Assessing the quality of the requirements specification by applying GQM approach and using NLP tools

Timoshchuk Evgenii

Abstract — Software requirements are quite difficult to measure in terms of quality without reviews and subjective opinions of stakeholders. Quality assessment of specifications in an automated way saves project resources and prevents future latent defects in software. Requirements quality can be evaluated based on a huge variety of attributes, but their meaning is quite vague without any mapping to specific measurement metrics. Application of goal-question-metric (GQM) approach in the quality model helps to choose the most important quality attributes and create a mapping with metrics, which can be collected and calculated automatically. Text of software requirements written in natural language can be analyzed by NLP tools due to identify weak signle words and phrases, which make statements ambiguous. Metrics for such quality attributes as ambiguity, singularity, subjectivity, completeness, and readability are proposed in this work. The quality model was implemented in a prototype by adopting natural language processing techniques for requirements written in the Russian language with the support of external API.

Index Terms — requirements quality, GQM approach, quality assessment, Natural Language Processing

1 Introduction

Low quality of requirements may cause expensive consequences during the software development lifecycle. Issues in requirements, such as ambiguity and incompleteness may lead to time and cost overrun in the project. Especially, if iterations are long and feedback comes too late – the faster a problem is found, the cheaper it is to fix. However, it is not so easy to properly detect in automated way whether a requirements specification has lack of clarity. Some of these issues require specific domain knowledge to be uncovered. For example, it is very difficult to detect with automatic approaches whether a requirements specification is lacking necessary features.

There are a variety of requirements management techniques, tools, and practices in the software development field. However, they should be tailored to the choosen development methods. Requirements engineering assumes that the requirements must meet a number of criteria. Descriptions of such criteria can be found in both the scientific and methodological literature and put on standards. For instance, the IEEE standard [1] for requirements engineering defines quality attributes for a single requirement: necessary, appropriate, feasible, verifiable, correct, conforming, complete, consistent, comprehensible etc. Several language criteria are also defined for the text of requirements. Unbounded and ambiguous terms should be avoided. Requirements should state 'what' is needed, not 'how'.

Despite the exact techniques on how to gather and validate this requirements quality metrics are not formulated by the standard (which is obviously beyond its area of consideration), but they are the topic for various researches.

Software requirements in industry are most ofthen written in natural language which has no any formal semantics. This is the main reason why issues in requirements are so hard to detect. Approach that is presented in this paper faces the problem of fast feedback and getting some knowledge about specification's semantics with concrete symptoms for a requirement artefact's quality defect.

Natural language processing (NLP) tools and systems have been applied to analysing requirements texts since the 1980's [29]. More and more NLP systems and tools with applying requirements evaluation are developed in recent years. It is a most appropriate technique to analyse human text and collect some useful data about it.

Goal-question-metric (GQM) involves defining achievable goals in order to attain quality thereby providing questions in relation to how to achieve the goals and metrics are provided to ascertain the progress in attaining set goals. This research work makes use of GQM by setting goals such as unambiguity, completeness, readability that the requirements must meet, questions on how to derive these quality attributes, and what to measure in determining if our requirements match defined goals.

The ultimate goal of this work was encapsulating the best of these techniques and methods for measurement requirement quality into a single model and provide a prototype of tool for automated validation of real-world requirements against it with Russian language support. This paper will present some measuring quality indicators for natural language requirements presented in textual natural format. Identified quality indicators first of all should point out concrete defects and provide suggestions for improvements. Proposed tool prototype combined all this indicators and computes quality measures in a fully automated way.

This paper describes a workflow of a model that helps to obtain low-level quality indicators based on some metrics of textual requirements, such as text length, number of ambiguous terms, imperative verbal forms, etc. This model has been implemented in a tool that computes quality metrics in a fully automated way.

2 Related studies

Many publications discuss different problems in requirements specifications. First problem is automatization of assessment process. Mostly researchers focus on approaches for automated detection of specific defects in requirements specifications without any additional interaction with user. The main features of such tools include detection of similarity in requirements [15] and ambiguity [16], detection of missing information, linguistic flaws and passive sentences [17].

K¨orner and Brumm presented RESI, a tool that scans documents for linguistic flaws and reports them to the user (see Section II-C). It can be used to detect defects in requirements specifications, but the high number of false positives results prohibits the actual use of this tool in a real case [18].

