-speech which corresponds to the sequence of word from the NL text.

The use of statistical approach has proven to be advantageous for lower levels NLP tasks of text analysis process [30] but the main disadvantage of this approach is that it requires a large amount of data in order to achieve statistically significant results.

3 Visual Models from NL Text

In this section, the discussion will point to visual models derived from NL text. The literature on data-centric models extracted from text shows a variety of data models used (e.g. Entity Relationship Diagrams, UML activity diagrams, class diagrams, sequence diagrams, object diagrams). Since 2007, an increasing interest for process models extraction from text has been shown.

3.1 Data Models Extraction from NL Text

Late ‘80s brought an increasing development of data-centric information systems. Starting with basic types of data modelling like Entity Relationship Diagrams (ERDs) [2] to different types of UML diagrams[4]. ERDs underlie the design of relational databases by offering a static data model. Another basic approach, this time based on control-flow is depicted by UML Activity Diagram. The class diagram is the central item of object-oriented modelling. An interaction diagram that depicts how objects interact with each other is the sequence (event) diagram. The current state of a system is described by object diagrams.

3.2 Object Oriented Software Modelling

In order to cope up with the increasing demands of fast growing information technology, new tools and technologies have been proposed in software engineering methodologies. Several changes were made in conventional methods of software analysis and design phase of the software development process. These changes reflect the use of object oriented design (OOD) paradigm in order to capture essential and relevant software requirements for constructing a software system [30]. OOD based software modeling is also called Component Added Software Engineering (CASE) [31] and encourages the use of Unified Modelling Language (UML) for modelling the user requirements visual the multiple dimensions and levels of details, docu-

[4] Documents Associated with Unified Modeling Language™ (UML®) Version 2.5, http://www.omg.org/spec/UML/2.5/PDF/

DOI: 10.12948/issn14531305/20.4.2016.01

ment software assets and accommodate incremental development and re-development of software [17].
In a conventional OOD software modelling approach, the system analyst first has to spend a lot of time understanding the user requirements and then, based on the requirements analysis made, orthodox CASE tools are used to draw the UML diagrams [30]. Object Oriented Analysis (OOA) applies the OOD paradigm to model software systems by defining classes, objects and relationships between them [32].

3.2.1 NLP Based UML CASE Tools

Nowadays projects stress the automatic extracting of OO concepts to generate static and dynamic system views from domain-specific NL descriptions [32]. Several NLP based CASE tools that utilize different levels, or combinations of levels of linguistic analysis have been designed in order to transform the NL specifications into OO models. Even though these NLP systems follow a OOA to model the software systems, none of the following tools are able to extract the complete information i.e. classes, objects and their respective, attributes, methods and associations. Some of the most known NLP based UML CASE tools that were proposed by researchers, are presented in chronological order.

3.2.1.1 LOLITA

Large-scale Object-based Linguistic Interactor Translator Analyzer (LOLITA) is an NLP system proposed by [33] in 1996. This system is able to automatically generate an object model from NL text. Hence, this tool extracts objects from every noun of the textual requirements and attempts to find relationships amongst these objects. LOLITA is built on a large scale Semantic Network (SN) which is a semantic graph that contains a large number of objects and event nodes used to bridge the gap between object diagrams and requirements [34]. Indeed, this approach considers nouns as objects and use links to find relationships between objects but it cannot distinguish differences between classes, attributes and objects. As a result, LOLITA is limited to extracting objects from NL text because it cannot identify classes.

3.2.1.2 RECORD

The *Requirements Collection, Reuse, and Documentation* (RECORD) system proposed by Börstler in [35] provides a semi-automatic process of generating object models from requirements. In this system, stakeholders play an important role to ensure that the right requirements are acquired using form-based user interface. The central idea of this approach is to match the stakeholder requirements, expressed in use cases like forms with object models. First, use cases are analyzed in order to extract objects, relationships between objects, operations and attributes. Then, the extracted information is used to classify the use case. This classification is utilized to query the repository to find object models with matching classifications. Finally, the results of the matching and linking process are reviewed and adjusted. This system relies on human interventions to link and edit use cases and object models in order to handle the conflicting matches resulted.

3.2.1.3 D-H

Delisle et al. [36] in their project *DIPETT-HAIKU* (D-H) present robust computational linguistic tools developed for knowledge extraction from NL requirements description. Syntactic analysis of NL text is performed by Domain-Independent Parser of English Technical Texts (DIPPET) and semantic analysis is performed by a separate module called HAIKU. During OO analysis of NL text, D-H is able to identify candidate objects, linguistically differentiating between Subjects (S) and Objects (O), and processes, Verbs (V), using the syntactic S-V-O sentence structure. Moreover, they found that candidate attributes could be identified in the noun modifier of compound nouns, e.g. reserved is the value of an attribute of "reserved book" [32].

3.2.1.4 LIDA

Overmyer and Rambow in 2001 [37] proposed a methodology and a prototype tool

DOI: 10.12948/issn14531305/20.4.2016.01

called *Linguistic Assistant for Domain Analysis* (LIDA) which provide linguistic assistance to construct UML class diagram from NL descriptions. This semi-automatic tool is used as a beginning point to facilitate the work of requirements analysts, and it requires considerable user intervention [38]. First, the analyst imports the documents to be analyzed and the LIDA system identifies the POS of words from document text. Then, the analyst works on the noun list marking relevant candidate classes and iteratively removing those classes that do not qualify as classes or that are candidate attributes instead. When candidate classes have been identified, the analyst moves to the adjectives list to identify candidate attributes of the classes. Finally, the analyst moves to the verb list to identify candidate methods and roles. After this identification process, the analyst uses LIDA Modeler to graphically associate attributes, methods and roles with the appropriate classes [39].

