Noname manuscript No.
(will be inserted by the editor)

Extracting Conceptual Models from User Stories with Visual Narrator

Garm Lucassen · Marcel Robeer · Fabiano Dalpiaz · Jan Martijn E.M. van der Werf · Sjaak Brinkkemper

Received: date / Accepted: date

Abstract Extracting conceptual models from natural language requirements can help identify dependencies, redundancies and conflicts between requirements via a holistic and easy-to-understand view that is generated from lengthy textual specifications. Unfortunately, existing approaches never gained traction in practice, because they either require substantial human involvement, or they deliver too low accuracy. In this paper, we propose an automated approach called Visual Narrator based on natural language processing that extracts conceptual models from user story requirements. We choose this notation because of its popularity among (agile) practitioners and its focus on the essential components of a requirement: Who? What? Why? Coupled with a careful selection and tuning of heuristics, we show how Visual Narrator enables generating conceptual models from user stories with high accuracy. Visual Narrator is part of the holistic Grimm method for user story collaboration that ranges from elicitation to the interactive visualization and analysis of requirements.

G. Lucassen (✉) · M. Robeer · F. Dalpiaz · J.M.E.M. van der Werf · S. Brinkkemper
Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands
E-mail: g.lucassen@uu.nl

M. Robeer
E-mail: m.j.robeer@uu.nl

F. Dalpiaz
E-mail: f.dalpiaz@uu.nl

J.M.E.M. van der Werf
E-mail: j.m.e.m.vanderwerf@uu.nl

S. Brinkkemper
E-mail: s.brinkkemper@uu.nl

Keywords User stories · requirements engineering · conceptual modeling · NLP · case study · conceptual model visualization

1 Introduction

The software industry commonly uses natural language (NL) notations to express software requirements [47], with NL being employed by over 60% of practitioners [32]. With the increasing adoption of agile development practices such as Scrum, the semi-structured NL notation of *user stories* is gaining momentum [31, 38].

NL requirements are easy to understand because they employ the very same language that we use to communicate with others. Nevertheless, NL suffers from several drawbacks too. The ambiguity of words and sentences is a well-known and widely studied problem that results in different interpretations of the same text. See Berry et al. [9] for an authoritative review. In this paper, we focus on another difficult problem: the identification and exploration of the key entities and relationships in a large set of requirements. Our work is intended to support the detection of dependencies between requirements, redundancies, and inconsistencies.

Our baseline consists of the existing literature on deriving conceptual models from NL requirements [20, 27, 28, 48]. However, we go beyond the limitations of these inspiring techniques, which either (i) *require human supervision* to appropriately tag the entities and relationships in the text [20, 49], or (ii) *have low accuracy*, often due to the ambitious attempt to support arbitrarily complex requirements statements [27, 57].

Several studies have shown the added value of conceptual modeling in software development [18, 22]. Yet, practitioners remain reluctant to adopt these methods.

In 2006, just 13% of the Australian Computer Society's 12,000 members indicated an interest in conceptual modeling [18]. A probable explanation is that the benefits do not outweigh the burden of constructing and maintaining a conceptual model manually.

Our goal is to overcome the limitations stated above and to promote the adoption of conceptual models for discussing about requirements. Our recipe includes three main ingredients: (i) our chosen notation is *user stories*, which are highly popular among practitioners [31, 38] and that express concisely the essential elements of requirements (Who? What? Why?); (ii) we minimize human supervision by proposing a *fully automated* software tool; and (iii) we deliver value by focusing on *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).

In previous work [53], we have shown the feasibility of this recipe by introducing the Visual Narrator tool for extracting conceptual models from user stories via NLP. Prior to that, we introduced a conceptual model and an NLP-enabled tool for assisting users in writing high-quality user stories [36] that obtained promising results [37]. In this paper, we extend the work in [53] by making two main contributions:

- We combine Visual Narrator with our other NLP tools *AQUSA* and *Interactive Narrator* into the comprehensive Grimm Method for conducting RE with user stories while stimulating the discussion about requirements among team members.
- We make technical improvements to Visual Narrator's algorithms and conduct a new quantitative evaluation with four cases, two of which are completely new, resulting in improved accuracy.

The rest of the paper is structured as follows. Section 2 introduces our Grimm Method that combines our research baseline. Section 3 reviews 23 heuristics from the literature and selects 11 for use with user stories. Section 4 presents the architecture, the main algorithm of the tool and elaborates upon the made technical improvements. Section 5 reports on our quantitative evaluation on four data sets from the software industry. We discuss related work in Section 6, while Section 7 presents conclusions and future directions.

2 Baseline: The Grimm Method

This paper is part of our ongoing research line on user stories. The premise of our research is the high adoption yet low quality of user stories in industry [32, 37, 38]. To improve this situation we focus on fostering deeper understanding of user stories by creating tool-assisted techniques that support practitioners in creating and communicating about high quality user stories [36, 39].

In Section 2.1 we present the conceptual anatomy of user stories, while in Section 2.2 we introduce the main elements of the Grimm method for conducting requirements engineering (RE) based on user stories.

2.1 Conceptual anatomy of user stories

Based on previous work [15, 37, 66] we define a generic representation of user stories—see the UML class diagram in Fig. 1—that dissects a user story into its constituents. User stories follow a standard predefined format [66] to capture three 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).

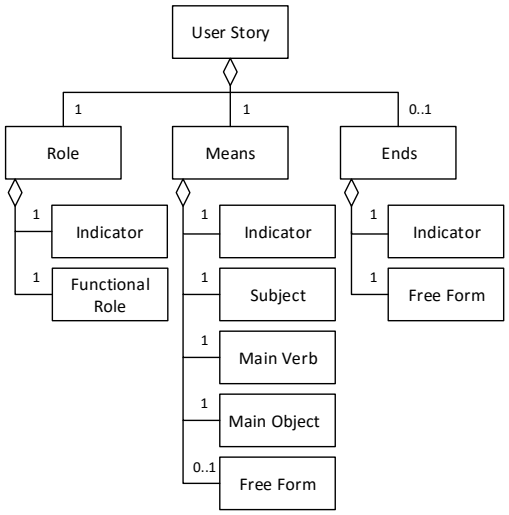

Fig. 1: Conceptual anatomy of a user story

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 Connextra template *"As a ⟨type of user⟩, I want ⟨some goal⟩ [so that ⟨some reason⟩]"* [15].

We distinguish between the *role*, *means* and *ends* parts of a user story. Each of these parts features an *indicator* delimiting the three basic parts of a user story. Jointly, the indicators are 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. 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*). The rest of the *means* can assume too many variations in practice; 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 [37]: there are at least three main purposes for having the ends: (1) clarifying the means, (2) referencing another user story, or (3) introducing a qualitative requirement. These functions can be combined for a single user story. This semantic distinction between ends types is beyond the scope of this paper and is left for future work.

2.2 The Grimm Method: Overview and Tooling

The Grimm method that we propose is our response to the low quality of user stories in industry [32,37,38], despite their popularity. Grimm features automated tools that improve the situation [37,39]. Fig. 2 illustrates how Grimm combines our tools in such a way to stimulate discussion around user stories among the stakeholders; this is one of the key objectives of user stories [15].

In the following paragraphs we elaborate on how each tool contribute to this goal: (1) AQUSA detects quality defects in human-made user stories, (2) Visual Narrator extracts entities and relationships from a user story collection to construct a conceptual model and (3) Interactive Narrator generates specialized views of the conceptual model that facilitate discussions for identifying inconsistencies, dependencies and ambiguities.

2.2.1 AQUSA

Grimm begins when stakeholders formulate some user stories. First, the AQUSA tool [37] validates their quality by automatically detecting defects using NL processing techniques. Based on the Quality User Story Framework, AQUSA focuses on those quality criteria can have the potential to be detected with 100% recall. Although

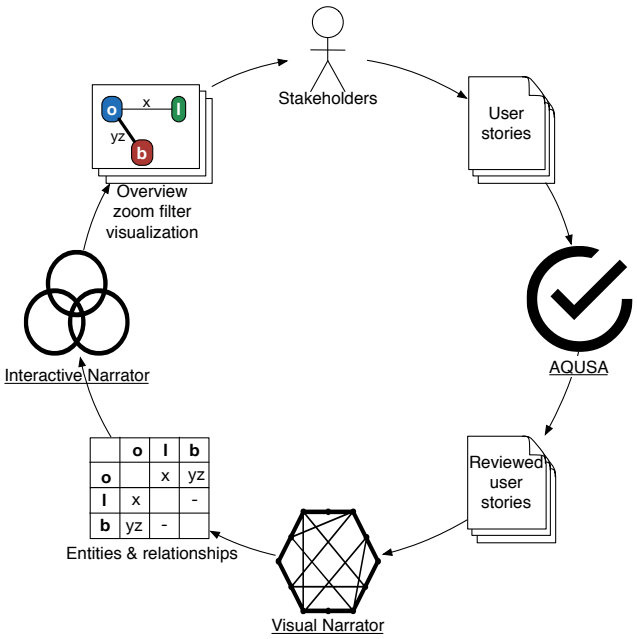

Fig. 2: The Grimm method for user-story-based RE

this Perfect Recall Condition is theoretically unattainable [55], AQUSA has proven capable of uncovering up to 90% of *easily preventable defects* in user stories [37]. Then, a human requirements engineer analyzes the reported errors and take corrective actions when needed. This leads to a revised collection of reviewed user stories, which may involve rephrasing some stories, splitting them, or removing text that is not essential for a user story such as references to design documents.

2.2.2 Visual Narrator

After preprocessing with AQUSA, the user stories can be analyzed further. Visual Narrator applies heuristics from the conceptual model field to extract the relevant entities and their relationships. The result is a comprehensive conceptual model of the entire user story collection. A key use case for this output is to check the completeness and consistency of the entities and relationships the user stories talk about. For example, one could identify isolated entities that are not connected to others and viewpoints that haven't been fully explored yet such as a role connected to very few entities. The remainder of this paper focuses on explaining Visual Narrator's inner workings and evaluating its accuracy.