Fabbrini et al. presented QuARS (Quality Analyzer for Requirement Specifications) tool that checks requirements specification by comparing with predefined word lists [16]. The lists give indicators for problems and if the number of indicators in the phrase exceeds a given threshold, requirements is ambigious.

Verma et al. presented their RAT (Requirements Analysis Tool) [19] tool - a word processor that able to analyze natural language requirements based on a user-defined glossary and constrained language. RAT highlights problematic requirements directly in the requirements specifications, but this process requires some training by real users.

Goldin and Berry [24] implemented a tool called Abstfinder to identify the abstractions from natural language text used for requirements elicitation.

Lee and Bryant [25] developed an automated system to assist the engineers to build a formal representation from informal requirements.

Overhelming majority of authors suppose that ambiguity carries a high risk of misunderstanding among different readers. Several studies dealing with ambiguity identification have aimed to help improve the quality of requirements documents. Some tools have been developed specifically to detect, measure and reduce possible structural ambiguities in text.

In paper Yang H, Willis A, De Roeck A and Nuseibeh B describe an automated approach for characterizing and detecting ambiguities that was implemented in NAI (Nocuous Ambiguity Identification) tool prototype [22]. Implemented tool uses machine learning algorithm to determine whether an ambiguous sentence is nocuous or innocuous, based on a set of heuristics that draw on human judgments, which we collected as training data. The tool focuses on coordination ambiguity.

Kamsties et al. [26] described pattern-driven inspection technique to detect ambiguities in requirements.

Mich and Garigliano [27] investigate the use of a set of ambiguity for the measurement in syntactic and semantic ambiguity, which is implemented in tool LOLITA using NLP algorithms.

Kiyavitskaya et al. [28] proposed a two-step approach in which lexical and syntactic analysys was performed to identify ambiguity. An automated tool was implemented to measure what is potentially might be ambiguous specifically for each sentence.

Another important proplem related to specification quality assesment is a choice of correct criteria for overall evaluation. Davis et al. evaluated 24 criteria and metrics for determination of the overall requirements specification quality [20]. Some of the criteria may affect and even contradict each other. Therefore, the authors made a conclusion that a perfect requirements specification does not exist. Another approach was proposed by Wilson et al. – he counted the occurrences of certain expressions in a document to evaluate its quality [21] with indicators that include completeness and consistency. This group of researches developed a tool that focus on a broader understanding of requirements quality, instead of just a single aspect. Implemented ARM tool is based on the IEEE 830 standard and aims at developing metrics for requirements quality.

Ambriola and Gervasi [23] developed a web-based NLP tool, called Circe, which was designed to facilitate the gathering, elicitation, selection, and validation of requirements.

Unlike other related works show attempts to evaluate requirements in automatically by only one quality criteria, current paper describes an approach to identify several correct quality attributes with correlated metrics to measure, combining them into one overall evaluation in an automated way. Moreover, all the mentioned-above tools support only English language and don't support requirements written in Russian.

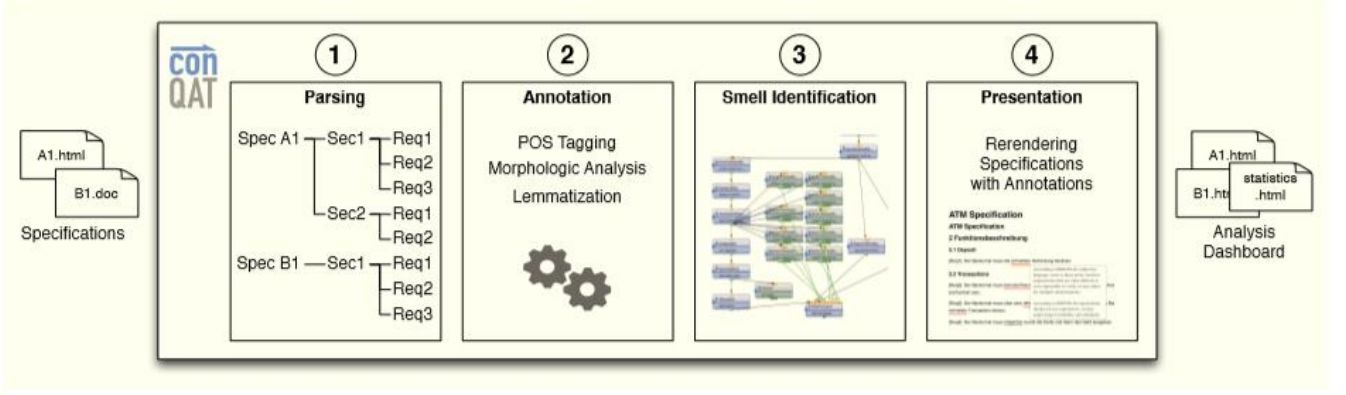

Fig. 1. Example of requirement analysis workflow for automated tool conQAT [30]

