Automated Extraction of Conceptual Models from User Stories via NLP

Marcel Robeer, Garm Lucassen, Jan Martijn E.M. van der Werf, Fabiano Dalpiaz and Sjaak Brinkkemper
Department of Information and Computing Sciences
Utrecht University
Email: {m.j.robeer, g.lucassen, j.m.e.m.vanderwerf, f.dalpiaz, s.brinkkemper}@uu.nl

***Abstract*—Natural language (NL) is still the predominant notation that practitioners use to represent software requirements. Albeit easy to read, NL does not readily highlight key concepts and relationships such as dependencies and conflicts. This contrasts with the inherent capability of graphical conceptual models to visualize a given domain in a holistic fashion. In this paper, we propose to automatically derive conceptual models from a concise and widely adopted NL notation for requirements: user stories. Due to their simplicity, we hypothesize that our approach can improve on the low accuracy of previous works. We present an algorithm that combines state-of-the-art heuristics and that is implemented in our Visual Narrator tool. We evaluate our work on two case studies wherein we obtained promising precision and recall results (between 80% and 92%). The creators of the user stories perceived the generated models as a useful artifact to communicate and discuss the requirements, especially for team members who are not yet familiar with the project.**

***Index Terms*—User stories, conceptual modeling, NLP**

I. Introduction

The software industry commonly uses natural language (NL) notations to express software requirements [1]; a recent study shows that NL is employed by over 60% of the users [2]. With the increasing adoption of agile development practices such as Scrum, moreover, semi-structured yet NL notations such as user stories are gaining momentum [3], [4].

Despite the advantages of NL requirements in terms of ease of understanding and learning—we use NL daily for communicating with others!—, this requirements notation also suffer from drawbacks. The well-known problem of ambiguity (see [5] for an authoritative review) is a central issue. However, we focus here on another limitation: the difficulty of creating a holistic view of the requirements that highlights the key entities and relationships, and thereby supports identifying dependencies, redundancies, and inconsistencies.

We align with existing proposals that complement NL requirements with conceptual models in order to create this holistic view that NL alone fails to provide (e.g., [6]–[9]). However, we aim to go beyond the limitations of these inspiring tools, which either (i) lack automation thereby requiring human involvement to tag the text [7], [10], or (ii) have low accuracy, often due to the ambitious attempt to support arbitrarily complex requirements statements [8], [11].

Our recipe to overcome these limitations is as follows: (i) we choose the notation of *user stories*, which is highly popular among practitioners [3], [4] and that expresses concisely the essential elements of requirements; (ii) to minimize the need of human processing, we propose a *fully automated* software tool; and (iii) we strive to obtain *high accuracy*, and we do so by carefully choosing heuristics that help create a holistic view of the requirements, and by ignoring those that contribute with too fine-grained details (and are often less accurate).

Specifically, we make the following contributions:

- We combine natural language processing (NLP) heuristics into an algorithm that creates conceptual models from user stories;
- We present the Visual Narrator: a fully automated, open-source tool that implements our algorithm and generates conceptual models as OWL ontologies;
- We conduct a quantitative evaluation of the accuracy of our implementation in terms of recall and precision against two data sets from two case companies. The evaluation shows promising results with respect to a manual tagging of the data sets from the two case companies.
- We report on a qualitative evaluation where we interviewed the lead analysts from two case companies about the usefulness of our approach.

This paper is a brick of our research line on user stories. In previous work, we have proposed a conceptual model and an NLP-enabled tool for writing high-quality user stories [12] that obtained positive results in several case studies [13] and we have studied the perception and adoption of user stories in industry [4]. Here, we take high-quality user stories and use them to extract conceptual models.

The rest of the paper is structured as follows. Sec. II introduces baseline and related work. Sec. III reviews 23 heuristics from the literature and explains those that we choose for our work. Sec. IV presents the architecture and the main algorithm of the tool. Sec. V and Sec. VI report on the quantitative and qualitative evaluations. Sec. VII discusses our approach and presents future directions.

II. Background

We present our baseline in Sec. II-A, and discuss the relevant related work in Sec. II-B.

A. Baseline: conceptual anatomy of user stories

We build on previous work [13]–[15] and define a generic conceptual model of stories that dissects a single user story and defines its syntactic structure. We do so by assembling the conceptual model as a UML class diagram in Fig. 1. User stories follow a standard predefined format [14] to capture three distinct aspects of a requirement:

1) *Who* wants the functionality;
2) *What* functionality the end users or stakeholders want the system to provide; and
3) *Why* the end users and stakeholders need this functionality (optional).

These three aspects are captured in a simple textual template to form a running sentence. Although many different templates exist, 70% of practitioners use the template *"As a ⟨type of user⟩, I want ⟨some goal⟩ [so that ⟨some reason⟩]"* [15].

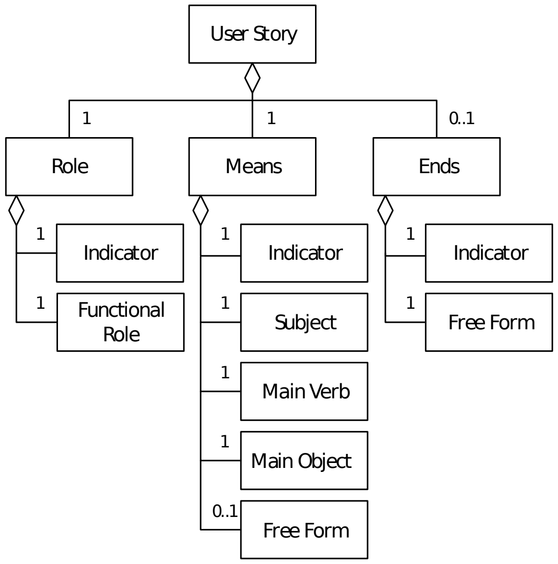

Fig. 1. Conceptual model defining the syntax of a user story

As per our previous work [13], the conceptual model distinguishes between the *role*, *means* and *ends* parts of a user story. Each of these parts features an *indicator* which delimits the three basic parts of a user story. Jointly, the indicators define the *template* of a user story.