3.2.1.5 GOOAL

The prototype tool *Graphic Object Oriented Analysis Laboratory* (GOOAL) [40] supports automatic OO modeling and produces static and dynamic models from NL description of user requirements taking decisions sentence by sentence. The underlying methodology of this system includes role posets and semi-natural language (4W). First, the NL text is automatically translated to 4W language and then, the produced sentences are analyzed with role posets to produce static model views. Finally, the 4W sentences are used to generate dynamic views of the problem.

3.2.1.6 CM-Builder

Class Model Builder (CM-builder) [41] is an NLP-based CASE tool presented by Harmain and Gaizauskasin in 2003 which performs domain independent OO analysis. CM-Builder uses robust NLP techniques to analyze textual requirements and then construct an integrated discourse model, represented in a Semantic Network (SN). This SN is then used to automatically construct an initial UML class model by converting nouns into classes and verbs into relationships. This class model represents the object classes extracted from the text and the relationships among objects of these classes. This CASE tool was restricted to capture candidate class models because there is no appropriate mechanism for capturing candidate objects from NL text requirements [32]. Nonetheless, the CM-builder has a limitation in its linguistic analysis due to the ambiguity, fuzziness, and redundancy of NL [38].

3.2.1.7 UMGAR

In 2009, Deeptimahanti and Babar presented *UML Generator from Analysis of Requirements* (UMGAR) [42] which is a semi-automatic tool that assists developers in generating UML models like Use-case diagram, Design class model, and Collaboration diagram from NL requirements. This tool generates these types of diagrams following an OOA approach based on a combination of the Rational Unified Process (RUP) [43] and ICONIX process [44]. The Noun-Phrase technique of RUP [45] helps a requirement analyst to identify all possible objects from a given requirements document and generate analysis class model by attaching attributes and methods with the associated object [42]. UMGAR also provides a generic XMI parser to generate XMI files for visualizing the generated object models in any UML modeling tool. The UMGAR tool has been developed using three efficient NLP technologies:

- *Stanford Parser*-to generate parse tree and extract concepts like actors, use cases, classes, methods, attributes, and associations;
- *WordNet2.1*-to perform morphological analysis;
- *JavaRAP*[5]-to replace all the possible pronouns with its correct noun form.

UMGAR can be used for large requirement document but this tool requires human interaction, for example, to eliminate irrelevant classes and to identify aggregation/composition relationships among objects [38].

[5] JavaRAP, http://www.comp.nus.edu.sg/~qiul/NLPTools/JavaRAP.html.

DOI: 10.12948/issn14531305/20.4.2016.01

3.2.1.8 UMLG

Unified Modeling Language Generator (UMLG) [30] follows a rule-based approach to automatically analyze the NL text and then generate OO modeling on the basis of the valuable extracted information from textual requirements. This system also provides the facility of converting the object-oriented modeling information in multiple languages such as Java, C#.NET or VB.net. First, the NL text is analyzed using a rule based algorithm order to extract classes, objects and their respective, attributes, methods and associations. Then, UML diagrams such as Class diagram, Activity diagram, Sequence diagram, Use Case diagram are drawn using previously extracted information.

3.2.1.9 DC-Builder

Diagram Class Builder (DC-Builder) [34] is an automated tool using NLP techniques and domain ontologies in the analysis of users' requirements to facilitate the extraction of the class diagram. First, the requirements descriptions are analyzed using the GATE framework which has an information extraction (IE) system called A Nearly-New Information Extraction System (ANNIE) for NL processing of the textual requirements. Then, sets of heuristic rules are defined to extract UML concepts such as classes, attributes and associations from the NL text. As a result, it is obtained an initial XML file that needs to be refined. Due to the fact that this file can contain erroneous elements, domain ontologies are used to eliminate these irrelevant elements. Hence, a new XML file is obtained with improved quality of concepts identifications.

3.2.1.10 RAPID

More and Phalnikar have proposed a prototype tool referred to as *Requirement Analysis to Provide Instant Diagrams* (RAPID) in 2012 [46]. This tool facilitates the requirements analysis process, extracting core concepts among with their relationships and producing UML diagrams. RAPID tool extracts the UML concepts using various NLP technologies such as:

- *OpenNLP*[6] which provides lexical and syntactical parsers of the input sentence from textual requirements;
- *RAPID Stemming Algorithm* to identify the base form of words from NL text;
- *WordNet2.1* as semantic parser used to validate de semantic correctness of the sentences generated at the syntactic analysis.

In order to improve the UML concepts identification, the RAPID tool uses domain ontologies. Then, the refined UML concepts are used in the process of class extraction where different heuristic rules are applied to extract the class diagram. These heuristic rules refer to class, attribute and relationship identification rules. Finally, the extracted class diagram is refined using domain ontology. The limitation of RAPID tool is that each sentence in the requirements document has to satisfy a specific structure defined by RAPID system. Otherwise, the user is asked to change the requirement sentence structure [38].

3.2.1.11 ABCD

Karaa et al. proposed in 2015 an *Automatic builder of class diagram* (ABCD) [38] which is an automated tool implemented in Visual Basic.Net that is able to generate UML class diagrams from user requirements expressed in NL. The lexical and syntactical processing of NL requirements relies on Stanford NLP toolkit. The concepts related to UML class diagram (such as aggregation, composition, association multiplicity and generalization) are extracted using a pattern-matching NLP technique and then are saved into an XML Metadata Interchange (XMI) format. Then, the XMI file is imported with a CASE tool such as ArgoUML[7] to build the corresponding UML diagrams. This system lacks advanced mechanisms to deal with redundant information problem and confuses the concepts of association and method identification [38].