Visual Narrator can generate output as a Prolog program (to be used for further automated reasoning) or an OWL 2 ontology. The latter can be used in readily available ontology visualization tools such as WebVOWL [35] to graphically explore the conceptual model.

2.2.3 Interactive Narrator

Although a preliminary evaluation with practitioners showed the potential of Visual Narrator, the extracted models quickly become too large for human analysts [53]. This cognitive overload is a well-known problem for conceptual model comprehensibility [4, 45]. As a response, we created the so-called *Interactive Narrator* [39] that generates specialized views of the conceptual model. Adhering to Shneiderman's Visual Information-Seeking Mantra: "overview first, zoom and filter, then details-on-demand" [58], we combine Visual Narrator's output with other data sources to create specialized views that highlight different user story elements: (1) clustering entities based on state-of-the-art semantic relatedness algorithm Word2Vec [25] and (2) filtering on specific user story details such as roles as well as project management data from issue trackers such as Jira[1].

As highlighted by our qualitative study via interviews concerning Visual Narrator [53], key requirements for this tool are to help identify and resolve inconsistencies, dependencies [64], and redundancies between the requirements. The literature shows that the conceptual model can also play a role in reducing the ambiguity of the requirements [50]. Interactive Narrator is intended as a real-time, modern documentation tool for agile development [54] that fosters and facilitates effective discussion among stakeholders about the software system (to be) [15]. These possible uses are enabled by the key advantage of graphical representations over NL in holistically representing a given domain.

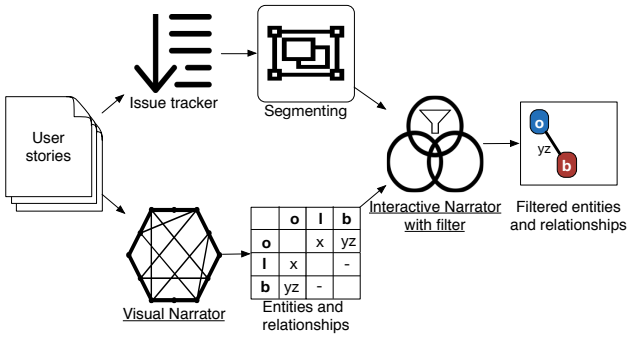

Fig. 3: Method for filtering a large conceptual model

While our initial prototype only supports semantic clustering and zooming[2], we have since built upon our proposal and realized a filtering prototype available on GitHub[3]. Aside from generating views that highlight a specific role or relationship, it supports filtering based on *agile artifacts*. In agile software development, user stories are frequently managed in an issue tracker such as Jira. These types of tools allow the user to organize a user story collection into meaningful chunks: epics, themes and sprints. Interactive Narrator combines this data with entities and relationships extracted using Visual Narrator as shown in Fig. 3. By providing the option to select any combination of these, the user can explore specific parts of the system (via epics and themes) or focus on certain development periods (sprints). For example: a user story can be part of the epic 'Presentation' in sprint 6 and simultaneously belong to the theme 'Conference' (see Fig. 4).

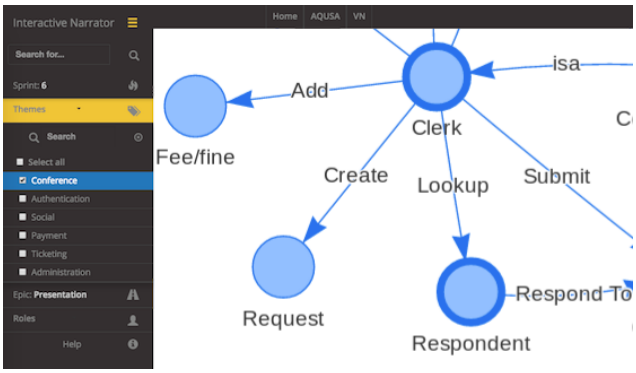

Fig. 4: Filter example showing entities and relationships for user stories in sprint 6 belonging to theme Conference and epic Presentation

Each of Interactive Narrator's views zoom in on specific aspects of the system, so as to help end users in identifying inconsistencies, dependencies and ambiguity between requirements [50, 63]. Stakeholders can then use the views as input for collaboratively resolving issues by modifying existing user stories or identifying new ones. This triggers a new method iteration in Fig. 2, which repeats until all issues are resolved.

3 NLP Heuristics for User Story Analysis

To extract meaningful models from NL requirements, researchers have been proposing *heuristic* rules for the identification of entities 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.

[1] https://www.atlassian.com/software/jira

[2] https://github.com/gglucass/Semantic-Similarity-Prototype

[3] https://github.com/Gionimo/VNwebapp

Table 1: Classification of heuristics found in previous literature for conceptual model generation.

ID	Rule head (if)	Rule tail (then)	Source(s)
Entities			
E1	Noun	Potential entity	[27, 43, 60]
E2	Common noun	Entity	[11, 28, 48, 60]
E3	Sentence subject	Entity	[57, 60]
E4	Compound noun	Take compound together to form entity	[57, 62]
E5	Gerund	Entity	[11, 57]
Non-hierarchical Relationships			
R1	Verb	Potential relationship	[43, 57]
R2	Transitive verb	Relationship	[11, 27, 28, 57]
R3	Verb (phrase) linking the subject and an object	Relationship	[27, 57, 60]
R4	Verb followed by preposition	Relationship including preposition	[48]
R5	Noun-noun compound	Non-hierarchical relationship between prefix and compound	[62]
Hierarchical Relationships			
H1	Verb 'to be'	Subjects are children of parent object	[28, 60]
H2	Head of noun-noun compound	IS-A relationship between compound and head	[62]
Attributes			
A1	Adjective	Attribute of noun phrase main	[11, 27, 28, 57, 60]
A2	Adverb modifying a verb	Relationship attribute	[11, 28]
A3	Possessive apostrophe	Entity attribute	[27, 57]
A4	Genitive case	Entity attribute	[48, 57, 60]
A5	Verb 'to have'	Entity attribute	[2, 27, 57, 60]
A6	Specific indicators (e.g. 'number', 'date', 'type', ...)	Entity attribute	[48]
A7	Object of numeric/algebraic operation	Entity attribute	[11]
Cardinality			
CA1	Singular noun (+ definite article)	Exactly 1	[27, 43, 60]
CA2	Indefinite article	Exactly 1	[27]
CA3	Part in the form "More than X"	$X..*$	[27, 48]
CA4	Indicators *many*, *each*, *all*, *every*, *some*, *any*	??..*	[27, 48]

Our literature study on conceptual model generation identified the 23 heuristics shown in Table 1. This overview groups the heuristics by the part of a conceptual model they generate: entities, non-hierarchical relationships, hierarchical relationships, attributes, and cardinality. The table presents a simple version of each rule as an implication from a condition (the *head*) to a consequence (the *tail*). As an example, the first entity heuristic should be read as **E1**: "If a word is a *noun*, then it is a *potential entity*."

The concise format and structure of user stories implies that not all heuristics are equally relevant. For example, user stories are not meant to include information about attributes or cardinality [15], thereby making those heuristics 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 mainstream Penn Treebank tags do not distinguish between gerunds and present participle. This exclusion process results in 11 heuristics that are particularly relevant for generating conceptual models from user stories. We explain and illustrate those 11 heuristics in the following.

3.1 Entities and non-hierarchical relationships

The most basic heuristics in the literature specify that (1) nouns in a sentence denote an *entity*, and (2) verbs indicate a potential *relationship* [10, 43]. This prompts us to define the first two heuristics:

E1. *"Every noun is a potential entity."*

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 E1 specifies to create two entities visitor and account and rule R1 originates a relationship between these entities named as the verb: create(visitor,account).

However, uncritically designating all nouns as entities would result in a conceptual model with superfluous entities. Previous authors have employed the distinction between *proper nouns* and *common nouns* to generalize some of the identified entities as more abstract instantiations [11, 28, 48, 60]. In general, common nouns are entities and proper nouns are instances of these entities that can be disregarded. *Transitive verbs* have a similar function, referencing an object in the sentence. These two phenomena lead to heuristics E2 and R2:

E2. *"A common noun indicates an entity."*

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

To form relationships between entities, a sentence should contain three components: 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:

E3. *"The subject of a sentence is an entity."*

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 two common nouns, one proper noun and one transitive verb. The person *John*—a proper noun—can be generalized to his job description *manager* (E2). Therefore, John is an instance of entity manager. Note that this proper noun defines the entity John because heuristic E3 says that, no matter its type, the subject of a sentence leads to an entity. The transitive verb has *website* as its direct object, 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 entity John (E3) with relationship design(John,website) (R3).

Concerning relationships, Omar et al. [48] 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* (E3). 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).

3.2 Compound Nouns

Compound nouns describe an entity that includes multiple words. Most often these are sequences of nouns or adjectives that precede a noun. To accurately construct a conceptual model, we consider the whole compound noun as the entity. Compound nouns are known to have many inherent relationships, as there are many ways to combine them [23]. However, extracting this requires the synthesis of lexical, semantic and pragmatic information, which is a complex task [33] that can hardly lead to accurate results. Therefore, we limit ourselves to a simple heuristic proposed by Vela and Declerck [62] by considering only compound nouns of length two:

E4. *"Noun compounds are combined to form an entity."*

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

Example D: The compound noun "event organizer" leads to entity event_organizer (E4) and a "has" relationship has_organizer(event,event_organizer) (R5).

3.3 Hierarchical Relationships

The ontology generation domain pays special attention to generalization relationships, often referred to as *IS-A* relationships [28, 60]. Tenses of the verb *to be* typically indicate a hierarchical relationship between two entities. 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 [62] note that the nouns in compounds have a generalization relationship.