3 GQM APPROACH

The Goal-Question-Metric is a method based on system of questions and simple answers about properties evaluation [3]. This approach consists of three main steps: specifying goals, pointing relevant attributes and providing measurements. GQM framework helped to define appropriate metrics and estimate the quality of requirements in current case. The goal should be defined for an object, with a purpose, from a perspective, in an environment. The overall goal of current project it to measure quality of requirements and it can be formulated by following template:

Analyze requirement quality
for the purpose of improving
with respect to quality attributes
from the viewpoint of project managers
in the context of product development.

In addition, several sub-goals were identified, which should be fulfilled to achieve the main goal. For instance:

Sub-goal: Analyze requirement unambiguity for the purpose of improving with re-spect to quality attributes from the viewpoint of project managers in the context of product development.
Question: How many vague words and weak phrases make requirement ambiguous?
Metric: Number of ambiguous words in 1 requirement divided by an average number of words in 1 requirement.

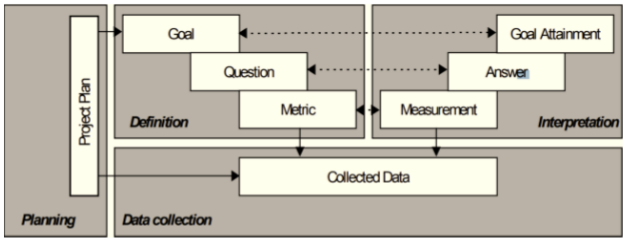

Fig. 2. Phases of Goal/Question/Metric approach [4]

In this approach identification of the questions and metrics allows to properly clarify the goals in order to achieve the transparency and propose how and why the goals are supposed to be achieved. Clarifiacation becomes more concrete during the movement from top level to bottom and helps to avoid abstract unreal goals.

GQM approach is supported by several specific methodological phases [Fig.1] [4]:

- Definition - goals, questions, metrics and hypotheses are defined and documented. Main attributes, formulas and measurement approaches and exact metrics are defined.
- Data Collection - searching and counting for ambiguous words and other quality indicators in source text.
- Interpretation - collected data is processed into quality measurement results, that provides answers on defined questions to reach the goal.

GQM approach in current case consists of 5 main steps [4]:

- Business goal setting. The main purpose of this case is automated evaluating quality of software requirements by range of attributes.
- Generating questions. Breaking down goals into components, defining them in a quantifiable way, f.e.: "How much should the proposal structure be complied with so that the requirement has the quality property of complete-ness"?
- Specifying measures. Detecting metrics that should be collected to answer questions, f.e.: "Percentage of matching requirement sentence structure template".
- Defining mechanism of data collection. Measures are collected by semantic and syntax text analysis based on matching words with predefined diction-aries.
- Gathering and analyzing of collected data. Calculated metrics should be interpreted into quality estimation for each requirement and overall Software Requirements Specification (SRS).

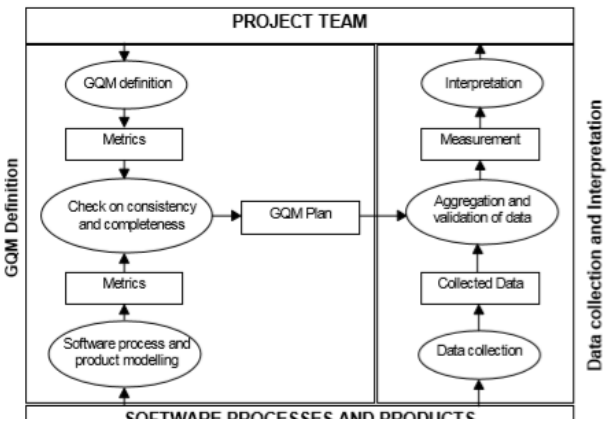

Fig. 3. Process of collecting metrics and measurements [4]

Process of measuring quality in software development has it's certain difficulties. In order to understand the effects of actions that are implemented in software development and gain the understanding of how the improvements can be made for a future process a certain purpose should be put in measurement process. The purpose may be:

1. Understanding of the product requirements. Correct measurements will allow to see the graphical or mathematical representation of a requirement elicitation process, whether it will be a time spend on describing every feature.