[6] OpenNLP: http://opennlp.sourceforge.net/

[7] ArgoUml: http://argouml.softonic.fr/

DOI: 10.12948/issn14531305/20.4.2016.01

3.3 Business Process Modeling

A business process is a network of coordinated coherent activities that interact to produce a business outcome[8]. Business Process Modelling can be seen as an extension of Workflow Management (WfM) approach by adding the diagnosis phase [47]. Since 2002, when the Sarbanes-Oxley Act[9] was enacted to protect investors from fraudulent accounting activities by corporations, companies already using a Business Process Management System (BPMS) must ensure the consistency and transparency of financial data. That was really challenging for BPMS vendors as the aim was to automate as much manual activities as possible, from all business processes. BPMSs must show the control-flow (the execution order of activities) of financial processes in order to provide visibility into bottlenecks of the entire process. Moreover, audit feature must be embedded into BPMSs. Security is also treated by Sarbanes-Oxley Act as BPMSs must assure multiple user groups. There are several types of notations used in process modeling, such as Event-driven Process Chain (EPC) [48], Petri Nets [49], Business Process Model and Notation (BPMN), etc. BPMN is one of the most used OMG standards for business process modeling. The main categories of BPMN models are: flow objects (activities and gateways: OR, XOR and AND), connection objects (links between BPMN objects), swimlanes (resources), and artefacts (e.g. data objects). They depict control-flow perspective of the process, but also data-flow and resources involved.

3.3.1 Process Model Discovery from Text

We divided our study regarding process models discovery from text into three categories: business, archaeological and medical. Most approaches that discover process models from text are in the business field, but there are also approaches validated using case-studies from fields reminded above. The literature shows an increasing development of semi-automated tools, but from 2011 automated tools have been developed.

3.3.1.1 Business

Wang et al. [50], [51] put the basis of a Policy-Driven Process Mapping (PDPM) methodology that extracts process models from business policy documents. In order to develop the entire process model, firstly, control flow is identified, then data flow constraints are analyzed, followed by routing rules. This is a semi-automated approach as for process model elements, knowledge carriers are needed. PDPM process starts with the draft of the process map based on Task View followed by adding the Data View. Subsequently, structural and domain constraints are applied. Lastly, the process map is reviewed and additional elements may be added if it is necessary. No NLP techniques are reminded in this paper, but a computational procedure was defined. More details about the NLP techniques used are developed in [52] when the research intersects process mining [53] field. In terms of text mining kernel-based methods [54] such as tree kernel method (syntactic structure) are applied. Tools like GATE[10], Ling-Pipe[11], CRF++[12] are used in order to solve Name-Entity Recognition and text chunking issues. Stanford POS Tagger [55] is used for POS tags. As final result the authors propose an UML activity diagram. Being the first approach on process models discovery from business policies, there are some drawbacks: parallelism is ignored and only XOR behaviors can be identified.

One of the first examples of process models extraction from natural language text is presented in [56] by using computational linguistics and NLP techniques:

- *Stanford Parser* [55] is used for syntax parsing;
- *FrameNet* [57] for semantic analysis;
- *WordNet* as lexical database.

Anaphora resolution [58] helps on the identification of the concepts which are references

[8] http://www.omg.org/oceb/defbusinessprocess.htm
[9] https://www.sec.gov/about/laws/soa2002.pdf
[10] GATE, https://gate.ac.uk/
[11] Ling-Pipe, http://alias-i.com/lingpipe/
[12] CRF++, https://taku910.github.io/crfpp/

DOI: 10.12948/issn14531305/20.4.2016.01

using pronouns and certain articles. World model [59] is used for extraction of flow objects, swimlanes, artefacts and connecting objects. Furthermore, this model serves as base for business process model extraction. First step of the procedure involves the nodes creation, followed by the building of sequence flows. Subsequently, dummy elements are removed. Next step assumes the construction of start and end nodes and meta activities are processed. As an optional step, black box pools and data objects may be added. Moreover, as a final point, the layout model is added. The proposed method generates suitable models in a proportion of 77% based on 47 text-model pairs.

In [60], [61] is put the basis on the process mining from group stories (TellStories) and then, [62] proposes a method called Story Mining for process model extraction from business people stories. Firstly, the groups of tellers are selected and they should tell stories related to the various situations in their day-to-day activities (process instances). Then, text examination follows. The techniques used for tokenization, morphological and lexical analysis, syntactic analysis, domain analysis are not specifically named. After text mining, a business process is built and, finally, models are reviewed by participants. On the other hand, on the paper from 2010 [62], the text mining techniques used are detailed, specifically algorithms from Bigua library [63], respectively from NLTk framework [8], [7] for syntactic analysis, e.g. Shallow Parsing [64]. Likewise, all these algorithms are encapsulated into a scientific workflow management system (SWfMS).

Table 1. NLP techniques used in business process models discovery [62]

Phase	**NLP libraries**
Tokenization	Bigua: RSLP Stemmer and NLTkTokenizer
Morphological and Lexical Analysis	NLTKnMacMorphoCorpus [65] and NLTkNGramTagger
Syntactic Analysis	NLTkRegExpParser
Domain Analysis	CREWS scenario metamodel [66]

After text analysis, two files are generated: a) a structured text file containing the log of the extraction process and b) a business process proto-model generated from a) using BPMN. As a final point, the business process model is built based on the proto-model and the knowledge carriers' remarks and comments.

The research conducted by [67] proposes a framework that generates process proto-models from enterprise repository, meant to help the business analysts. The entire process has two phases, namely text to model, respectively model to model. For the first step, the NLP framework used is NLTk (for verb phrases, verbs, temporal connectives [8], [7], meanwhile, the second phase treats the conversion of Strategic Rationale models [68] to BPMN proto-models, respectively the UML interaction diagram converted into business process models. Then, both models are analyzed by analysts in order to detect possible inconsistencies.