Compound noun entities are a form of a more abstract entity, e.g., database_administrator is a type of administrator. This is captured by the following heuristic:

H2. *"If a noun-noun compound exists, the prefix of the compound is the parent of the compound entity."*

Example E: Consider the user story "As a visitor, I can change my account password." Here, heuristics E4, E5 and H2 apply on noun compound *account password*. We create compound entity account_password (E4), 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).

4 The Visual Narrator Tool

To automatically extract conceptual models from user stories, we developed the *Visual Narrator* tool that implements the 11 heuristics detailed in Sec. 3. Visual Narrator takes a set of user stories as input and generates a conceptual model as output[4]. It is built in Python and relies on the natural text processor *spaCy* (`http://spacy.io`), a recent proposal in NLP that implements algorithms needing minimal to no tuning and with excellent performance. Additionally, phrasal verb extraction is performed using Li's algorithm [34].

Our tool only accepts user stories that use the indicators as identified by Wautelet et al. [66]: *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 as shown in Fig 2 [37].

In addition to generating the conceptual model, Visual Narrator can also generate separate models per role to help 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 entity (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 entity weight.

4.1 Architecture

The conceptual architecture of Visual Narrator is shown in Fig. 5 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

[4] `github.com/MarcelRobeer/VisualNarrator`

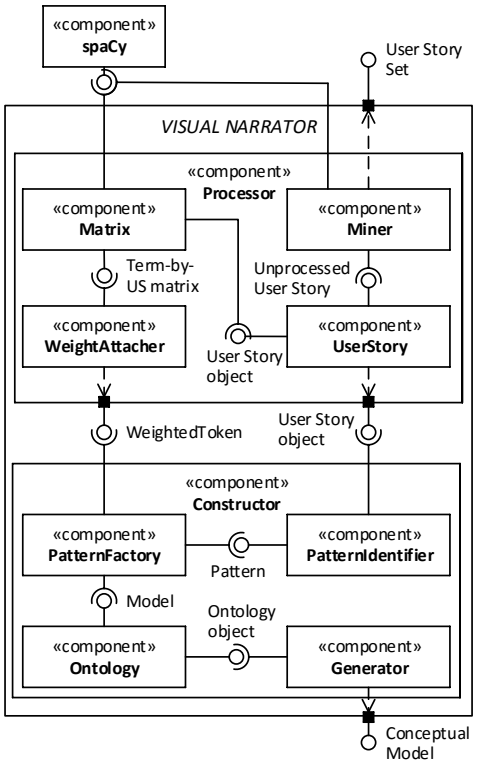

Fig. 5: Component Diagram of the Visual Narrator tool

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 entities 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 entities and relationships with a weight below an user-specified threshold. Finally, two Genera-

tors output an ontological representation of the Ontology object as an OWL 2 ontology and as a Prolog program.

4.2 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 user stories S as well as empty sets of entities E and relationships R. The procedure then populates E and R while parsing the user 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 entities, 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 (E2) and the subject of the sentence (E3) are executed (lines 9–10); all identified nouns are added to the set of entities E. Lines 11–14 process compound nouns: the compound is added to E according to heuristic E4, a specialization relationship is created linking the sub-entity to the super-entity (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 entities in E are linked through associations created by processing the verbs in the user stories. Lines 16–17 modify the means and the end by resolving the pronoun reference: 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 subject is "I" (e.g., "so that I. . . ").

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 user story 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 user story element (line 25). If both subject and object are in E, a relationship with the name of the verb is created between them (lines 26–27). In case only the subject is in E and we are analyzing the means, a relationship is created from the subject to a special entity called system. For example, the story "As a user, I want to login" would result in a relationship login(user,system). This rule applies only to the means because this part refers to a desired functionality, while the structure of the end is less rigid [37].

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

```
 1: procedure DeriveCM(Stories S, Entities E, Rels R)
 2:     ind = ({As a, ...},{I want, I can, ...},{So that})
 3:     for each s ∈ S do
 4:         (r,m,e) = split-by-indicators(s,ind)
 5:         if (r,m,e) == null then continue
 6:         else S' = S' ∪ (r, m, e)
 7:             for each p ∈ {r, m, e} do
 8:                 pt = create-parse-tree(p)
 9:                 E = E ∪ find-nouns(pt) [E2]
10:                 E = E ∪ subject-of(pt) [E3]
11:                 for each cn∈comp-nouns(pt) do
12:                     E = E ∪ {cn} [E4]
13:                     R = R ∪ IS-A(cn, prefix(cn)) [H2]
14:                     R = R ∪ has(prefix(cn), cn) [R5]
15:     for each (r, m, e) ∈ S' do
16:         replace-subject(m,r)
17:         if subject-of(e) == "I" then replace-subject(e,r)
18:         for each p ∈ {m, e} do
19:             subj = find-subject(p)
20:             if subj == null then continue
21:             v = find-main-verb(p)
22:             obj = find-direct-object(p) [R2]
23:             if obj == null then
24:                 obj = find-non-direct-object(p) [R3,R4]
25:             if obj == null then continue
26:             if subj, obj ∈ E then
27:                 R = R ∪ v(subj, obj)
28:             else if subj ∈ E ∧ p == m then
29:                 R = R ∪ v(subj, system)
```

4.3 Tool Implementation and Improvements

The aforementioned architecture and algorithm based on the heuristics of Sec. 3 capture the design and intention of Visual Narrator. The actual implementation, however, deviates from this in some aspects. It adapts some of the heuristics and includes a number of techniques to go beyond spaCy's initial Part-of-Speech (PoS) tagging in order to further optimize the results.

We modify E2 by including both common and proper nouns, because proper nouns often refer to domain-specific notions such as the name of a software product or a library. To improve accuracy, we replace the pronoun "I" with the noun they refer to when no ambiguity exists (see Sec. 4.2). Effectively, this means Visual Narrator assigns every noun to either a new or an existing entity. We did not implement H1 because the

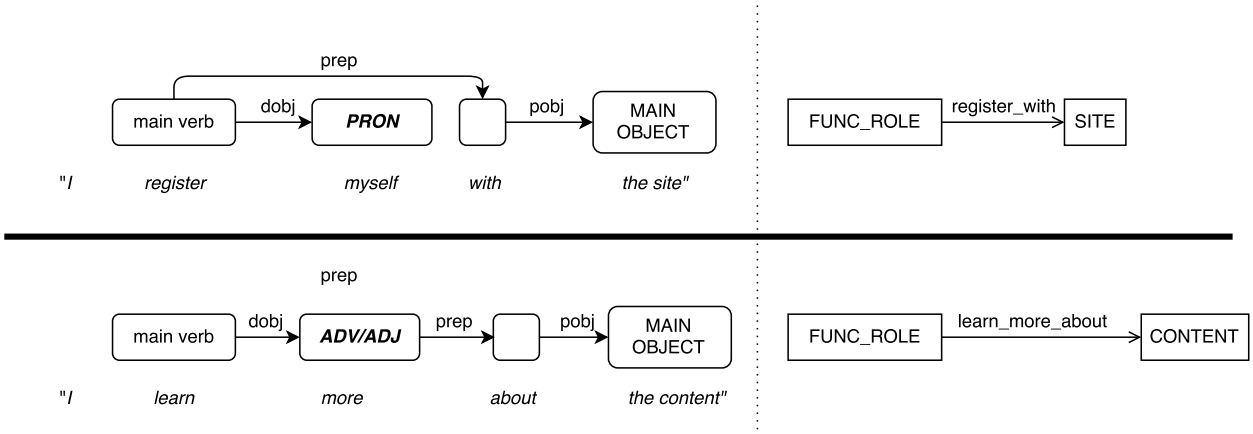

Fig. 6: Prepositional phrase parsing examples depicting assigning the pobj as main object improvement over [53]

correct application of the heuristics requires a deep understanding of the semantics of the "to be" relationship.

Furthermore, after separating a user story in its role, means and ends parts, Visual Narrator reruns the PoS tagger on each of these shorter parts. Because these partial user story phrases are tagged with higher accuracy we use these as the basis for further analysis. Even if the PoS tagger fails due to missing word classes in a fragmented or grammatically incorrect sentence, Visual Narrator includes fall-back mechanisms to parse the user story part: (i) when no noun is present in the role part the entire text between the two indicators is adopted as the role; (ii) if spaCy does not detect a verb and/or a direct object in the means, Visual Narrator presumes the word directly after the indicator is the verb and assigns the first object in the sentence as the direct object; and (iii) if no object is available at all, it assigns *System* as a default object.

To improve accuracy over our previous work [53], we carefully examined Visual Narrator's output to identify opportunities for improvement. This resulted in several minor bug fixes and the introduction of three new features. The first two features enhance Visual Narrator's detection of compound nouns, while the third changes how we detect a user story's main object:

- **Amod -** Previously, Visual Narrator relied solely on spaCy's "compound" dependency tag to identify two nouns as a compound noun such as "environment language". SpaCy, however, excludes all compound nouns that comprise a noun adjunct and a noun, instead assigning these the *adjectival modifier* dependency "amod". By including compound nouns with this dependency, Visual Narrator now correctly creates a compound for phrases such as "content types" and "chicken soup".
- **Rightmost child -** There are no agreed upon rules concerning the sequence of a compound noun. Generally speaking, however, the primary noun is the very last one with the nouns before it specializing its meaning: a full moon or a bus stop. Visual Narrator now favors the rightmost child for noun compounds with more than 2 nouns. Parsing the phrase "photo editing tools" now results in compound "editing tools" instead of "photo editing" as in [53].
- **Pobj is main object -** The direct object in a user story is, in general, the object of the role's action. This changes, however, when a user story has a pronoun, adverb or adjective as direct object and includes a *prepositional phrase*. Grammatically, an adverb or adjective cannot be the direct object as they qualify something else. Yet, the data does include this construction because POS taggers parse single adverbs and adjectives as a "noun phrase", resulting in a missing valid "dobj". When a prepositional phrase is present, the main object is found by following prep's "pobj" dependency link as shown in Fig.6. Where Visual Narrator would previously extract learn(I,more) from the user story 'I can learn more about the content', the tool now creates the arguably more meaningful relationship learn_more_about(I,content). Note that this new feature results in extra information by including the adverbs in the action, but simply replaces the pronoun with the functional role. We argue, however, that linking the pobj is more informative than keeping the pronoun's context. Consider for example "I register *the site*" versus "I register *myself*" in the case of a user story "I register myself with the site".