The *role* part encompasses the role indicator and the *functional role*, describing a generalized type of person *who* interacts with the system. Throughout this paper, when no ambiguity exists, we use the term *role* to denote both the full part and the *functional role*. The *means* consists of a *subject*, a *main object* of the functionality, and a *main verb* describing the relationship between the subject and main object. The main object can be represented explicitly by the direct object or implicitly (e.g., 'I want to log in' actually refers to logging in the *system*). Note that we extend our previous work [13] that used the term *action verb*—which does not support state verbs like 'to see'—and replace it with the more general term *main verb*. The rest of the *means* can assume too many variations in reality; as such, we do not make any further distinction: words are captured by the *free form* class.

For example, consider the user story *"As a visitor, I want to purchase a ticket so that I can attend the conference"*. Here, 'visitor' is the functional role, 'I' is the subject, 'a ticket' the main object (a *direct* object), and 'purchase' is the main verb linking subject and object. There is no free form part for the means, and the indicators are 'As a', 'I want to', and 'so that'.

Although the ends has a similar structure to the means in this example, this is not always the case. In [13], we have identified three main purposes that the ends may take: (1) clarify the means, (2) reference another user story, or (3) introduce a qualitative requirement. These functions can be combined for a single user story. This semantic distinction between the types of ends goes beyond the scope of this paper, where we limit ourselves to the syntactic structure of the user stories.

B. Related work: extracting models from requirements

Extracting conceptual models from NL requirements is a long-standing research topic that is relevant in several domains. Already in 1989, Saeki et al. described a method where verbs and nouns are automatically extracted from NL, in order for a requirements engineer to derive a formal specification of the system [16]. One of the first tools that implement this idea is NL-OOPS [17]. The authors demonstrate the capabilities of NL-OOPS by generating a data model from a 250 word text. Since then, many tools have been proposed with diverse approaches and results. CM-builder [8] managed to extract *candidate* attributes, classes and relationships from a 220 word text with 73% recall and 66% precision. CIRCE [18] is a sophisticated tool that generates many different models including ERD, UML and DFD from NL requirements. Experimental application in three case studies indicated improvements in software model analysis and changing requirements. All these tools, however, require either human intervention or artificially restricted NL in order to generate complete and consistent models, prohibiting widespread adoption among practitioners. Recognizing this gap in a structured literature review, Yue et al. called for future approaches that fully automatically generate complete, consistent and correct UML models [19]. Their latest tool, aToucan, generates reasonably high quality class diagrams from use cases in comparison to diagrams created by experts, managing to consistently outperform fourth-year software engineering students in terms of completeness, consistency and redundancy. Moreover, its output constitutes initial models to be refined by experts [20]. Similarly, the tool presented in [11] outperforms novice human modelers in generating conceptual models from natural language requirements. Overall, their recall for identifying classes ranges from 85% to 100% depending on the case, while their precision is between 81% and 94%. The performance for relationships between classes, however, is less impressive: recall is in between 50% and 60%, while precision is in the 80%-100% range.

III. NLP Heuristics for User Story Analysis

To extract meaningful models from NL requirements, researchers have been using *heuristic* rules that identify concepts and relationships whenever the text matches certain patterns of the given language (usually English). The purpose of this section is to select NLP heuristics that can be effectively employed to derive conceptual models from user stories.

TABLE I
CLASSIFICATION OF HEURISTICS FOUND IN PREVIOUS LITERATURE FOR CONCEPTUAL MODEL GENERATION.

ID	Rule head (if)	Rule tail (then)	Source(s)
Concepts			
C1	Noun	Potential concept	[8], [21], [22]
C2	Common noun	Concept	[6], [9], [22], [23]
C3	Sentence subject	Concept	[11], [22]
C4	Compound noun	Take compound together to form concept	[11], [24]
C5	Gerund	Concept	[11], [23]
Non-hierarchical Relationships			
R1	Verb	Potential relationship	[11], [21]
R2	Transitive verb	Relationship	[8], [9], [11], [23]
R3	Verb (phrase) linking the subject and an object	Relationship	[8], [11], [22]
R4	Verb followed by preposition	Relationship including preposition	[6]
R5	Noun-noun compound	Non-hierarchical relationship between prefix and compound	[24]
Hierarchical Relationships			
H1	Verb 'to be'	Subjects are children of parent object	[9], [22]
H2	Head of noun-noun compound	IS-A relationship between compound and head	[24]
Attributes			
A1	Adjective	Attribute of noun phrase main	[8], [9], [11], [22], [23]
A2	Adverb modifying a verb	Relationship attribute	[9], [23]
A3	Possessive apostrophe	Concept attribute	[8], [11]
A4	Genitive case	Concept attribute	[6], [11], [22]
A5	Verb 'to have'	Concept attribute	[8], [11], [22], [25]
A6	Specific indicators (e.g. 'number', 'date', 'type', ...)	Concept attribute	[6]
A7	Object of numeric/algebraic operation	Concept attribute	[23]
Cardinality			
CA1	Singular noun (+ definite article)	Exactly 1	[8], [21], [22]
CA2	Indefinite article	Exactly 1	[8]
CA3	Part in the form "More than X"	$X..*$	[6], [8]
CA4	Indicators *many*, *each*, *all*, *every*, *some*, *any*	$??..*$	[6], [8]

Through our study of the literature on conceptual model generation, we have identified the 23 heuristics shown in Table I. This overview groups the heuristics by the part of a conceptual model they generate: concepts, non-hierarchical relationships, hierarchical relationships, attributes, and cardinality. The table presents as simple version of each rule as an implication from a condition (the *head*) to a consequence (the *tail*). As an example, the first concept heuristic should be read as **C1**: "If a word is a *noun*, then it is a *potential concept*."

For user stories, which are concise statements about the functionality of a system, not all heuristics are equally relevant. For example, user stories are not meant to include information about attributes or cardinality [15], thereby making these heuristic categories poorly relevant for our work. Other heuristics are still ignored by or too difficult for state-of-the-art part-of-speech taggers. For example, the de-facto standard Penn Treebank tags cannot distinguish between gerunds and present participle. This exclusion process results in 11 heuristics that are particularly relevant for generating conceptual models from user stories. In the following sub-sections, we explain the chosen heuristics and provide illustrative examples.