IBM [69] developed a semi-automated, for on-line analysis of natural language (NL). First, text is pre-processed with a part-of-speech tagger and a shallow parser (based on which a Finite State Transducer is used), then words are annotated with dictionary concepts, which classify verbs using a domain ontology and, finally, an anaphora resolution algorithm and a context annotator are applied. The NLP tools used are developed by the authors and embedded into an Unstructured Information Management Architecture [70]. The result is a Use case description meta-model that can be easily converted into a BPMN process model. Schmidt et al. (2015) propose a solution for text mining applied on questionnaires and interviews (developed in Rapid Miner [71]) in

DOI: 10.12948/issn14531305/20.4.2016.01

order to provide a pre-check of quality of process models. The approach consists of five steps: text conversion, tokenization, case conversion, stop word removal and stemming. This method is based on words' frequency.
A recent study [72] introduces a web application as a prototype for a) process models extraction from controlled natural language and b) process model extraction through users' interactions[13]. For the first direction, the proposed algorithm uses sentence templates and basic workflow patterns are implemented (Sequence, Alternative, Parallel Split, Synchronization and Simple Merge).
The parser used for descriptions is ANTLR. It returns an Abstract Syntax Tree (AST) that subsequently is converted into a Petri Net that can be easily transformed into business process model. The second method based on users' descriptions proposes a new approach of Adaptive Case Management (ACM). First, preconditions and post conditions are examined and the corresponding workflow is generated. Moreover, this approach is similar to event logs and process mining [73] techniques can be applied.

3.3.1.2 Archaeological Field

One recent research [74] introduces an unsupervised technique called *TextProcessMiner* that generates a log from text and converts it in a process instance model. The procedure has two phases: a) Activity Miner based on verb semantics and b) Activity Relationship Miner based on rules. NLP tools used are: NLTk [7], [8], Stanford Parser and Tagger [9] and PyEnchant[14], Verbnet [75] and as lexical databases - Wordnet [10]. VerbNet and WordNet help on the Verb Knowledge Base creation which is subsequently used on activity log discovery. This activity log depicts only one instance and can easily be converted into an event log having XES format [76], [77]. In terms of validation an archaeological case study is analysed and the results indicate that TextProcessMiner discovers correct activities in a proportion of 88%.

3.3.1.3 Medical field

Medical field [78] reveals a new approach based on the identification of four concepts: activities, resources, actors, and control flows. *Computer-interpretable guidelines* (CIGs) [79] are used to define process fragment recognition. CIGs consist of two components [78]: a) static component such as activities, actors, data and resources and b) dynamic components such as control flows. *Unified Medical Language System* (UMLS) Metathesaurus[15] is used as lexical database and MetaMap to discover Metathesaurus concepts referred to in clinical documents. Syntactic features are extracting using Stanford Parser [9].

4 Discussions

The capability for model generation of each of the NLP based UML CASE tools systems previously discussed is illustrated in Table 2 and Table 3. Studies on automatically generating object models from NL text begin in 1996 when [33] proposed a NLP system called LOLITA. This system is limited in extracting object from NL text due to the fact that it is incapable of identifying classes. Also, this tool requires user's intervention [38]. In the same year, Börstler [35] proposed RECORD tool which is able to generate use case diagram from NL requirements but relies on user intervention in the object extraction process.
After two years, Delisle et al. [36] developed robust linguistic tools for knowledge extraction from NL requirements description. DIPETT tool is used for syntactic analysis; meanwhile HAIKU tool is used for semantic analysis of NL text.
The D-H system proposed in [36] is capable of automatic OO modeling but it can only create Object Diagram from user requirements [32]. Later, Overmyer and Rambow [37] proposed in 2001 a prototype tool called LIDA which provides a semi-automatic process of generating Class and Object Diagram [32]. Even though it is capable of identifying candidate classes and associations, LIDA requires considerable user intervention.

[13] http://bpm.caporale.eu
[14] PyEnchant, http://pythonhosted.org/pyenchant/
[15] Metathesaurus, https://www.ncbi.nlm.nih.gov/books/NBK9684/

DOI: 10.12948/issn14531305/20.4.2016.01

The prototype GOOAL tool proposed by Perez-Gonzalez in 2002 supports automatic OO modelling and it is able to produce static and dynamic models from NL requirements specifications. The underlying methodology consists of role posets and 4W. The main disadvantage of this tool is that Class and Sequence Diagram are the only diagrams types generated [32].

Table 2. The capability for model generation of UML CASE tools

Diagram type	**Tool**									
	RECORD	**D-H**	**LIDA**	**GOOAL**	**CM-Builder**	**UMGAR**	**UMLG**	**DC-Builder**	**RAPID**	**ABCD**
Use case Diagram	yes	no	no	no	no	yes	yes	no	no	no
Class Diagram	no	no	yes	yes	yes	yes	yes	yes	yes	yes
Object Diagram	no	yes	yes	no	no	no	no	no	no	no
Sequence Diagram	no	no	no	yes	no	no	yes	no	no	no
Collaboration Diagram	no	no	no	no	no	yes	no	no	no	no
Activity Diagram	no	no	no	no	no	no	yes	no	no	no

In 2003, Harmain and Gaizauskasin presented CM-Builder tool which can automatically construct a UML class diagram based on OO analysis of textual requirements. This tool provides a limited linguistic analysis and cannot capture candidate objects from NL text.

The semi-automatic UMGAR tool presented by Deeptimahanti and Babar in 2009 assist developers in creating UML diagrams like Usecase, Class and Collaboration diagram from NL requirements. This tool relies on three efficient technologies: Stanford Parser, WordNet2.1 and JavaRAP. The only drawback of this tool is that requires considerable user intervention to eliminate irrelevant classes and to identify aggregation/composition relationships among objects [38]. In the same year, Bajwa et al. presented the UMLG tool that is able to automatically analyze NL requirements and generate OO modeling on the basis of analyzed text. The approach presented in [30] highlights the NLP heuristics and domain ontology techniques used in the process of Class diagram, Activity diagram, Sequence diagram, Use Case generation. This tool is not based on user intervention but the input requirement document is not free NL text [38].