These new features combined with an update to the PoS tagger spaCy means Visual Narrator output is different from the output evaluated in [53]. In Sec. 5 we

report on a thorough new evaluation of Visual Narrator to investigate whether and to what extent these changes result in better recall and precision.

5 Quantitative Evaluation: Accuracy

We evaluate the feasibility of our approach and heuristic accuracy by applying Visual Narrator to four data sets from real-world projects: an interactive story telling project by WebCompany, a year of developing CMSCompany's flagship product, a public works project for Michigan State and six months of development for the open source archival software ArchivesSpace. In the following we introduce each in more detail. The data sets and their evaluation documents are available online[5]. An overview of the data sets' characteristics is provided in Table 2. The first five rows show the totals per data set: the number of stories n_{us}, of words n_{word}, of entities n_{ent}, of relationships n_{rel} and of words in the user story template n_{tmpl} (e.g., 'so that' means two words). The following three rows indicate averages per user story: number of words $\bar{x}_{word}$, of entities $\bar{x}_{ent}$ and of relationships $\bar{x}_{rel}$. Finally, to measure the user story writing style, the last three rows report on the conceptual density per data set by introducing three metrics for a given set of user stories, i.e., entity density ρ_{ent}, relationship density ρ_{rel} and concept density ρ_{conc}:

$$\rho_{ent} = \frac{n_{ent}}{n_{word} - n_{tmpl}}$$

$$\rho_{rel} = \frac{n_{rel}}{n_{word} - n_{tmpl}}$$

$$\rho_{conc} = \rho_{ent} + \rho_{rel}$$

Looking at the averages, the table reveals that CMSCompany's user stories are long and entity-rich and that despite being shorter, ArchivesSpace's user stories contain more concepts than Michigan State's. Among these four sets the average number of words does not correlate with average entities and average relationships. Looking at the density metrics, we can see how the ArchivesSpace' set is much more conceptually rich than Michigan State's and CMSCompany's; 85% as opposed to 58%. Nevertheless, we could find no correlation between density and the number of words. A detailed study of these metrics and their effect on quality in RE is left for future work.

In Sec. 5.1, 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 the authors. The outcome of this comparison is encouraging, we obtain accuracies as high as 97% recall and 98% precision with a lower bound of 88% recall and 92% precision. A small but important accuracy improvement over our earlier work [53]. In Sec. 5.2, we discuss the limitations of our approach in terms of important entities and relationships that are not recognized due to NLP limitations, our algorithm, or unanticipatable structure of the user stories.

[5] `http://www.staff.science.uu.nl/~lucas001/vn_user_stories.zip`

Table 2: Lexical characteristics of the evaluation cases.

	Web	*CMS*	*Michigan*	*Archives*
Totals per data set				
n_{us}	79	32	17	49
n_{word}	1549	954	414	783
n_{ent}	400	272	91	294
n_{rel}	247	168	42	156
n_{ind}	645	207	185	261
Averages per user story				
$\bar{x}_{word}$	19.6	29.8	24.4	16.0
$\bar{x}_{ent}$	5.1	8.5	5.4	6.0
$\bar{x}_{rel}$	3.1	5.3	2.5	3.2
Conceptual density per data set				
ρ_{ent}	0.44	0.36	0.40	0.56
ρ_{rel}	0.27	0.22	0.18	0.29
ρ_{conc}	0.71	0.58	0.58	0.85

5.1 Heuristics Accuracy

We evaluate the accuracy of our implemented heuristics by comparing the output of Visual Narrator to manually created golden data sets. In comparison to our evaluation in [53], we have taken a considerably more rigorous approach to constructing these golden data sets in order to ensure their validity: (1) two independent taggers applied Algorithm 1 to the user stories to identify all the entities and their relationships, (2) one tagger compared the two resulting documents and records all discrepancies between them, (3) together, the two taggers resolved the discrepancies by discussing the correct application of Algorithm 1. In our case, there was high agreement between the two taggers, with only 5% to at most 9% discrepancies depending on the data set.

To evaluate our implementation of Algorithm 1 we compare the golden data sets to the analysis by Visual Narrator.

Our evaluation has two objectives:

- To determine *quantitatively* to what extent the employed NLP toolkit fails to deliver accurate results due to the difficulty of correctly tagging and parsing 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 user story elements (entities and relationships). As is customary in Information Retrieval reporting, we do not report on true negatives because they shouldn't affect how good or bad the outcome is of your algorithm [41]:

- *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 (see Table 1) apply to a compound: C4, R5, and H2. 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 incorrect relationship) and a false negative (the missed out relationship).

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

Making this distinction is important when an unrecognized entity appears many 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 entities and 40 false negative relationships (20 times H2 and 20 times R5), while the resulting conceptual model—the actual artifact that we propose for the stakeholders to use in their discussion—would only miss one entity and two relationships.

Tables 3–10 report the results using the same format. They have three macro-columns: entities, relationships, and overall (entities+relationships). Each macro-column has three sub-columns to denote true positives (*TP*), false positives (*FP*) and false negatives (*FN*). The rows indicate the number of instances (*Inst.*), i.e., the number of identified and missed out entities and relationships; the percentile splitting of TP, FP, and FN (*%*), the precision (*PRC*), the recall (*RCL*) and the weighted harmonic mean of PRC and RCL using the F_1 score [41] (F_1).

5.1.1 WebCompany

This data set comes from a young company in the Netherlands that creates tailor-made web business applications. The team consists of nine employees who iteratively develop applications in bi-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.

79 of these 98 user stories were syntactically correct, usable and relevant for conceptual model generation [37]. Part of the generated conceptual model is shown in Fig. 7.

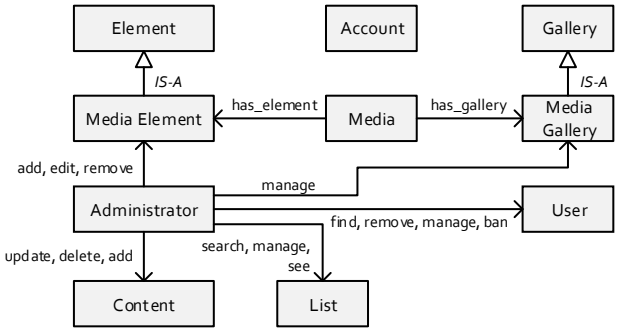

Fig. 7: Partial conceptual model for WebCompany's Administrator role based on Visual Narrator output

Table 3: Accuracy of individual user story analysis for the WebCompany case (N=79).

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	393	10	7	230	10	17	618	20	24
%	95.9	2.4	1.7	89.3	4.0	6.7	93.4	3.0	3.6
PRC	97.5			95.7			96.9		
RCL	98.3			93.0			96.3		
F_1	97.9			94.4			96.6		

Table 4: Accuracy of the generated conceptual model for the WebCompany case.

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	102	5	5	149	10	15	251	15	20
%	91.1	4.5	4.5	85.6	5.7	8.6	87.8	5.2	7.0
PRC	95.3			93.7			94.4		
RCL	95.3			90.9			92.6		
F_1	95.3			92.3			93.5		

The accuracy results of the individual user story analysis shown in Table 3 are positive. The overall precision and recall are 96.9% and 96.3%. The accuracy is higher for entities than for relationships; for entities, precision and recall are approximately 98%, while for relationships precision is 95.7% and recall is 93.0%. This happens because correctly identifying a relation-

ship depends on correctly identifying both the relationship name as well as its source and target entities.

The accuracy of the generated conceptual model (Table 4) is also very good, although a bit less accurate than that of the individual user story analysis: overall precision is 94.4% and recall is 92.6%. Interestingly, despite the large overlap in individual errors, the drop in recall and precision are comparable for both entities and relationships (95.3% for both vs. 93.7% precision and 90.9% recall).

Overall, Visual Narrator's output for the *WebCompany* data set is highly accurate. A closer examination of the false positives and false negatives reveals that there are two main causes for errors: (i) *human error* such as using *its* instead of it's confuses spaCy, and (ii) *ambiguous or complex phrases* that the NLP tooling fails to correctly identify, such as *up to date* erroneously resulting in a entity *date*.

5.1.2 CMSCompany

A data set from a mid-sized software company located in the Netherlands with 120 employees and 150 customers. They supplied 34 user stories for a complex CMS product for large enterprises; those stories represent a snapshot of approximately a year of development in 2011. A partial conceptual model is shown in Fig. 8. The data set of *CMSCompany* included 32 syntactically correct user stories. Despite the smaller size, this data set is particularly interesting due to the use of lengthy 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 5 presents the results of the analysis of individual stories. The accuracy for the CMSCompany data set is still high, despite the high complexity of its user stories in terms of average words, entities and relationships per user story as shown in Tab. 2. The overall precision and recall for individual user stories are 95.1% and 91.8%. This lower accuracy is mostly due to the 86.3% recall for relationships and in smaller part its 92.4% precision. Although the identification of entities also shows lower accuracy (precision 96.6% and recall 95.2%) the decrease is small considering the high user story complexity.

Table 6 reports on the accuracy of the generated conceptual model. The less accurate parsing is also reflected here with a less accurate conceptual model: overall precision and recall are 93.1% and 88.9%. While accuracy for entities is nearly identical to in the individual analysis case, the precision and recall for relationships drop to 89.6% and 83.0%.