A. Concepts and Non-hierarchical relationships

The most basic heuristics in the literature specify that (1) nouns in a sentence denote a *concept*, and (2) verbs indicate a potential *relationship* [21], [26]. This prompts us to define the first concept and relationship heuristic as follows:

C1. *"Every noun is a potential concept."*

R1. *"Every verb is a potential relationship."*

Example A: Consider the user story "As a visitor, I want to create a new account." that comprises two nouns (*visitor* and *account*), and one verb (*create*) when we exclude role and means indicators. Rule C1 specifies to create two concepts visitor and account and rule R1 originates a relationship between these concepts named as the verb: create(visitor,account).

However, uncritically designating all nouns as concepts would result in a conceptual model full of superfluous concepts. Previous authors have employed the distinction between *proper nouns* and *common nouns* to generalize some of the identified concepts as more abstract instantiations [6], [9], [22], [23]. In general, common nouns are concepts and proper nouns are instances of these concepts that can be disregarded. *Transitive verbs* have a similar function, referencing an object in the sentence. These two phenomena lead to heuristics C2 and R2:

C2. *"A common noun indicates a concept."*

R2. *"A transitive verb indicates a relationship."*

To form relationships between concepts, a sentence should contain three elements: the subject, the object and the verb (phrase) linking the previous two. The *subject* is certainly essential: in an active sentence, for instance, the subject is the initiator—the so-called *agent*—of the main action performed in the sentence. Therefore, it has its own heuristics:

C3. *"The subject of a sentence is a concept."*

R3. *"The verb (phrase) linking the sentence subject and an object forms the relationship between these two."*

Example B: Let us consider the story "As John the manager, I want to design a website." The sentence comprises three common nouns, one proper noun and one transitive verb. The person *John*—a proper noun—can be generalized to his job description *manager* (C2). Therefore, John is an instance of concept manager. Note that this proper noun defines the concept John because heuristic C3 says that, no matter its type, the subject of a sentence leads to a concept. The transitive verb has direct object *website*, and subject *I* which refers to *John* (R2). As we do not know whether the ability to design a website applies to John or to managers in general, we create concept John (C3) with relationship design(John,website) (R3).

Concerning relationships, Omar et al. [6] distinguish between two types of verbs that indicate relationships: general transitive verbs and verbs followed by a preposition. These prepositions significantly change the meaning of a relationship, and are therefore captured in a separate heuristic:

R4. *"If a verb is followed by a preposition, then the preposition is included in the relationship name."*

Example C: For the user story "As a visitor, I want to search by category", we first identify the subject *I* (C3). The sentence has no direct object. The preposition *by* (R4) changes the meaning of the relationship from searching something to searching by something. Therefore, we obtain the relationship search_by(I,category). Since *I* refers to the functional role *visitor*, it results in the relationship search_by(visitor,category).

B. Compound Nouns

Compound nouns describe a concept using multiple words. Most often these are sequences of nouns or adjectives followed by one or more nouns. To accurately construct a conceptual model, we consider the whole compound noun as the concept. Compound nouns are known to have many inherent relationships, as there are many ways to combine them [27]. However, extracting this requires the synthesis of lexical, semantic and pragmatic information, which is a complex task [28] that can hardly lead to accurate results. Therefore, we limit ourselves to a simple heuristic proposed by Vela and Declerck [24] by considering compound nouns of length two only:

C4. *"Noun compounds are taken together to form a concept."*

R5. *"In noun-noun compounds there is a non-hierarchical relationship between the prefix and compound."*

Example D: For the compound noun "event organizer" we create a compound concept event_organizer (C4) and a "has" relationship has_organizer(event,event_organizer) (R5).

C. Hierarchical Relationships

The ontology generation domain pays special attention to generalization relationships, often referred to as *IS-A* relationships [9], [22]. Tenses of the verb *to be* typically indicate a hierarchical relationship between two concepts. The subject of the relationships on the left side of the verb is a specialization of the parent object on the right side of the verb *to be*:

H1. *"The verb 'to be' indicates a hierarchical relationship: the subject is taken as a specialization of the parent object."*

In addition, Vela and Declerck [24] note that the nouns in compounds have a generalization relationship. Compound-noun concepts are a form of a more abstract concept, e.g., database_administrator is a type of administrator. This is captured by the following heuristic:

H2. *"If there is a noun-noun compound present, the head of the compound is the parent of the compound concept."*

Example E: Consider the user story "As a visitor, I can change my account password." In this user story, heuristics C4, C5 and H2 apply on noun compound *account password*. First, we create compound concept account_password (C4), which is a type of password: IS-A(account_password,password) (H2). In addition, we create a 'has' relationship (R5) has_password(account,account_password).

IV. The Visual Narrator Tool

To enable the automated extraction of conceptual models from user stories, we developed the *Visual Narrator* tool on the basis of the 11 heuristics detailed in Sec. III. Visual Narrator takes a set of user stories as input and generates a conceptual model as output. It is built in Python and relies on the natural text processor *spaCy* (http://spacy.io), a recent proposal in NLP with excellent performance and that implements algorithms needing minimal to no tuning. Additionally, phrasal verb extraction is performed using an algorithm proposed in [29].

Our tool only accepts user stories that use the indicators as identified by Wautelet [14]: *As* / *As a*(*n*) for the role, *I want* (*to*) / *I can* / *I am able* / *I would like* for the means, and *so that* for the ends part. Syntactically invalid user stories are not processed; in order to sanitize these stories, analysts should pre-process them using tools such as AQUSA [13].

In addition to generating a holistic conceptual model, Visual Narrator can also generate separate models per role to help

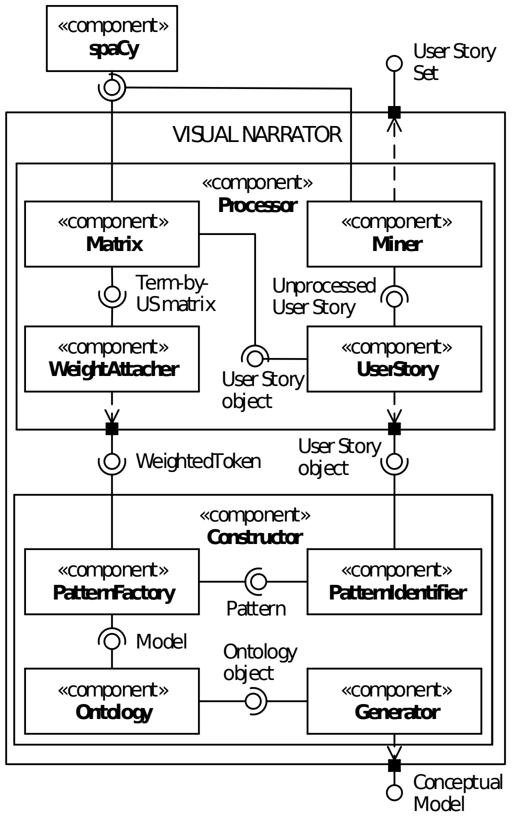

Fig. 2. Component Diagram of the Visual Narrator tool

analysts focus on an individual role. Furthermore, analysts have the option to fine-tune the sensitivity of the tool: (i) *weights* for each type of concept (role, main object, compound, etc.) can be specified to determine their relative importance, and (ii) a *threshold* can be expressed to exclude from the generated models the least frequent entities by computing a ranking based on frequency and concept weight.