The DC-Builder [34] tool proposed by Herchi and Abdessalem in 2012 also used NLP techniques and domain ontology to automatically extract the UML Class Diagram from user requirements. The main disadvantage of this tool is that it identifies just some concepts of class diagram such as: classes, attributes, associations, aggregation and generalization [34], [38].

In the same year, More and Phalnikar [46] present the RAPID tool that is able to extract core concepts among with their relationships and produce UML diagrams using NLP heuristics and domain ontology. The RAPID uses the subsequent NLP technologies in order to extract these UML concepts from NL text: OpenNLP, Rapid Stemming Algorithm and

DOI: 10.12948/issn14531305/20.4.2016.01

WordNet2.1. The only drawback of this tool is that the sentences from NL requirements have to satisfy a specific structure [38]. A more recent approach in presented in [38] by Karaa et al. where a tool called ABCD is able to generate UML class diagram from NL text using NLP technologies such as Standford NLP toolkit. This tool is able to extract concepts related to UML class diagrams such as aggregation, composition, association multiplicity and generalization. The main disadvantage of this tool is the incapability of dealing with redundant information problem the confusion of confuses the concepts of association and method identification [38].

Table 3. Comparison of existing UML CASE tools

Year	Tool	Author	User's intervention	Limitations
1996	LOLITA	Mich [32]	Yes	Cannot identify classes
1996	RECORD	Börstler [34]	Yes	Considerable user intervention
1998	D-H	Delisle et al. [35]	Automatic tool	Can generate only Object Diagrams
2001	LIDA	Overmyer and Rambow [36]	Semi-automatic tool	Needs considerable user intervention
2002	GOOAL	Perez-Gonzalez [39]	Automatic tool	Can generate only Class and Sequence diagram
2003	CM-Builder	Harmain and Gaizauskasin [41]	Automatic tool	Cannot capture candidate objects from NL text
2009	UMGAR	Deeptimahanti and Babar [42]	Semi-automatic tool	Requires human interaction to eliminate irrelevant classes and to identify aggregation/ composition
2009	UMLG	Bajwa et al. [29]	Automatic tool	The input requirement document is not free NL text
2012	DC-Builder	Herchi and Abdessalem [33]	Automatic tool	Only some concepts are identified
2012	RAPID	More and Phalnikar [46]	Can ask a user to change a sentence	Each sentence in the requirements document has to satisfy a specific structure
2015	ABCD	Karaa et al. [37]	Automatic	Lacks advanced mechanisms to deal with redundant information problem and confuses the concepts of association and method identification

Regarding process modelling, studies from 2007 to 2010 propose semi-automated methods for text conversion into process models. First researches on text transformation to visual representation started in 2007 when Ghose

DOI: 10.12948/issn14531305/20.4.2016.01

et al. (2007) propose a Process Mapping approach that uses process proto-models. This approach is semi-automated as it requires domain knowledge. One year later, IBM came with another semi-automated approach having as result an UML Use-Case. They claim that a BPMN process model can easily be generated from it. Wang et al. [50], [51] started the research on process mapping and, then, give it another functionally [52] by link it to process mining. Then, de Goncalves et al. (2009) [61] bring a new concept called TellStory, subsequently transformed into StoryMining [62].

Starting from 2011, automated approaches have been developed. One of the most comprehensive researches on process models extraction from natural language text is [56]. The authors propose as final result a BPMN model. Then, two approaches from medical [78] having as result CIG fragments represented with BPMN, respectively archaeological [74] fields have been proposed. The only drawback of this approach [74] is that it generates and models only for one process instance. The most recent study [72] provides a web-based application for process models extraction from controlled natural language.

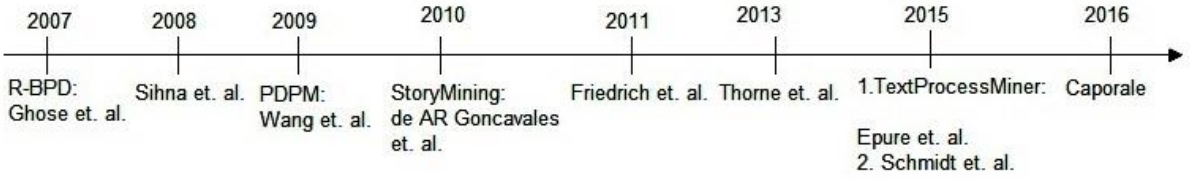

Fig. 1. Process discovery from NL text: Evolution

The majority of NLP libraries belong to NLTk framework [8] (e.g.: NLTKnMacMorphoCorpus, NLTkNGramTagger, NLTkRegExpParser).

5 Conclusions

The main concern of the paper was to present the current status of the existing tools that convert text to data and process models. In the OO analysis depicted for extracting UML core concepts NL descriptions, several NLP based CASE tools have been proposed. These tools utilize different levels or combinations of levels of linguistic analysis in order to transform the NL specifications into OO models. A review of approaches adopted by each UML CASE tool has been presented in chronological order. Some of these tools such as GOOAL,

UMLG or ABCD can automatically extract valuable information and generate data models from NL text. In contrast, there exist tools that require consistent human intervention in the process of UML diagrams generation. LOLITA, RECORD, LIDA or UMGAR tool are just a few NLP based systems that are based on user's intervention. Even though significant improvements have been made in the past years, none of these NLP based CASE tools are able to extract all the UML core concepts i.e. classes, objects and their respective, attributes, methods and associations.