The main determinant of this performance is the existence of hard-to-process compound nouns that include an ambiguous term such as "content" or "flash" which could also be an adjective or verb. In the worst case, Visual Narrator identifies the wrong compound (C4) and its relationships (H2, R5). As stated earlier, 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).

Fig. 8: Partial conceptual model for CMSCompany's Marketeer role in the conceptual model visualization tool Interactive Narrator

Table 5: Accuracy of individual user story analysis for the CMSCompany case (N=32).

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	259	9	13	145	12	23	404	21	36
%	92.2	3.2	4.6	80.6	6.7	12.8	87.6	4.6	7.8
PRC	96.6			92.4			95.1		
RCL	95.2			86.3			91.8		
F_1	95.9			89.2			93.4		

Table 6: Accuracy of the generated conceptual model for the CMSCompany case.

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	129	5	7	112	13	23	241	18	30
%	91.5	3.5	5.0	75.7	8.8	15.5	83.4	6.2	10.4
PRC	96.3			89.6			93.1		
RCL	94.9			83.0			88.9		
F_1	95.6			86.2			90.9		

5.1.3 Michigan State

These user stories are extracted from a State of Michigan Enterprise Procurement document. This publicly available document created by the Department of Technology, Management, and Budget in Lansing, Michigan describes a *Statement of Work* concerning the scope and definition of the professional services to be provided by a contracting company: Accela, Inc. For our analysis, we extracted 27 user stories for a complaints system. Although these user stories are concise and obviously written by English native speakers, Visual Narrator could only parse 17 out of 27 user stories, of which Fig 9 shows a partial model. The non-parsed user stories contain minor well-formedness issues that could quickly be resolved by preprocessing with AQUSA [37].

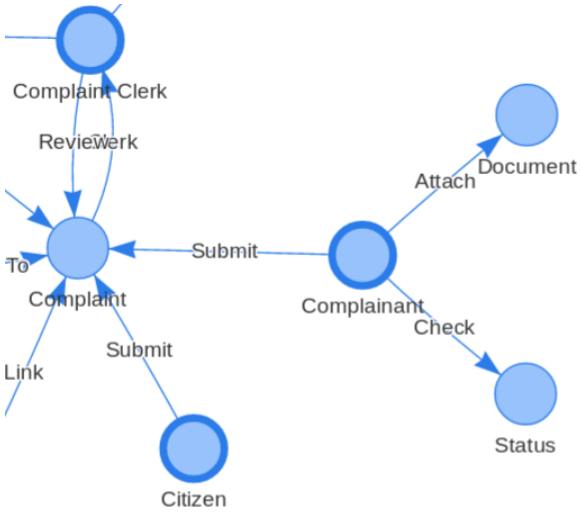

Fig. 9: Partial conceptual model for Michigan State's Complaint concept in Interactive Narrator

Despite the small magnitude, the data set is interesting for its extreme simplicity and consistent structure. Although its 5.4 average entities per user story is actually slightly higher than *WebCompany*'s 5.1, the accuracy results shown in Table 7 for individual stories push the boundary of what Visual Narrator can achieve: 97% precision and 98.5% recall. One reason for this high accuracy is the inclusion of just 6 compound nouns. In total, there are four errors. Two wrongly identified entities lead to a 98% precision and 100% recall for entities, and two mistakenly generated relationships result in identical precision and recall of 95.2%. Due to the low number of errors, the accuracy of the generated conceptual model is nearly identical: 96.6% precision and recall overall, 95.8% precision and 97.9% recall for entities and 97.4% precision and 95.0% recall for relationships (Table 8). Of all the errors in the conceptual model, 4 errors are the consequence of mistakes by the NLP tooling, while the remaining 2 emerge because Visual Narrator does not include personal pronouns in the conceptual model even when they are the subject (with the exception of "I", which is replaced by the referential element as per line 17 in Algorithm 1).

Table 7: Accuracy of individual user story analysis for the Michigan State case (N=17).

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	91	2	0	40	2	2	131	4	2
%	97.8	2.2	0	90.0	4.5	4.5	95.6	2.9	1.5
PRC	97.8			95.2			97.0		
RCL	100.0			95.2			98.5		
F_1	98.9			95.2			97.8		

Table 8: Accuracy of the generated conceptual model for the Michigan State case.

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	46	2	1	38	1	2	84	3	3
%	93.9	4.1	2.0	92.7	2.4	4.9	93.3	3.3	3.3
PRC	95.8			97.4			96.6		
RCL	97.9			95.0			96.6		
F_1	96.8			96.2			96.6		

5.1.4 ArchivesSpace

ArchivesSpace[6] is an open source software product created by archivists such as those of the British Royal Archive. The user stories of this project are available online[7]. For our analysis, we extracted 56 user stories that span development from the start of the project in August 28, 2012 until February 28, 2013. Out of those stories, 49 are syntactically correct. The user stories in this collection are quite peculiar: all of them omit the *so that* part of the Connextra template [15], but many user stories contain unnecessarily capitalized words, compound nouns and idiosyncratic phrases such as "...edit for (Accession | Resources) before ...".

This non-standard approach has a substantial impact on the analysis and the conceptual model (Fig. 10). Despite omitting the *so that*, Visual Narrator's accuracy is lowest on the ArchivesSpace data set. For individual user stories, the overall precision is 92.3% and recall is 88.4%. For relationships in particular, precision is 87.6% and recall is 81.4%. Likewise, accuracy in identifying entities is the lowest out of all four sets: 94.8% precision and 92.2% recall.

[6] `http://www.archivesspace.org/`
[7] `archivesspace.atlassian.net`

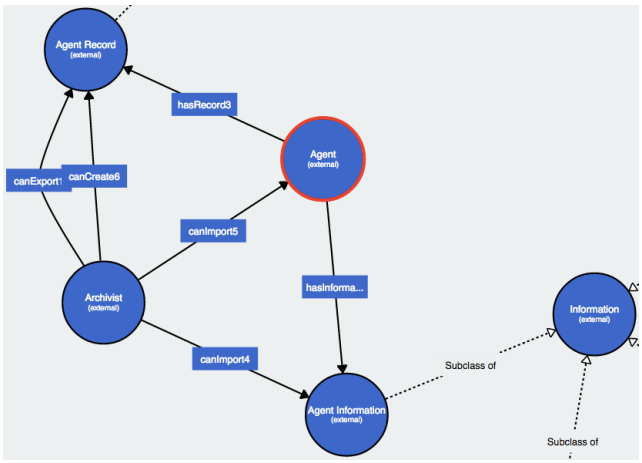

Fig. 10: Partial conceptual model for ArchivesSpace's Agent concept using the publicly available visualization tool WebVOWL

Table 9: Accuracy of individual user story analysis for the ArchivesSpace case (N=49).

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	271	15	23	127	18	29	398	33	52
%	87.7	4.9	7.4	73.0	10.3	16.7	82.4	6.8	10.8
PRC	94.8			87.6			92.3		
RCL	92.2			81.4			88.4		
F_1	93.4			84.4			90.4		

Table 10: Accuracy of the generated conceptual model for the ArchivesSpace case.

	Entities			*Relationships*			*Overall*		
	TP	*FP*	*FN*	*TP*	*FP*	*FN*	*TP*	*FP*	*FN*
Inst.	137	6	10	116	17	23	253	23	33
%	89.5	3.9	6.5	74.4	10.9	14.7	81.9	7.4	10.7
PRC	95.8			87.2			91.7		
RCL	93.2			83.5			88.5		
F_1	94.5			85.3			90.0		

Table 10 reports on the accuracy of the conceptual model. Again, the accuracy results are the worst out of all four data sets: overall precision is 91.7% and recall is 88.5%. Remarkably, however, the precision of entities and recall of both entities and relationships in the conceptual model is superior to individual user story analysis. 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.

The majority of incorrectly identified relationships and entities are due to the (non-)identification of compound nouns. For example: the text chunk "(Accession | Resources)" resulted in the creation of the compound noun / *Resources*, while Visual Narrator did not identify a compound noun for the jargon term *Subject heading*.

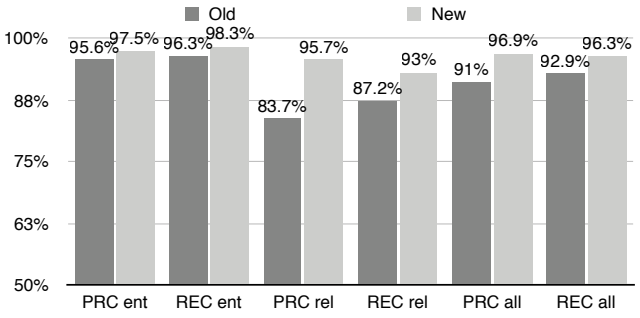

Fig. 11: Web individual PRC and RCL change

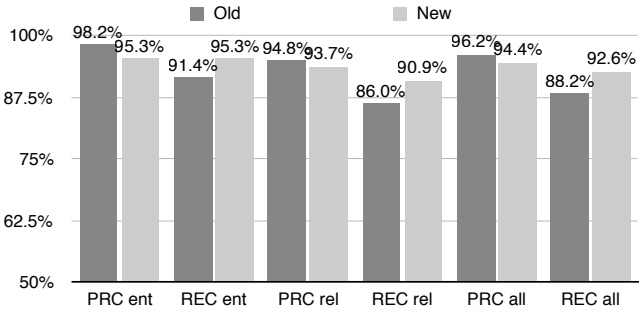

Fig. 12: Web set PRC and RCL change

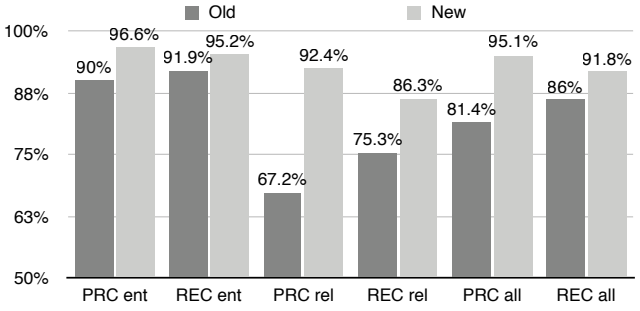

Fig. 13: CMS individual PRC and RCL change

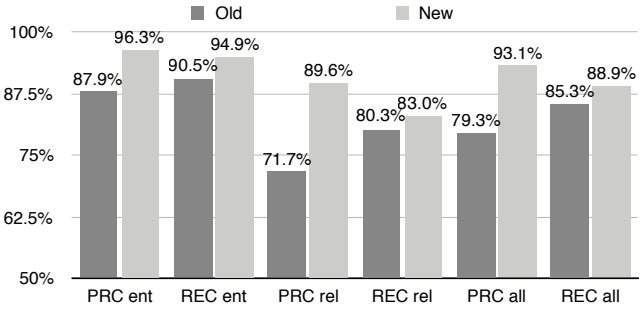

Fig. 14: CMS set PRC and RCL change

5.2 Analysis and Discussion

The changes described in Sec. 4.3 generally improved the accuracy of Visual Narrator over our previous results [53] as highlighted by the bar charts in Figs. 11,