A. Architecture

The main architecture of Visual Narrator is shown in Fig. 2 and depicts two main components: (1) the Processor analyzes and parses user stories according to the syntactic model for user stories (Fig. 1), while (2) the Constructor creates the actual conceptual model starting from the parsed stories.

First, the Processor analyzes the user story set. The Miner component uses spaCy to parse each user story into tokens, which hold the term itself, its part-of-speech tag and relationships with other tokens. These tokens are stored in the UserStory component and used to infer concepts and relationships, and to determine token weights.

Next, the Matrix component removes stop words from the collection of tokens and then attaches a weight to each term, based on the frequency and on the weights that were specified as input parameters. This step results in a *Term-by-User-Story matrix* containing a weight for each term in the individual user stories. The summation of the weights for a given term is added to each token, resulting in a set of WeightedTokens.

The Constructor then generates the conceptual model by further processing the WeightedTokens. This starts in the PatternIdentifier component, which applies the heuristics to identify all patterns in the user story. The PatternFactory component creates an internal conceptual Model based on the patterns and stores it in the Ontology component. Parts of the Ontology are linked to the user story they originated from. Note that the PatternFactory filters out all concepts and relationships with a weight smaller than a user-specified threshold. Finally, two different Generators output an ontological representation of the Ontology object as an OWL 2 ontology and as a Prolog program.

Our implementation adapts some of the heuristics, either due to the specific NL toolkit spaCy or to further optimize the results. We modify C2 by including both common and proper nouns, because proper nouns often refer to domain-specific concepts such as the name of a software product or a library. To improve accuracy, we replace some pronouns with the noun they refer to when no ambiguity exists (see Sec. IV-B). Effectively, this means Visual Narrator assigns every noun to a new or existing concept. We did not implement H1 because the correct application of the heuristics requires a deep understanding of the semantics of the "to be" relationship.

B. Extraction Algorithm

To extract a conceptual model from user stories, Visual Narrator implements the procedure DERIVECM presented in Algorithm 1. The procedure takes as input a set of raw user stories *S* and empty sets of concepts *C* and relationships *R*, and populates *C* and *R* while parsing the stories and applying the heuristics defined in the previous sections.

The procedure starts in line 2 by defining the valid indicators *ind* for splitting the user stories into role, means, end. The cycle of lines 3–14 excludes syntactically incorrect user stories and creates the set of concepts, including hierarchical ones. Every story (r, m, e) is initially split using the indicators (line 4); if this operation fails, the story is discarded and the loop continues to the next story (line 5). If a story is identified, it is added to the set of syntactically valid user stories *S'*.

Every part of a syntactically valid user story (role, means, end) is parsed (line 8). Then, the heuristics to identify nouns (C2) and the subject of the sentence (C3) are executed (lines 9–10); all identified nouns are added to the set of concepts C. Lines 11–14 process compound nouns: the compound is added to C according to heuristic C4, a specialization relationship is created linking the sub-class to the super-class (H2), and a non-hierarchical "has" relationship is created from the prefix of the compound to the compound itself (R5).

Lines 15–29 iterate over the set S' of syntactically correct user stories with the intent of identifying relationships where concepts in C are linked through associations created by processing the verbs in the user stories. Lines 16–17 modify the means and the end by setting their subject correctly: the subject of the means (the "I" of the indicator "I want to") is replaced by the subject of the role r; a similar processing applies to ends whose raw subject is "I" (e.g., "so that I…").

Algorithm 1 Pseudo-code of DERIVECM that builds a conceptual model by mining stories and applying the heuristics

```
procedure DERIVECM(Stories S, Concepts C, Rels R)
    ind = ({As a, …},{I want, I can, …},{So that})
    for each $s \in S$ do
        (r,m,e) = split-by-indicators(s,ind)
        if (r,m,e) == null then continue
        else $S' = S' \cup (r, m, e)$
            for each $p \in \{r, m, e\}$ do
                $pt$ = create-parse-tree($p$)
                $C = C \cup$ find-nouns($pt$) [C2]
                $C = C \cup$ subject-of($pt$) [C3]
                for each cn∈comp-nouns(pt) do
                    $C = C \cup \{cn\}$ [C4]
                    $R = R \cup$ IS-A($cn$, head($cn$)) [H2]
                    $R = R \cup$ has(prefix($cn$), $cn$) [R5]
    for each $(r, m, e) \in S'$ do
        replace-subject(m,r)
        if subject-of(e) == "I" then replace-subject(e,r)
        for each $p \in \{m, e\}$ do
            $subj$ = find-subject($p$)
            if $subj$ == null then continue
            $v$ = find-main-verb($p$)
            $obj$ = find-direct-object(p) [R2]
            if $obj$ == null then
                $obj$ = find-non-direct-object($p$) [R3,R4]
            if $obj$ == null then continue
            if $subj, obj \in C$ then
                $R = R \cup v(subj, obj)$
            else if subj $\in C \wedge p == m$ then
                $R = R \cup v(subj,$ system$)$
```

Both means and end are processed in lines 18–29. Subject, main verb and direct object (R2) are identified in lines 19, 21, 22, respectively. If no subject is identified, the algorithm continues to the next element: we do not look for verbs when the subject is unclear or nonexistent. If a direct object is not found, an indirect object is searched for applying heuristics R3 and R4 (lines 23–24). If no object is found, the cycle continues to the next element (line 25). If both subject and object are in C, a relationship with the name of the verb is created between them (lines 26–27). In case only the subject is in C and we are analyzing the means, a relationship is created from the subject to a special concept called system. For example, the story "As a user, I want to login" would result in a relationship login(user,system). The same rule applies only to the means because this part refers to a desired functionality, while the structure of the end is less rigid [13].