Stanford Parser and the lexical database WordNet2.1 are the most used NLP techniques in the process of analyzing textual requirements, extracting OO core concepts and generating UML diagrams.

We also focused our attention to process modelling, specifically to its automation. Only one recent study, to our knowledge, has come up with the state-of-the-art of [80], but it depicts few approaches, and only concerning business process models. We have addressed not only to business field, but also related to the medical and archaeological field [69]. An analysis engine for dependable elicitation on natural language use case description and its application to industrial use cases. IBM Report, 2008] does not remind the libraries used in order to develop the tool, while [67], [74] are using algorithms from NLTk framework. Also, Stanford Parser is widely used in the analyzed studies [52], [62], [78], [74]. [56], [74] use lexical databases like WordNet and VerbNet. Moreover, Friedrich et al. (2011) [56] use

DOI: 10.12948/issn14531305/20.4.2016.01

Anaphora resolution and [74] employs Enchant as spelling checker. The only approach based on ANTLR library is [72].
From the research that has been carried out, it is possible to conclude that each tool employ different NLP libraries, the most used being the Stanford Parser that analyses the grammatical structure of sentences.

Acknowledgement
We acknowledge support from UEFISCDI under project PN-II-PT-PCCA-2013-4-1644.

References

[1] C.W. Bachman, "Data Structure Diagrams", *ACM Sigmis Database*, vol. 1, no. 2, pp. 4-10, July 1969.
[2] P.P.-S. Chen, "The entity-relationship model toward a unified view of data," *SIGIR Forum*, vol. 10, no. 3, pp. 9-36, 1975.
[3] D. Harel, "State charts: A visual formalism for complex systems", *Science of Computer Programming*, vol. 8, no. 3, pp. 231 -274, 1987.
[4] M. Chinosi and A. Trombetta, "BPMN: An introduction to the standard", *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124-134, 2012.
[5] R.M. Dijkman, M. Dumas, and C. Ouyang, "Semantics and analysis of business process models in BPMN", *Information and Software technology*, vol. 50, no. 12, pp. 1281-1294, 2008.
[6] A.M. Turing, "Computing machinery and intelligence", *Mind*, vol. 59, no. 236, pp. 433–460, 1950.
[7] E. Klein, "Computational semantics in the natural language toolkit", in *Proc. of the Australasian Language Technology Workshop*, Sydney, Australia, 2006, pp. 26–33.
[8] S. Bird, "Nltk: the natural language toolkit", in *Proc. of the COLING/ACL on Interactive presentation sessions*, Sydney, Australia, 2006, pp 69–72.
[9] D. Klein and C.D. Manning, "Accurate unlexicalized parsing", in *Proc. of the 41st Annual Meeting on Association for Computational Linguistics*, vol. 1, Sapporo, Japan, 2003, pp. 423-430.
[10] G.A. Miller, "Wordnet: a lexical database for English", *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
[11] T. Winograd, "Understanding natural language", *Cognitive psychology*, vol. 3, no. 1, pp. 1–191, 1972.
[12] E.D. Liddy, "Enhanced Text Retrieval Using Natural Language Processing", *Bulletin of the American Society for Information Science and Technology*, vol. 24, no. 4, pp. 14-16, 1998.
[13] A. Pirkola and K. Järvelin, "The effect of anaphor and ellipsis resolution on proximity searching in a text database", *Information processing & management*, vol. 32, no. 2, pp. 199-216, 1996.
[14] K. Spärck Jones, "Natural language processing: a historical review", *Current issues in computational linguistics: in honour of Don Walker*, pp. 3–16, 1994.
[15] K. Spärck Jones, "Natural language processing: a historical review", *University of Cambridge*, pp. 2–10, 2001.
[16] K. Spärck Jones, G.J.M. Gazdar, and R.M. Needham, "Computers, language and speech: Formal theories and statistical data", *Computational Linguistics*, vol. 26, no. 4, pp. 1225–1431, 2000.
[17] M. Mohanan and P. Samuel, "Open NLP based Refinement of Software Requirements", *International Journal of Computer Information Systems and Industrial Management* Applications, vol. 8, pp. 293–300, 2016.
[18] G.C. Chowdhury, "Natural Language Processing", *Annual Review of Information Science and Technology*, vol. 37, no. 7, pp. 51–89, 2003.
[19] L. Mich, M. Franch and P.N. Inverardi, "Market research for requirements analysis using linguistic tools", *Requirements Engineering*, vol. 9, no. 2, pp. 40-56, 2004.
[20] R.R. Young, *Effective Requirements Practices*, Addison-Wesley, Boston, 2001.
[21] T. Yue, L.C. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models", *Requirements Engineering*, vol. 16, no. 2, pp. 75-99, 2011.

DOI: 10.12948/issn14531305/20.4.2016.01

[22] T. Bures, P. Hnetynka, P. Kroha, and V. Simko, "Requirement Specifications Using Natural Languages", Technical Report D3S-TR-2012-05, December 2012.

[23] S. Feldman, "NLP Meets the Jabberwocky : Natural Language Processing Information Retrieval", *Online-Weston Then Wilton*, pp. 62-73, May 1999.

[24] E.D. Liddy, "Natural Language Processing", *Encyclopedia of library and information science (2nd edition)*, Marcel Decker Inc., New York, 2001.

[25] K. Shaalan, "Rule-based approach in Arabic natural language processing", *International Journal on Information and Communication Technologies (IJICT)*, vol. 3, no. 3, pp. 11-19, 2010.

[26] T. Bures, P. Hnetynka, P. Kroha, and V. Simko, *Requirement Specifications Using Natural Language*. Technical Report D3S-TR-2012-05. Czechoslovakia: Charles University in Prague, 2012.