12, 13, 14. For the WebCompany case, the average precision for entities and relationships actually dropped by 1%. In exchange, however, recall improved substantially. Entity recall for individual analysis and generated model improved by 2% and 4%, while relationship recall improved by 6% and 5%. The improvement for CMSCompany is even better, although there were more opportunities for improvement. On average, precision improved by 1.5%. Entity recall improved by 3% and 4%, whereas relationship recall improved by 11% and 3%. WebCompany's overall accuracy raised from 97.9% precision and 92.9% recall to 96.9% and 96.3% for individual analysis, and from 96.2% precision and 88.2% recall to 94.4% and 92.6% for the generated model. CMSCompany's overall accuracy changed from 93.8% precision and 86.0% recall to 96.1% and 91.8% for individual analysis, and from 91.9% precision and 85.3% recall to 93.1% and 88.9% for the generated model.

The simple and semi-structured template of user stories makes them an ideal candidate for NLP analysis as demonstrated by the high accuracy of Visual Narrator. Problems arise when, as in the second and fourth case study, people deviate from the basic format and formulate complex requirements that go beyond the purpose of the user story template. This had a substantial impact for the ArchivesSpace set in particular, with overall accuracies of just 92% recall and 88.5% precision for both individual user story analysis and generated conceptual model. Conversely, the Michigan State user stories are parsed with the highest accuracy thanks to their simplicity and consistency. In the following, we present some key challenges concerning NLP processing for user stories that our evaluation revealed.

Compounds are difficult 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". This is due to the fact that modern NLP taggers are statistical and try to find the most probable tagging in many situations [40]. Furthermore, we do not yet support compounds that consist of three or more nouns, such as "profile page statistics". Note, however, that these are often grammatically complex structures with ambiguous semantics: in the example, do "page statistics" refer to the "profile", or do "statistics" refer to the "profile page"? While a part of speech tagger is capable of recognizing "profile page statistics" as a three-word compound, it is unable to disambiguate it semantically.

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). Although spaCy is capable of correctly parsing this sentence, this error occurs due to our simplification of the sentence which replaces "Marketeer can" with "I". In this case spaCy incorrectly tags the ambiguous adjective "duplicate" as a verb and the equally ambiguous noun "content" as an adjective. In future work, we will re-evaluate whether a different approach to simplification is necessary.

Furthermore, 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 entities 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 [37]. However, our second and fourth case study include multiple examples of stories that violate these principles, using the conjunction "and" (but also logics-inspired operators such as "|") to specify multiple requirements or expressing conditions using the conjunction "when".

Additionally, we currently **exclude adjectives and adverbs from the conceptual models**. 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 for the system to comply with. Future work should study whether or not tagging these additional components would produce *more useful* conceptual models: the gained completeness may result in less accuracy, and the trade-off has to be investigated empirically in industrial practice.

Finally, **the human element is difficult to plan for**. A substantial portion of the errors in the WebCompany case are caused by grammatical errors or misspellings. The lower accuracy of the ArchivesSpace case can partly be attributed to the usage of the logical OR sign |. Preprocessing these types of issues would further improve the results of Visual Narrator. An interesting direction is to provide interactive support to the stakeholders expressing these requirements, e.g., via a grammar checker or the AQUSA tool [37].

5.3 Threats to validity

External validity threats reduce the generalizability of the results. To obtain even more reliable accuracy results requires substantially larger evaluations. Moreover, our data sets were obtained through convenience sampling from industry contacts and online searches. It is unclear whether these data sets are representative of user stories in general.

Construct validity threats are about 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 general case of recognizing all entities and relationships. In fact, we argue that comparing Visual Narrator output with theoretically ideal output is a more objective evaluation than comparing with a human's subjective recognition of all relevant concepts and relationships in a text.

Nevertheless, by taking this approach we are unable to empirically evaluate the validity of the algorithm itself. While our precision and recall for detecting what we want to detect is high, we do not know whether what we want to detect is actually useful. In earlier work, we conducted a preliminary evaluation with two practitioners who indicated Visual Narrator output is usable and promising [53]. However, a larger scale evaluation with practitioners is necessary to determine the subjective validity of the algorithm.

Internal validity threats focus on how the experiments were conducted. The golden data sets were tagged and applied manually. Although multiple human taggers were used, there is still a low risk that human error caused incorrect analysis.

6 Related Literature

6.1 User stories

The growing popularity of agile development practices such as Scrum leads to a continuously increasing adoption of user stories [31, 64]. Although research interest is slowly growing, the currently available literature is limited. Arguably, the first literature on incorporating stories into requirements is within the scenario-based requirements engineering domain in the 1990s [29]. The earliest research work on *user* stories proposes their use as the first artifact for describing interactions containing a role, an action and some object [30]. For implementation purposes the authors propose transforming user stories into a more formal notation such as use cases.

The majority of research in the field attempts to create methods and tools that support or improve user story practice. In 2002, Rees proposed to replace the pen-and-card approach for writing user stories with a software tool called DotStories [52]. Today's popular agile project management tools such as Jira and Trello are all built around the skeuomorph index card metaphor introduced with DotStories.

As of 2015 academic interest in user stories is renewed, leading to a variety of research initiatives. Recent studies predominantly investigate how to connect and/or integrate user stories with different modeling techniques. For example, Trkman et al. propose a method to associate user stories with business process modeling activities [61]. They find that undergraduate students better understand user stories' execution order and integration dependencies when business process models are available.

A variety of studies propose new ways to employ or work with user stories to achieve some goal. For example, Barbosa et al. demonstrate how to identify semantically duplicate user stories by applying well known similarity measures. They test their approach with 3 case sets, automatically identifying up to 92% of duplicates [7]. Observing that practitioners encounter difficulties incorporating user experience concerns into user stories, Choma et al. propose extending the Connextra template [13]. Although their case company fully adopted this new template, more research is necessary to evaluate the added benefit.

Other studies investigate some aspect of user stories in practice. Dimitrijevic et al. qualitatively compare five agile software tools in terms of their functionality, support for basic agile RE entities and practices, and user satisfaction. They conclude that basic functionality is well supported by tools, but that user role modeling and personas are not supported at all [19]. Soares et al. investigate the link between user stories and documentation debt, finding that the low level of detail of user stories is the primary reason for difficulties [59].

Finally, two authors take a different approach to ours for transforming a user story set into a visual representation. Wautelet et al. propose a method that results in Use-Case Diagrams [65]. The authors demonstrate a CASE-tool that automates their approach and allows end-users to iteratively improve and extend the output. In another project [67], these same authors propose a method for mapping user stories to agent-based software architecture using i* [17, 68]. Similarly, the US2StarTool [42] derives skeletons of i* goal models from user stories.

Whereas our approach employs NLP to extract all entities and relationships from a user story, these tools strictly map the entire action text to a task and the benefit to a goal. While employing the i* and use case notations enable these models to be more expressive than ours, they also require a human to actually construct a model by assembling the extracted parts. Visual Narrator's fully automatic model generation allows all stakeholders to quickly get an understanding of the software system's functionalities and partake in relevant and meaningful discussion around the user stories. A key activity according to Cohn [15].

Aside from facilitation communication, Visual Narrator output functions as the foundation for employing user stories to achieve other goals. For example, combining the extracted entities with semantic similarity calculations enables grouping user stories in clusters as we do in [39] or identifying semantically duplicate user stories similar to the work in [7]. Another application we are currently exploring is reconciling Visual Narrator output with class diagrams by automatically connecting an extracted entity to the source code it executes via automated acceptance tests.

6.2 NLP for RE: Extracting models from requirements

Historically the final frontier of RE has been applying natural language processing. Nowadays, the ambitious objective of full automation is considered unattainable in the foreseeable future [8,55]. Therefore, RE research has applied NLP to specific use cases. Berry et al. [8] categorizes the fundamental approach of all NLP RE tools into four types:

1. Finding defects and deviations in natural language (NL) requirements document;
2. Generating models from NL requirements descriptions;
3. Inferring trace links between NL requirements descriptions;
4. Identifying the key abstractions from NL documents

In [37] we provide an overview of contemporary tools in NLP for RE and introduce the AQUSA tool for automatically detecting quality defects in user stories. Furthermore, observing that no objective comparison of NLP for RE tools is available, Arendse and Lucassen apply three tools of type I on 112 requirements [5]. They find that none of the available tools is clearly superior and call for a next generation tool framework that can dynamically incorporate the new state of the art.