V. QUANTITATIVE EVALUATION: ACCURACY

We evaluate the accuracy and feasibility of our approach by applying Visual Narrator to two data sets obtained from two case study companies. The data sets and their evaluation documents are available online[1]. In Sec. V-A, we determine the accuracy of our implementation of the heuristics (Algorithm 1) by comparing the results of Visual Narrator to a manual labeling of the data sets done by one of the authors. In Sec. V-B, we discuss the limitations of our approach in terms of important concepts and relationships that are not recognized due to either NLP limitations, our algorithm, or unanticipatable structure of the user stories.

[1]http://www.staff.science.uu.nl/~lucas001/conc_modeling_user_stories.zip

A. Heuristics Accuracy

To determine the accuracy of our implemented heuristics, one author—different from the developer—analyzed and tagged the concepts and relationships in the user stories of the two case studies by manually applying Algorithm 1. This tagging was then compared against the outputs of the tool. We argue for this to be a more objective evaluation than comparing with the subjective output consisting of all human recognized concepts and relationships in a text.

Our evaluation has two aims:

- To determine *quantitatively* to what extent the NLP toolkit that was employed fails to deliver accurate results due to the difficulty of correctly tagging sentences.
- To analyze *qualitatively* the limitations of our implementation in terms of important information that is not recognized correctly.

We determine true positive, false positive and false negative elements (concepts and relationships) as follows:

- *True positive*: the element is identified both by the tool and by the manual analysis.
- *False positive*: the element is identified by the tool but not by the manual analysis.
- *False negative*: the element is not identified by the tool while it was listed in the manual analysis.

Note that multiple heuristics may apply to a given text chunk. For example, three heuristics apply to a compound: C4, H2, and R5. Thus, if the tool misses a compound, it actually generates three false negatives; on the other hand, if the tool and the manual analysis match, three true positives are generated. Moreover, the incorrect identification of a relationship (e.g., see(visitor,content) instead of see(visitor,display_name)) results in both a false positive (the wrong relationship) and a false negative (the missed out relationship).

We report on the accuracy in two ways: (i) on individual user stories, by aggregating the number of true positives, false negatives and false positives for concepts and relationships; and (ii) on the obtained conceptual model, by comparing the manually created one and the generated one.

Making this distinction is important when some concepts appear multiple times: consider, for example, a compound noun (C4, H2, R5) that appears in 20 different user stories in a data set. This would imply 20 false negative concepts and 40 false negative relationships (20 times H2 and 20 times R5). On the other hand, the resulting conceptual model would only miss one concept and two relationships.

Tables II–V report the results using the same format. They have three macro-columns: concepts, relationships, and overall (concepts+relationships). Each macro-column has three sub-columns to denote true positives (TP), false positives (FP) and false negatives (FN). The rows represent the number of instances (the number of identified and missed out concepts and relationships), the percentile splitting of TP, FP, and FN, the precision, and the recall.

1) WebCompany: This is a young Dutch company that creates tailor-made web business applications. The team consists of nine employees who iteratively develop applications in weekly Scrum sprints. WebCompany supplied 98 user stories covering the development of an entire web application focused on interactive story telling that was created in 2014.

73 of these 98 user stories were syntactically correct, usable and relevant for conceptual model generation [13]. Part of the generated conceptual model is shown in Fig. 3.

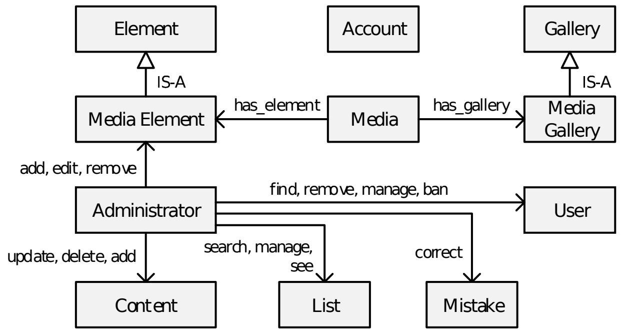

Fig. 3. Partial model for WebCompany generated with Visual Narrator

The accuracy results of the individual user story analysis shown in Table II are quite positive. The overall precision and recall are 91% and 92%, respectively. The accuracy is higher for concepts than for relationships; for concepts, precision and recall are above 95%, while for relationships precision is 83.7% and recall is 87.2%. This is not surprising, for a relationship is identified correctly only if the relationship name is perfectly matched and source and end concepts are identified. The very low number of false positives is striking. The few exceptions are caused by complex phrase chunks that the NLP tooling fails to identify (e.g., *User-only* was recognized as a compound concept user_only).

The accuracy of the generated conceptual model (Table III) is also pretty good, although a bit less precise than the accuracy of the individual user story analysis: overall precision is 85.2% and recall is 88.2%. Interestingly, precision and recall for relationships are comparable to those of Table II, while concepts have obtained lower accuracy: 89.8% precision and 91.4% recall. This is due to the fact that the individual user story set contained several occurrences of the same concepts that were correctly identified by the tool, while the concepts that the tool would not correctly identify had less occurrences.

2) CMSCompany: This company is located in the Netherlands, has 120 employees and serves circa 150 customers. Their supplied user story set consists of 35 stories for a complex CMS product for large enterprises. The user stories represent a snapshot of approximately a year of development in 2011. The data set of *CMSCompany* included 32 syntactically correct user stories. Despite the smaller size, this data set is particularly intriguing due to the use of long user stories with non-trivial sentence structuring such as: *"As an editor, I want to search on media item titles and terms in the Media Repository in a case insensitive way, so the number of media item results are increased, and I find relevant media items more efficiently"*.

TABLE II
ACCURACY OF INDIVIDUAL USER STORY ANALYSIS FOR THE WEBCOMPANY CASE (N=73).

	Concepts			**Relationships**			**Overall**		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Instances	387	3	15	211	10	31	598	13	46
Percentage	95.6	0.7	3.7	83.7	4.0	12.3	91.0	2.0	7.0
Precision	95.6			83.7			91.0		
Recall	96.3			87.2			92.9		

TABLE III
ACCURACY OF THE GENERATED CONCEPTUAL MODEL FOR THE WEBCOMPANY CASE.