[27] K. Crowston, X. Liu, and E.E. Allen, "Machine learning and rule-based automated coding of qualitative data", in *Proc. of the American Society for Information Science and Technology*, vol. 47, no. 1, 2010.

[28] F. Hogenboom, F. Frasincar, and U. Kaymak, "An Overview of Approaches to Extract Information from Natural Language Corpora", in *10th Dutch-Belgian Information Retrieval Workshop (DIR 2010)*, 2010, pp.69-70.

[29] F. Hogenboom, F. Frasincar, U. Kaymak, and F. De Jong. "An overview of event extraction from text", in *Workshop on Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE 2011) at Tenth International Semantic Web Conference (ISWC 2011)*, vol. 779, pp. 48-57, 2011.

[30] I.S. Bajwa, A. Samad, and S. Mumtaz, "Object oriented software modeling using NLP based knowledge extraction", *European Journal of Scientific Research*, vol. 35, no. 1, pp.22-33, 2009.

[31] G. Booch, I. Jacobson, and J. Rumbaugh, *The Unified Modeling Language: User Guide*. Addison Wesley, 1999.

[32] K. Li, R. G. Dewar, and R.J. Pooley, "Object-Oriented Analysis Using Natural Language Processing," in *Linguistic Analysis*, 2005.

[33] L. Mich, "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA," in *Natural Language Engineering*, vol. 2, no. 2, pp. 167-181, 1996.

[34] H. Herchi and W. B. Abdessalem, "From user requirements to UML class diagram", in *Proc. of International Conference on Computer Related Knowledge (ICCRK' 2012)*, Sousse, Tunisia, 2012.

[35] J. Börstler, "User-Centered Requirements Engineering in RECORD - An Overview", in *Proc. Nordic Workshop on Programming Environment Research (NWPER'96)*, Aalborg, Denmark, May 1996, pp. 149-156.

[36] S. Delisle, K. Barker and I. Biskri, "Object-Oriented Analysis: Getting Help from Robust Computational Linguistic Tools", in *Proc. of The Fourth International Conference on Applications of Natural Language to Information Systems (OCG Schriftenreihe 129)*, Klagenfurt, Austria, 1999, 167- 171.

[37] S. P. Overmyer, L. Benoit, and R. Owen, "Conceptual modeling through linguistic analysis using LIDA," in *Proc. of the 23rd International Conference on Software Engineering (ICSE)*, 2001, pp. 401–410.

[38] W.B. A. Karaa, Z. B. Azzouz, A. Singh, N. Dey, A. S. Ashour, H. B. Ghazala, "Automatic Builder of Class Diagram (ABCD): an Application of UML Generation From Functional Requirements", *Journal of Software Practice and Experience*, vol. 46, no.12, pp. 1443–1458, 2016.

[39] S.L.V. Overmyer and O. Rambow, "Conceptual Modeling through Linguistics Analysis Using LIDA", in *Proc. of the 23rd International Conference on Software engineering (ICSE'01)*, July 2001, pp. 401-410.

[40] H. G. Perez-Gonzalez, "Automatically Generating Object Models from Natural Language Analysis", in *Proc. of the 17th*

DOI: 10.12948/issn14531305/20.4.2016.01

annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, New York, USA, 2002, pp. 86 – 87.

[41] H. M. Harmain and R. Gaizauskas, "CM-Builder: A Natural Language-based CASE Tool", *Journal of Automated Software Engineering*, vol. 10, no. 2, pp. 157-181, 2003.

[42] D. K. Deeptimahanti and M. A. Babar, "An automated tool for generating UML models from natural language requirements", in *Proc. of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, 2009, pp. 680-682.

[43] P. Kruchten, *The Rational Unified Process An Introduction*, Second Edition, Addison Wesley, 2000.

[44] D. Rosenberg and K. Scott, *Use Case Driven Object Modeling With UML: A Practical Approach*, Addison-Wesley, 1999.

[45] R. Wirfs-Brock, B. Wilkerson, L. Wiener, *Designing Object-Oriented Software*, Prentice-Hall, 1990.

[46] P. More, R. Phalnikar, "Generating UML Diagrams from Natural Language Specifications", in *International Journal of Applied Information Systems*, vol. 1, no. 8, pp. 19-23, 2012.

[47] W. M. Aalst, *Process-Aware Information Systems: Design, Enactment, and Analysis.* Wiley Encyclopedia of Computer Science and Engineering, 2009.

[48] J. Mendling, "Event-driven process chains (epc)", In *Metrics for Process Models*, Springer Berlin Heidelberg, 2008, pp. 17-57.

[49] C. A.Petri, "Kommunikation mit automaten", 1962.

[50] H. Wang, L., Zhao, L., L. Zhao, and L. Zhang, "Policy-driven process mapping (PDPM): towards process design automation", in *ICIS 2006 Proceedings*, 2006, pp.12.

[51] H.J .Wang, J.L.Zhao, and L.J. Zhang, "Policy-Driven Process Mapping (PDPM): Discovering process models from business policies", in *Decision Support Systems*, vol. 48, no.1, pp. 267-281, 2009.

[52] J. Li, H.J. Wang, Z. Zhang, and J.L. Zhao,"A policy-based process mining framework: mining business policy texts for discovering process models", *Information Systems and E-Business Management*, vol 8, no. 2, pp.169-188, 2010.

[53] W. Van Der Aalst, A. Adriansyah, A.K.A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J.C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, and A. Burattin, "August. Process mining manifesto", in *Proc. of International Conference on Business Process Management*, 2011, pp. 169-194.

[54] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[55] D. Klein and C. D. Manning, "Accurate unlexicalized parsing", in *Proc. of the 41st Annual Meeting on Association for Computational Linguistics*, vol. 1, 2003, pp. 423–430.