The objective of type II tools, generating models from NL requirements descriptions 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 [56]. One of the first tools that implement this idea is NL-OOPS [44]. 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 [27] managed to extract *candidate* attributes, entities and relationships from a 220 word text with 73% recall and 66% precision. CIRCE [3] 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 the form of a controlled vocabulary and grammar in order to generate complete and consistent models, thereby hampering adoption in practice. Note that while the Connextra user story template required by our solution imposes a three-part structure, each part is completely free form.

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 [69]. 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. However, its output constitutes initial models that require human intervention in the form of refinements done by experts [70]. Similarly, the tool presented in [57] outperforms novice human modelers in generating conceptual models from natural language requirements. Overall, their recall for identifying entities ranges from 85% to 100% depending on the case, while their precision is between 81% and 94%. The performance for relationships between entities, however, is less impressive: recall is in between 50% and 60%, while precision is in the 80%-100% range. Arora et al. [6] take a similar approach to the work we present in this paper, combining existing state-of-the-art heuristics for domain model extraction. They apply their approach to four industrial requirements documents. According to expert evaluation the extraction rules in their implementation achieve correctness between 74% and 100%.

Although [42,65,67] discussed in Sec. 6.1 all introduce methods and tools for extracting models from user stories, these works lack empirical evaluations. Because of this, comparing their accuracy with the current state

of the art or our work is impossible. Nevertheless, the proliferation of work on extracting models from user stories using NLP signifies the relevance and timeliness of our work. This paper is the first to empirically demonstrate user stories' potential in automatically extracting conceptual models and not without merit. The results of our evaluation show that our approach achieves state-of-the-art recall and precision for both identifying entities as well as relationships. Note that it is difficult to compare the recall and precision of our approach with aforementioned studies, because of the different evaluation methods each employs.

6.3 Visualization of Conceptual Models and Requirements

To take advantage of human's visual processing power conceptual models are typically presented as a visualization. Typically, new visual notation initiatives first concern is formally specifying the information types to support. As the popularity of a graphical notation increases, the community around it proposes innovative features to augment or simplify the notation. Unfortunately, aesthetic aspects such as attractive design and user experience are mostly disregarded when it comes to conceptual models [26]. Indeed, the most popular conceptual modeling paradigms such as ER, UML and BPMN primarily distinguish elements using basic symbols and shapes [24].

There is a large body of work to inspire innovative visualization. For example, in 2009 Moody proposed the Physics of Notations [45] that provides some guidelines for creating cognitively effective notations and for pinpointing flaws in existing notations such as i^* [46] and use case maps [24]. Moody's guidelines inspired further research that is relevant for our work. For example, Dudáš, Zamazal and Svátek identified seventeen relevant features implemented by ontology visualization tools [21] such as *incremental exploration* and *fisheye distortion*. Their notation, however, is mostly disregarded. Aside from this, other domains such as interaction design and geography are considerably more sophisticated in effectively visualizing information [12].

A systematic review of the requirements engineering visualization (REV) literature by Abad et al. [1] concludes that more investigation and research is needed to support knowledge visualization for RE. Similarly, Cooper et al. [16] review the papers that appeared in the REV workshops between 2006 and 2008. They distinguish between different types of visualizations: tabular, relational, sequential, hierarchical, and metaphorical/quantitative. The most relevant categories for our work are the relational (i.e., graphs) and hierarchical (decomposing a system into its parts). While many relational approaches exist, very few focus on hierarchal aspects, which are the key in our work.

Indeed, there are few contemporary tools available that focus on visualizing requirements as models. To date, ReCVisu+ [51] is the most effective tool for requirements visualization. ReCVisu+ supports different visual exploration tasks and employs clustering techniques and semantic similarity to reduce complexity. While Visual Narrator only visualizes one type of entity and directed relationship, our work on Interactive Narrator continuously explores innovative techniques to convey additional meaning. For this we take inspiration from existing tools, yet are careful to consider their appropriateness in our context. While RecVisu+ determines similarity based on the frequency of co-occurrence in system documentation, the Interactive Narrator we propose in [39] relies on corpus-based techniques that do not require the existence of additional system documentation. Moreover, we do not consider only concepts but also relationships, necessitating a different yet similar approach.

7 Conclusion and Future Work

Natural language is the most adopted notation for requirements [32]. Unfortunately, text does not readily provide a holistic view of the involved entities and relationships. Aligned with other authors [20, 27, 28, 48], we argue that extracting a conceptual model can significantly ease the communication between stakeholders.

We have proposed the Grimm method that combines three NLP-enabled tools to support conducting user story based RE. Specifically, the main artifact described in this paper is the Visual Narrator tool, which automatically generates a conceptual model from a collection of agile requirements expressed as user stories. To do so, the tool orchestrates a selection of state-of-the-art NL processing heuristics and relies on an off-the-shelf NLP toolkit, spaCy.

Our evaluation on four case studies showed positive accuracy results, especially when user stories are concise statements of the problem to solve [15] and not lengthy descriptions of the solution. Overall, our approach achieves similar recall and precision to state-of-the-art tools such as [57], for both entities and relationships. This happens thanks to the careful selection and implementation of heuristics that can deliver accurate results combined with our application of these heurstics based on careful dissection of user stories' syntactical properties.

Improvements can be made to the user story processing algorithms. In particular, we want to develop

support for elaborate entities such as complex compounds, n-ary relationships, conjunctions, adjectives, adverbs and references such as 'this' and 'that'. While doing so, however, we aim to make considerate decisions based on maximizing accuracy.

Furthermore, we intend to create a fully functional version of the Interactive Narrator that integrates with the AQUSA Tool and Visual Narrator. We will then empirically evaluate the Grimm method introduced in Section 2.2 with practitioners.

Another direction is to study the relationship between user story metrics such as average number of entities, relationships and their density (see Table 2) and quality of the created specifications and software. To do so, we need to collect data about how the user stories were used in the later stages of software engineering.

Finally, we want to investigate automated techniques to connect the generated conceptual models to software architecture views. One promising direction is taking advantage of automated acceptance tests, which typically include a hand-coded cross-reference to the user story they test [14]. For interpreted languages, executing these test cases produces a runtime trace related to the user story. By extracting the accessed data entities, classes and methods it is possible to dynamically construct a class diagram that includes only those classes contained within the runtime trace.

References

1. Abad, Z.S.H., Ruhe, G., Noaeen, M.: Requirements Engineering Visualization: A Systematic Literature Review. In: Proceedings of the International Requirements Engineering Conference (RE). IEEE (2016)
2. Aguado De Cea, G., Gómez-Pérez, A., Montiel-Ponsoda, E., Suárez-Figueroa, M.C.: Natural Language-Based Approach for Helping in the Reuse of Ontology Design Patterns. In: Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW), *LNCS*, vol. 5268, pp. 32–47. Springer (2008)
3. Ambriola, V., Gervasi, V.: On the Systematic Analysis of Natural Language Requirements with CIRCE. Automated Software Engineering **13**(1), 107–167 (2006)
4. Aranda, J., Ernst, N., Horkoff, J., Easterbrook, S.: A Framework for Empirical Evaluation of Model Comprehensibility. In: Proceedings of the Workshop on Modelling in Software Engineering (MiSE) (2007)
5. Arendse, B., Lucassen, G.: Toward Tool Mashups: Comparing and Combining NLP RE Tools. In: Proceedings of the International Workshop on Artificial Intelligence for Requirements Engineering (AIRE) (2016)
6. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting Domain Models from Natural-language Requirements: Approach and Industrial Evaluation. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 250–260. ACM (2016)
7. Barbosa, R., Silva, A.E.A., Moraes, R.: Use of Similarity Measure to Suggest the Existence of Duplicate User Stories in the Scrum Process. In: Proceedings of the Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), pp. 2–5 (2016)
8. Berry, D., Gacitua, R., Sawyer, P., Tjong, S.: The Case for Dumb Requirements Engineering Tools. In: Proceedings of International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), *LNCS*, vol. 7195, pp. 211–217. Springer (2012)
9. Berry, D.M., Kamsties, E., Krieger, M.M.: From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. Tech. rep., School of Computer Science, University of Waterloo, ON, Canada (2001)
10. Btoush, E.S., Hammad, M.M.: Generating ER Diagrams from Requirement Specifications Based on Natural Language Processing. International Journal of Database Theory and Application **8**(2), 61–70 (2015)
11. Chen, P.P.: Entity-Relationship Diagrams and English Sentence Structure. In: Proceedings of the International Conference on the Entity-Relationship Approach to Systems Analysis and Design, pp. 13–14 (1983)
12. Chi, E.H.H.: A Taxonomy of Visualization Techniques using the Data State Reference Model. In: Proceedings of the IEEE Information Visualization Conference (InfoVis), pp. 69–75. IEEE (2000)
13. Choma, J., Zaina, L.A.M., Beraldo, D.: UserX Story: Incorporating UX Aspects into User Stories Elaboration, pp. 131–140. Springer (2016)
14. Cleland-Huang, J.: Traceability in Agile Projects. In: J. Cleland-Huang, O. Gotel, A. Zisman (eds.) Software and Systems Traceability, pp. 265–275. Springer (2012)
15. Cohn, M.: User Stories Applied: for Agile Software Development. Addison Wesley Professional, Redwood City, CA, USA (2004)
16. Cooper Jr, J.R., Lee, S.W., Gandhi, R.A., Gotel, O.: Requirements Engineering Visualization: A Survey on the State-of-the-art. In: Proceedings of the International Workshop on Requirements Engineering Visualization (REV), pp. 46–55 (2009)
17. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. CoRR **abs/1605.07767** (2016). URL `http://arxiv.org/abs/1605.07767`
18. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do Practitioners Use Conceptual Modeling in Practice? Data & Knowledge Engineering **58**(3), 358–380 (2006)
19. Dimitrijević, S., Jovanović, J., Devedžić, V.: A Comparative Study of Software Tools for User Story Management. Information and Software Technology **57**, 352–368 (2015)
20. Du, S., Metzler, D.P.: An Automated Multi-component Approach to Extracting Entity Relationships from Database Requirement Specification Documents. In: Proceedings of the International Conference on Applications of Natural Language to Information Systems (NLDB), *LNCS*, vol. 3999, pp. 1–11. Springer (2006)
21. Dudáš, M., Zamazal, O., Svátek, V.: Roadmapping and navigating in the ontology visualization landscape. In: K. Janowicz, S. Schlobach, P. Lambrix, E. Hyvönen (eds.) Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW), pp. 137–152. Springer International Publishing (2014)
22. Fettke, P.: How Conceptual Modeling is Used. Communications of the Association for Information Systems **25**(1), 43 (2009)
23. Gagné, C.L.: Lexical and Relational Influences on the Processing of Novel Compounds. Brain and Language **81**(1–3), 723–735 (2002)