	Concepts			**Relationships**			**Overall**		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Instances	106	2	10	147	8	24	253	10	34
Percentage	89.8	1.7	8.5	82.1	4.5	13.4	85.2	3.4	11.4
Precision	89.8			82.1			85.2		
Recall	91.4			86.0			88.2		

Table IV presents the results of the analysis of individual stories. The number of identified concepts is proportionally higher than for *WebCompany*: the manual analysis (TP+FN) reveals 8.9 concepts per user story (N=285) for *CMSCompany* compared to 5.5 concepts per user story (N=402) for *WebCompany*. This is due to the fact that the user stories for *CMSCompany* are longer and concept-rich.

If we analyze accuracy, we clearly see that precision and recall are lower for the CMSCompany data set. The lower accuracy is mostly due to the relationships (precision 67.2% and recall 75.3%); the identification of concepts also shows lower accuracy (precision 90.0% and recall 91.9%) but not significantly lower than in *WebCompany*. The main determinant of this performance is the existence of hard-to-process compound nouns consisting of three or four nouns, which the NLP tooling was unable to identify consistently. While English is a right-headed, left-branching language for compounds, the NLP parser did not always apply this rule, thereby identifying the wrong compound (C4) and its relationships (H2, R5). A missed out compound also implies missing out the relationships when the compound is a direct object (R2) or a non-direct object (R3 or R4).

Table V reports on the accuracy of the conceptual model. The less accurate parsing is also reflected on the conceptual model, which is not as accurate as that of *WebCompany* (Table III): overall precision and recall are 79.3% and 85.3%. However, unlike the first case study, the results for the relationships in the conceptual model are better than those for the individual user stories. The reason is that some relationships were missed out in some user stories, but recognized correctly in others, depending on the complexity of the sentence. Among the incorrectly identified relationships, most of them are due to the (non-)identification of a compound noun.

TABLE IV
ACCURACY OF INDIVIDUAL USER STORY ANALYSIS FOR THE CMSCOMPANY CASE (N=32).

	Concepts			**Relationships**			**Overall**		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Instances	262	6	23	119	19	39	381	25	62
Percentage	90.0	2.1	7.9	67.2	10.7	22.0	81.4	5.3	13.3
Precision	90.0			67.2			81.4		
Recall	91.9			75.3			86.0		

TABLE V
ACCURACY OF THE GENERATED CONCEPTUAL MODEL FOR THE CMSCOMPANY CASE.

	Concepts			**Relationships**			**Overall**		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Instances	124	4	13	114	17	28	238	21	41
Percentage	87.9	2.8	9.2	71.7	10.7	17.6	79.3	7.0	13.7
Precision	87.9			71.7			79.3		
Recall	90.5			80.3			85.3		

B. Analysis and Discussion

The overall results are promising and generally positive. The simple, semi-structured nature of user stories makes them an ideal candidate for NLP analysis. The problems arise when, as in the second case study, people deviate from the simplicity of the format and formulate complex requirements that go beyond the purpose of the "As a…I want…so that" template. In the next paragraphs, we present some key challenges concerning NLP processing that our evaluation revealed.

Compounds are a difficult element to identify correctly. Their proper identification depends on their position within a sentence. For example, the spaCy tagger would find the compound events_section in the sentence "I keep the events section", but it would miss it in the sentence "I can keep the events section". Also, we do not support compounds that consist of three nouns, such as "profile page statistics".

Verbs may be difficult to link to the proper object. For example, for *CMSCompany*, our implementation identifies the relationship avoid(marketeer,redirects) in the sentence "Marketeer can avoid duplicate content easily without having to set permanent redirects", instead of avoid(marketeer,content). While we focused on linking the subject to the main object of the sentence through the verb, there are also sentences requiring higher arity relationships. For instance, the sentence "The user can add new elements to the gallery" would require the creation of an n-ary association "add-something-to" tying together the concepts user, element and gallery.

Conjunctions are also a challenge. For the time being, we decided to overlook them, guided by our conceptual model of user stories that calls for their atomicity and minimality [13]. However, our second case study shows several examples of stories that violate these principles, using the conjunction "and" to specify multiple requirements or expressing conditions using the conjunction "when".

Additionally, we currently omit adjectives and adverbs. This prevents us from identifying specializations and quality requirements. For example, "external link" is an adjective-noun compound that could induce an attribute of link or a specialization. Also, adverbs that qualify verbs ("easily", "intuitively", "faster") could lead to more accurate relationships that indicate the qualities that the system should comply with.

VI. EVALUATION WITH EXPERTS

We evaluate the feasibility and usefulness of our approach by discussing the output of Visual Narrator with leading analysts from the companies that supplied the two data sets. During a 30-60 minute interview, the researcher showed two types of conceptual models. A single large conceptual model of the entire user story set and multiple smaller conceptual models; one for each role in the set. The purpose of the interview was to answer three questions:

Q1. What is the interviewee's initial perception of the applicability of the conceptual models?
Q2. How does the interviewee currently detect incompleteness, conflicts and dependencies between user stories?
Q3. How would the interviewee use the conceptual models to detect incompleteness, conflicts and dependencies?

To prevent hypothesis guessing and/or acquiescence bias, the researcher started the discussion by showing the generated images and asking the interviewee to talk freely. Only after (s)he explained the possible applications, the researcher mentioned incompleteness, conflicts and dependencies.

A. WebCompany

We conducted an interview with the founder and project lead of WebCompany. His initial perception (Q1) of the conceptual model was positive. At first glance, he immediately recognized which set of user stories the conceptual model was created from. While studying the details, the interviewee quickly noticed two inconsistencies within the conceptual model. In both cases, different words refer to the same real-world concept or action: information vs content and delete vs remove.

On the other hand, the interviewee pointed out that the conceptual model should distinguish visually between different concept types: roles, means, ends. Role, in particular, should be visually recognizable from the other concepts in a user story. Indeed, when reading the smaller conceptual model for each of the roles, the interviewee identified the actions quickly and easily. Nevertheless, the interviewee thought there is value in the holistic visualization when combined with the smaller role-centered conceptual models. By presenting the information on different levels of abstraction, the interviewee believes he can effectively present specific viewpoints.