[56] F. Friedrich, J. Mendling, and F. Puhlmann, "Process model generation from natural language text", in *Proc. of International Conference on Advanced Information Systems Engineering*, vol. 1, 2011, pp. 482–496.

[57] C.F. Baker, C. J. Fillmore, and J.B. Lowe, "The berkeley framenet project", in *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistic*, vol. 1, 1998, pp. 86-90.

[58] F. Friedrich, "Automated generation of business process models from natural language input," PhD diss., HUMBOLDT-UNIVERSITAT ZU BERLIN, 2010.

[59] L. Carlson and S. Nirenburg, *World modeling for NLP*. Pittsburgh, PA: Center for Machine Translation, Carnegie Mellon University, 1990.

[60] R. Perret, M.R. Borges, and F.M. Santoro, "Applying group storytelling in knowledge management", in *Proc. of International Conference on Collaboration and Technology*, 2004, pp. 34-41.

[61] J.C. de AR Goncalves, F.M. Santoro, and F.A. Baiao, "Business process mining from

DOI: 10.12948/issn14531305/20.4.2016.01

group stories", in *IEEE of The 13th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2009)*, pp. 161-166, 2009.

[62] J. C. de AR Gonçalves, F. M. Santoro, and F. A. Baião, "A case study on designing business processes based on collaborative and mining approaches", in *IEEE of 14th International Conference on Supported Cooperative Work in Design (CSCWD 2010)*, pp. 611–616, 2010.

[63] D. Oliveira, F. Baião, and M. Mattoso, "Miningflow: adding semantics to text mining workflows", in *First Poster Session of the Brazilian Symposium on Databases*, pp. 15–18, 2007.

[64] M. Osborne, "Shallow parsing as part-of-speech tagging", in *Proc. of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, vol. 7, 2000, pp. 145–147.

[65] S.M. Aluísio, G.M. Pinheiro, A.M. Manfrin, L.H. de Oliveira, , L.C Genoves Jr., and S.E. Tagnin, "The Lácio-Web: Corpora and Tools to Advance Brazilian Portuguese Language Investigations and Computational Linguistic Tools", in *LREC*, 2004.

[66] C.B. Achour, "Guiding Scenario Authoring", in the *8th European-Japanese Conference on Information Modelling and Knowledge Bases*, Vamala, Finland, 1998.

[67] A. Ghose, G. Koliadis, and A. Chueng, "Process discovery from model and text artefacts", in *IEEE Congress on Services*, 2007, pp. 167-174.

[68] E.S. Yu, "Models for supporting the redesign of organizational work", in *Proc. of conference on Organizational computing systems*, 1995, pp. 226-236.

[69] A. Sinha, A. Paradkar, P., Kumanan, and B. Boguraev, "An Analysis Engine for Dependable Elicitation on Natural Language Use Case Description and its Application to Industrial Use Cases", in *IBM Report*, 2008.

[70] D. Ferrucci, and A. Lally, "UIMA: an architectural approach to unstructured information processing in the corporate research environment", in *Natural Language Engineering*, vol. 10, no. 3-4, pp.327-348, 2004.

[71] G. Ertek, D. Tapucu, and I. Arin, *Text mining with rapidminer*. RapidMiner: Data Mining Use Cases and Business Analytics Applications, page 241, 2013.

[72] T. Caporale, "A tool for natural language oriented business process modelling", in *ZEUS 2016*, pp. 49, 2016.

[73] W. Van Der Aalst, A. Adriansyah, A.K.A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J.C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, and A. Burattin, "Process mining manifesto", in *International Conference on Business Process Management*, Springer Berlin Heidelberg, pp. 169-194, 2011.

[74] E.V. Epure, P. Martín-Rodilla, C. Hug, R. Deneckère, and C. Salinesi, "Automatic process model discovery from textual methodologies", in *IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, 2015, pp. 19-30.

[75] K. Kipper, B. Snyder, and M. Palmer, "Extending a Verb-lexicon Using a Semantically Annotated Corpus", in LREC, 2004.

[76] C.W. Günther, and H.M.W. Verbeek, *Xes-standard definition*, 2014.

[77] H.M.W Verbeek, J.C. Buijs, B.F. Van Dongen, and W.M Van Der Aalst, "Xes, xesame, and prom 6", in *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*, Springer Berlin Heidelberg, pp. 60-75, 2010.

[78] C. Thorne, E. Cardillo, C. Eccher, M. Montali, and D. Calvanese, "Process fragment recognition in clinical documents", in *Congress of the Italian Association for Artificial Intelligence*, Springer International Publishing, pp. 227-238, 2013.

[79] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, and S. Miksch, "Comparing computer-interpretable guideline models: a case-study approach", in *Journal of the American Medical Informatics Association*, vol. 10, no.1, pp.52-68, 2003.

DOI: 10.12948/issn14531305/20.4.2016.01

[80] M. Riefer, S. F. Ternis, and T. Thaler. "Mining process models from natural language text: A state-of-the-art analysis".

Cristina-Claudia OSMAN has graduated the Faculty of Economics and Business Administration, Babeş-Bolyai University, Cluj-Napoca in 2008. She holds a bachelor degree in Business Informatics, master degree in E-Business and a PhD degree in Cybernetics and Economical Statistics from 2014. Her current research interest includes Process Mining, Workflow Management and Text Mining.

Paula-Georgiana ZĂLHAN has graduated the Faculty of Economics and Business Administration, Babeş-Bolyai University, Cluj-Napoca in 2014. She holds a bachelor degree in Business Informatics and a master degree in Economic Decision Support Systems. She is currently a PhD student in the field of Cybernetics and Statistics. Her current research interest includes Natural Language Processing, Automatic Speech Recognition and Machine Learning.

DOI: 10.12948/issn14531305/20.4.2016.01