24. Genon, N., Heymans, P., Amyot, D.: Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In: Proceedings of the ACM SIGPLAN International Conference on Software Language Engineering (SLE), pp. 377–396. Springer (2010)
25. Goldberg, Y., Levy, O.: word2vec Explained: Deriving Mikolov et al.'s Negative-sampling Word-embedding Method. arXiv preprint arXiv:1402.3722 (2014)
26. Gulden, J., Reijers, H.A.: Toward Advanced Visualization Techniques for Conceptual Modeling. In: Proceedings of the Forum at the International Conference on Advanced Information Systems Engineering (CAiSE), pp. 33–40 (2015). URL `http://ceur-ws.org/Vol-1367/#paper-05`
27. Harmain, H., Gaizauskas, R.: CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. Automated Software Engineering **10**(2), 157–181 (2003)
28. Hartmann, S., Link, S.: English Sentence Structures and EER Modeling. In: Proceedings of the Asia-Pacific Conference on Conceptual Modelling (APCCM), pp. 27–35 (2007)
29. Holbrook III, H.: A scenario-based methodology for conducting requirements elicitation. SIGSOFT Software Engineering Notes **15**(1), 95–104 (1990)
30. Imaz, M., Benyon, C.: How Stories Capture Interactions. In: Proceedings of the IFIP International Conference on Human-Computer Interaction (INTERACT), pp. 321–328 (1999)
31. Kassab, M.: The Changing Landscape of Requirements Engineering Practices over the Past Decade. In: Proceedings of the International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 1–8. IEEE (2015)
32. Kassab, M., Neill, C., Laplante, P.: State of Practice in Requirements Engineering: Contemporary Data. Innovations in Systems and Software Engineering **10**(4), 235–241 (2014)
33. Lapata, M.: The Disambiguation of Nominalizations. Computational Linguistics **28**(3), 357–388 (2002)
34. Li, W., Zhang, X., Niu, C., Jiang, Y., Srihari, R.: An Expert Lexicon Approach to Identifying English Phrasal Verbs. In: Proceedings of the Meeting of the Association for Computational Linguistics (ACL), pp. 513–520 (2003)
35. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based Visualization of Ontologies. In: Proceedings of EKAW Satellite Events, *LNAI*, vol. 8982, pp. 154–158. Springer (2015)
36. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Forging High-Quality User Stories: Towards a Discipline for Agile Requirements. In: Proceedings of the International Requirements Engineering Conference (RE), pp. 126–135. IEEE (2015)
37. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Improving Agile Requirements: The Quality User Story Framework and Tool. Requirements Engineering **21**(3), 383–403 (2016)
38. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: The Use and Effectiveness of User Stories in Practice. In: Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), *LNCS*, vol. 9619, pp. 205–222. Springer (2016)
39. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Visualizing User Story Requirements at Multiple Granularity Levels via Semantic Relatedness, pp. 463–478. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-46397-1_35. URL `http://dx.doi.org/10.1007/978-3-319-46397-1_35`
40. Manning, C.D.: Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?, pp. 171–189. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
41. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to information retrieval. Cambridge university press Cambridge (2008)
42. Mesquita, R., Jaqueira, A., Agra, C., Lucena, M., Alencar, F.: US2StarTool: Generating *i** Models from User Stories. In: Proceedings of the International i* Workshop (iStar) (2015)
43. Meziane, F., Vadera, S.: Obtaining E-R Diagrams Semi-Automatically from Natural Language Specifications. In: Proceedings of the International Conference on Enterprise Information Systems (ICEIS), pp. 638–642 (2004)
44. Mich, L.: NL-OOPS: From Natural Language to Object Oriented Requirements Using the Natural Language Processing System LOLITA. Natural Language Engineering **2**, 161–187 (1996)
45. Moody, D.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering **35**(6), 756–779 (2009)
46. Moody, D.L., Heymans, P., Matulevičius, R.: Visual Syntax does Matter: Improving the Cognitive Effectiveness of the i* Visual Notation. Requirements Engineering **15**(2), 141–175 (2010)
47. Neill, C.J., Laplante, P.A.: Requirements Engineering: The State of the Practice. IEEE Software **20**(6), 40 (2003)
48. Omar, N., Hanna, J., McKevitt, P.: Heuristics-Based Entity-Relationship Modelling through Natural Language Processing. In: Proceedings of the Irish Conference on Artificial Intelligence & Cognitive Science (AICS), pp. 302–313 (2004)
49. Overmyer, S.P., Lavoie, B., Rambow, O.: Conceptual Modeling through Linguistic Analysis Using LIDA. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 401–410. IEEE Computer Society (2001)
50. Popescu, D., Rugaber, S., Medvidovic, N., Berry, D.M.: Reducing Ambiguities in Requirements Specifications Via Automatically Created Object-Oriented Models. In: Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs, *LNCS*, vol. 5320, pp. 103–124. Springer (2008)
51. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual Requirements Analytics: A Framework and Case Study. Requirements Engineering **19**(3), 257–279 (2014)
52. Rees, M.: A Feasible User Story Tool for Agile Software Development? In: Proceedings of the Asia-Pacific Software Engineering Conference (APSEC), pp. 22–30 (2002)
53. Robeer, M., Lucassen, G., Van der Werf, J., Dalpiaz, F., Brinkkemper, S.: Automated Extraction of Conceptual Models from User Stories via NLP. In: Proceedings of the International Requirements Engineering Conference (RE). IEEE (2016)
54. Rubin, E., Rubin, H.: Supporting Agile Software Development through Active Documentation. Requirements Engineering **16**(2), 117–132 (2010)
55. Ryan, K.: The Role of Natural Language in Requirements Engineering. In: Proceedings of the IEEE International Symposium on Requirements Engineering (ISRE), pp. 240–242. IEEE (1993)
56. Saeki, M., Horai, H., Enomoto, H.: Software Development Process from Natural Language Specification. In: Proceedings of the International Conference on Software Engineering (ICSE), pp. 64–73. ACM (1989)

57. Sagar, V.B.R.V., Abirami, S.: Conceptual Modeling of Natural Language Functional Requirements. Journal of Systems and Software **88**, 25–41 (2014)
58. Shneiderman, B.: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: Proceedings of the IEEE Symposium on Visual Languages (VL), pp. 336–343 (1996)
59. Soares, H.F., Alves, N.S.R., Mendes, T.S., Mendonça, M., Spínola, R.O.: Investigating the Link between User Stories and Documentation Debt on Software Projects. In: Proceedings of the International Conference on Information Technology - New Generations (ITNG), pp. 385–390 (2015)
60. Tjoa, A.M., Berger, L.: Transformation of Requirement Specifications Expressed in Natural Language into an EER Model. In: Proceedings of the International Conference on Conceptual Modeling (ER), *LNCS*, vol. 823, pp. 206–217. Springer (1993)
61. Trkman, M., Mendling, J., Krisper, M.: Using Business Process Models to better Understand the Dependencies among User Stories. Information and Software Technology **71**, 58 – 76 (2016)
62. Vela, M., Declerck, T.: A Methodology for Ontology Learning: Deriving Ontology Schema Components from Unstructured Text. In: Proceedings of the Workshop on Semantic Authoring, Annotation and Knowledge Markup (SAAKM), pp. 22–26 (2009)
63. Wang, J., Wang, Q.: Analyzing and Predicting Software Integration Bugs Using Network Analysis on Requirements Dependency Network. Requirements Engineering **21**(2) (2016)
64. Wang, X., Zhao, L., Wang, Y., Sun, J.: The Role of Requirements Engineering Practices in Agile Development: An Empirical Study. In: Proceedings of the Asia Pacific Requirements Engineering Symposium (APRES), vol. 432, pp. 195–209 (2014)
65. Wautelet, Y., Heng, S., Hintea, D., Kolp, M., Poelmans, S.: Bridging User Story Sets with the Use Case Model. In: S. Link, J.C. Trujillo (eds.) Proceedings of ER Workshops, pp. 127–138 (2016)
66. Wautelet, Y., Heng, S., Kolp, M., Mirbel, I.: Unifying and Extending User Story Models. In: Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), *LNCS*, vol. 8484, pp. 211–225. Springer (2014)
67. Wautelet, Y., Heng, S., Kolp, M., Scharff, C.: Towards an Agent-driven Software Architecture Aligned with User Stories. In: Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART), pp. 337–345 (2016)
68. Yu, E.S.K.: Modelling strategic relationships for process reengineering. Ph.D. thesis, University of Toronto (1996)
69. Yue, T., Briand, L.C., Labiche, Y.: A Systematic Review of Transformation Approaches between User Requirements and Analysis Models. Requirements Engineering **16**(2), 75–99 (2010)
70. Yue, T., Briand, L.C., Labiche, Y.: aToucan: An Automated Framework to Derive UML Analysis Models from Use Case Models. ACM Transactions on Software Engineering and Methodology **24**(3), 13:1–13:52 (2015)