The interviewee thinks Visual Narrator is primarily useful for one task: creating a common understanding of the intended software among business analysts, developers and clients. This is preferable than discussing the user stories themselves, because clients actively dislike reading a set of user stories. The visualization helps the stakeholders (Q3):

- Detect completeness of the application by discussing whether the visualization is missing any functionality;
- Ensure effective domain-driven design practices by detecting and resolving inconsistencies in used terminology;

- Prioritize requirements in an optimal sequence by identifying technical dependencies before development starts.

Concerning Q2, WebCompany ensures completeness by guiding clients through a *concept trajectory*: multiple sessions where the customer and project lead write user stories and create wireframes to ensure a common understanding. Yet, stakeholders frequently change their mind during development.

B. CMSCompany

We interviewed a software product manager of CMSCompany. His initial perception of the conceptual models (Q1) was positive. The interviewee's first reaction to the large overview was recognition of core elements of the user stories: roles, actions and objects. This was followed by hints on how to correct some details. While doing this, the interviewee remarked that he recognized the structure of the application in the conceptual model, despite the small size of the user story set. The interviewee was particularly impressed by Visual Narrator's ability to extract and distinguish between item, item_title and title (a compound). He explained that although the distinction between item_title and title might *seem* redundant, the system actually has four different kinds of item!

As a drawback, the interviewee noted that products with hundreds or thousands of user stories may lead to overly large and complicated models, thereby inhibiting the comprehensibility of the generated visualization. As a solution, the interviewee suggested to split the visualization along different dimensions: per release, role, epic or clustered in functional areas. This improvement could make our tool useful for:

- Identifying dependencies between user stories by visualizing the interrelationships;
- Educating new employees by explaining difficult details and nuances of the system;
- Analyzing the impact of a new release by projecting new changes on the current situation;
- Detecting unnecessary and/or redundant roles.

Concerning Q2, the interviewee explained that the company employs no specific way for detecting and resolving incompleteness, conflict and dependency issues. The poker planning meeting is the primary forum to uncover technical issues. Although incompleteness, conflicts and dependencies are not explicitly discussed during this meeting, the participants will bring them up when necessary. The interviewee expects the usefulness of Visual Narrator to be marginal during this meeting (Q3), because the generated conceptual models are not 100% correct, and software engineers are reluctant to work with inaccurate models.

From a functional perspective, the interviewee relies on instinct and experience to pro-actively detect potential conflicts as well as opportunities to combine dependencies. He estimated that a new employee needs between 1 or 2 years of experience to obtain this degree of proficiency. The interviewee finds that the highest potential for Visual Narrator is in educating and supporting recently hired employees that are not yet familiar with the system in detecting and resolving incompleteness, conflict and dependency issues (Q3).

VII. Discussion and Outlook

We summarize and present our conclusions in Sec. VII-A, discuss the most relevant threats to validity in Sec. VII-B, and outline our main future directions in Sec. VII-C.

A. Summary and conclusions

Natural language is the most adopted notation for requirements [2]. Unfortunately, text does not readily provide a holistic view of the involved entities and relationships. In line with other authors [6]–[9], we argue that extracting a conceptual model can ease the communication between stakeholders.

Our proposed Visual Narrator tool automatically generates a conceptual model from a collection of agile requirements expressed as user stories. The tool orchestrates a selection of state-of-the-art NL processing heuristics to do so. In order to obtain high precision and recall we focus on the key details of user stories, concepts and relationships, and ignoring more sophisticated aspects such as attributes and cardinality.

Our evaluation on two case studies showed promising accuracy results, especially when user stories are concise statements of the problem to solve—as they should be [15]—and not lengthy, low-level descriptions of the solution. The creators of the data sets gave a positive opinion of our approach, and their insights inform our future vision (Sec. VII-C). In comparison to state-of-the-art tools such as [11], our approach performs similarly in terms of class recall and precision, but it is superior in relationship recall. Keep in mind, however, that our approach includes fewer relationships heuristics.

B. Threats to validity

External validity threats reduce the generalizability of the results. More reliable accuracy results require evaluations on further data sets. Also, our examined user stories are written by people with professional English proficiency instead of by native speakers. Some words were misspelled and some uncommon grammar structures were used: the effectiveness of NLP may have been slightly affected.

Construct validity threats concern the degree to which a test measures what it intends to measure. Measuring accuracy requires understanding the correct interpretation of a sentence; it is well known that NL is *inherently* ambiguous. To reduce the risks, we limited ourselves to the more objective criterion of compliance with the algorithm, instead of the more general case of recognizing all concepts and relationships in the text.

Internal validity threats focus on how the experiments were conducted. The data set was tagged manually and there could be some subjective interpretation. Moreover, our data sets were obtained through convenience sampling from industry contacts that would be available to discuss the results.

C. Future directions: the Interactive Narrator

Our promising results pave the way for further work that extends and improves Visual Narrator. We aim to create the so-called *Interactive Narrator*, an advanced tool that fosters and facilitates effective discussion among stakeholders about user stories [15]. We envision Interactive Narrator as a real-time, modern documentation tool for agile development [30].

This tool should provide multiple visualization options of the conceptual model. Zooming in/out should enable revealing and hiding elements based on the weights of the elements computed by the WeightAttacher in Fig. 2. The Interactive Narrator should also support specific views for different stakeholder types [31], e.g., developers, clients, and users. We refer the interested reder to [32] for an exploratory study of techniques for visualizing Visual Narrator's output.

As highlighted by the interviewees in Sec. VI, Interactive Narrator should help identify and resolve inconsistencies, dependencies [33], and redundancies between the requirements. The conceptual model can also play a role in reducing the ambiguity of the requirements [34]. These possible uses are enabled by the advantage of a graphical representation over NL in holistically representing a given domain.

Several improvements can be made to the natural language processing algorithms. We should develop support for elaborate concepts such as complex compounds, verbs entailing n-ary relationships, conjunctions, adjectives, adverbs and references ('this', 'that', etc.), but particular attention should be paid to maintaining good-enough accuracy. Also, we could use techniques to identify synonyms and homonyms [35], especially in the context of redundancy identification.

REFERENCES

[1] C. J. Neill and P. A. Laplante, "Requirements Engineering: The State of the Practice," *IEEE software*, vol. 20, no. 6, p. 40, 2003.

[2] M. Kassab, C. Neill, and P. Laplante, "State of Practice in Requirements Engineering: Contemporary Data," *Innovations in Systems and Software Engineering*, vol. 10, no. 4, pp. 235–241, 2014.

[3] M. Kassab, "The Changing Landscape of Requirements Engineering Practices over the Past Decade," in *Proc. of EmpiRE*. IEEE, 2015, pp. 1–8.

[4] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "The Use and Effectiveness of User Stories in Practice," in *Proc. of REFSQ*, ser. LNCS, vol. 9619. Springer, 2016, pp. 205–222.

[5] D. M. Berry, E. Kamsties, and M. M. Krieger, "From contract drafting to software specification: Linguistic sources of ambiguity," School of Computer Science, University of Waterloo, ON, Canada, Tech. Rep., 2001.

[6] N. Omar, J. Hanna, and P. McKevitt, "Heuristics-Based Entity-Relationship Modelling through Natural Language Processing," in *Proc. of AICS*, 2004, pp. 302–313.

[7] S. Du and D. P. Metzler, "An Automated Multi-component Approach to Extracting Entity Relationships from Database Requirement Specification Documents," in *Proc. of NLDB*, ser. LNCS, vol. 3999. Springer, 2006, pp. 1–11.

[8] H. Harmain and R. Gaizauskas, "CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis," *Automated Software Engineering*, vol. 10, no. 2, pp. 157–181, 2003.

[9] S. Hartmann and S. Link, "English Sentence Structures and EER Modeling," in *Proc. of APCCM*, 2007, pp. 27–35.

[10] S. P. Overmyer, B. Lavoie, and O. Rambow, "Conceptual Modeling through Linguistic Analysis Using LIDA," in *Proc. of ICSE*. IEEE Computer Society, 2001, pp. 401–410.

[11] V. B. R. V. Sagar and S. Abirami, "Conceptual Modeling of Natural Language Functional Requirements," *Journal of Systems and Software*, vol. 88, pp. 25–41, 2014.

[12] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Forging High-Quality User Stories: Towards a Discipline for Agile Requirements," in *Proc. of RE*. IEEE, 2015, pp. 126–135.

[13] ——, "Improving agile requirements: the quality user story framework and tool," *Requirements Engineering*, pp. 1–21, 2016.

[14] Y. Wautelet, S. Heng, M. Kolp, and I. Mirbel, "Unifying and Extending User Story Models," in *Advanced Information Systems Engineering*, ser. LNCS, vol. 8484. Springer, 2014, pp. 211–225.

[15] M. Cohn, *User Stories Applied: for Agile Software Development*. Redwood City, CA, USA: Addison Wesley Professional, 2004.

[16] M. Saeki, H. Horai, and H. Enomoto, "Software Development Process from Natural Language Specification," in *Proc. of ICSE*. ACM, 1989, pp. 64–73.

[17] L. Mich, "NL-OOPS: From Natural Language to Object Oriented Requirements Using the Natural Language Processing System LOLITA," *Natural Language Engineering*, vol. 2, pp. 161–187, 6 1996.

[18] V. Ambriola and V. Gervasi, "On the Systematic Analysis of Natural Language Requirements with CIRCE," *Automated Software Engineering*, vol. 13, no. 1, pp. 107–167, 2006.

[19] 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, 2010.

[20] ——, "aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 3, pp. 13:1–13:52, 2015.

[21] F. Meziane and S. Vadera, "Obtaining E-R Diagrams Semi-Automatically from Natural Language Specifications," in *Proc. of ICEIS*, 2004, pp. 638–642.

[22] A. M. Tjoa and L. Berger, "Transformation of Requirement Specifications Expressed in Natural Language into an EER Model," in *Proc. of ER*, ser. LNCS. Springer, 1993, vol. 823, pp. 206–217.

[23] P. P. Chen, "Entity-Relationship Diagrams and English Sentence Structure," in *Proc. of the 1st International Conference on the Entity-Relationship Approach to Systems Analysis and Design*, 1983, pp. 13–14.

[24] M. Vela and T. Declerck, "A Methodology for Ontology Learning: Deriving Ontology Schema Components from Unstructured Text," in *Proc. of SAAKM*, 2009, pp. 22–26.

[25] G. Aguado De Cea, A. Gómez-Pérez, E. Montiel-Ponsoda, and M. C. Suárez-Figueroa, "Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns," in *Proc. of EKAW*, ser. LNCS, vol. 5268. Springer, 2008, pp. 32–47.

[26] E. S. Btoush and M. M. Hammad, "Generating ER Diagrams from Requirement Specifications Based on Natural Language Processing," *International Journal of Database Theory and Application*, vol. 8, no. 2, pp. 61–70, 2015.

[27] C. L. Gagné, "Lexical and Relational Influences on the Processing of Novel Compounds," *Brain and Language*, vol. 81, no. 1–3, pp. 723–735, 2002.

[28] M. Lapata, "The Disambiguation of Nominalizations," *Comput. Linguist.*, vol. 28, no. 3, pp. 357–388, 2002.

[29] W. Li, X. Zhang, C. Niu, Y. Jiang, and R. Srihari, "An Expert Lexicon Approach to Identifying English Phrasal Verbs," in *Proc. of ACL*, 2003, pp. 513–520.

[30] E. Rubin and H. Rubin, "Supporting Agile Software Development through Active Documentation," *Requirements Engineering*, vol. 16, no. 2, pp. 117–132, 2010.

[31] P. F. Pires, F. C. Delicato, R. Cóbe, T. Batista, J. G. Davis, and J. H. Song, "Integrating Ontologies, Model Driven, and CNL in a Multi-Viewed Approach for Requirements Engineering," *Requirements Engineering*, vol. 16, no. 2, pp. 133–160, 2011.

[32] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Visualizing user story requirements at multiple granularity levels via semantic relatedness," in *Proc. of ER*, 2016.

[33] J. Wang and Q. Wang, "Analyzing and Predicting Software Integration Bugs Using Network Analysis on Requirements Dependency Network," *Requirements Engineering*, pp. 1–24, 2014.

[34] D. Popescu, S. Rugaber, N. Medvidovic, and D. M. Berry, "Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models," in *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs*, ser. LNCS, vol. 5320. Springer, 2008, pp. 103–124.

[35] I. Omoronyia, G. Sindre, T. Stålhane, S. Biffl, T. Moser, and W. Sunindyo, "A Domain Ontology Building Process for Guiding Requirements Elicitation," in *Proc. of REFSQ*, ser. LNCS, vol. 6182. Springer, 2010, pp. 188–202.