2. Controlling the product requirements. While having a graphical representation of the SRS document, the relations between different requirements and user actions can be identified, which further would allow to control the impact on the development process in total.

3. Improving the product requirements. can be achieved after the control of the development processes is gained. Certain improving effect can be applied to processes, variables and their relationships.

Metrics for the requirements should allow to determine their quality for the current development process and to represent collected resulting data in a graphical way.

4 QUALITY ATTRIBUTES

Many authors in their methodologies have already defined the key interdependent [Fig. 4] quality attributes [6].

- Validity – the clients should be able to validate (confirm) the requirement according to their needs.
- Verifiability – the engineer must be able to verify that the system-to-be meets the specified system.
- Modifiability – requirements must be able modifiable with ease for maintenance.
- Completeness – all client's needs must be covered.
- Consistency – should not be any contradiction among requirement.
- Understandability – the requirements are correctly understood without difficulty.
- Unambiguity – there exists only one interpretation of the requirement (no ambiguous words in the requirement sentence).
- Traceability – there exists an explicit relationship of each requirement with design, implementation and testing artefacts.
- Singularity – each requirement is clearly determined and identified, without mixing it with other requirements.

Mentioned-above attributes come out from following types of unbounded or ambiguous terms that should be avoided according to the standard:

- superlatives ('best', 'most');
- subjective language ('user friendly', 'easy to use', 'cost-effective');
- vague pronouns ('it', 'this', 'that');
- ambiguous terms such as adverbs and adjectives ('almost always', 'significant', 'minimal') and ambiguous logical statements ('or', 'and/or');
- open-ended, non-verifiable terms (such as 'provide support', 'but not limited to', 'as a minimum');
- comparative phrases ('better than', 'higher quality');
- loopholes ('if possible', 'as appropriate', 'as applicable');
- terms that imply totality ('all', 'always', 'never', and 'every');

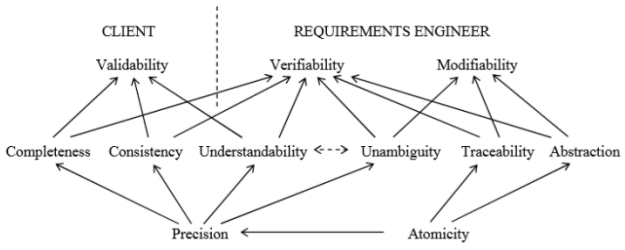

Fig. 4. Dependencies between of qualitative attributes [6]

Unambiguity. It requires that only one semantic interpretation of the requirement exists. To evaluate the ambiguity of each requirement, we propose to use dictionaries with a set of words, which indicates ambiguity in the requirement [13][14]. There are several types of them:

Type	English examples	Russian examples
Quality	'best', 'most', 'appropriate', 'adequate'	'лучший', 'самый', 'подходящий', 'адекватный'
Subjectivity	'user-friendly', 'easy to use', 'rational'	'легкий в использовании'
Quantity	'about', 'significant', 'minimal', 'few', 'all', 'each', 'every', 'any', 'few', 'little', 'many', 'much', 'several', 'some'	'примерно', 'несколько', 'немного'
Frequency	'almost always', 'usually', 'as a rule', 'never'	'почти всегда', 'обычно', 'как правило'
Persistency	'long', 'longer', 'durable', 'momentary'	'долго', 'больше', 'прочный', 'сиюминутного'
Probability	'probably', 'possibly', 'if it possible', 'unlikely'	'возможно', 'если это возможно', 'вряд ли'
Open listings	'etc.', 'and so forth', 'and so on'	'и так далее'
Position / size	'close', 'bigger', 'tall', 'far', 'short', 'small', 'huge'	'близко', 'больше', 'падение', 'далеко', 'короткий', 'небольшой', 'огромный'
Comparative phrases	'better than', 'higher quality', 'same as'	'лучше, чем', 'выше качество', 'то же, что'
Loopholes	'if possible', 'as appropriate', 'as applicable'	'если возможно', 'соответствующие', 'в качестве применимого'
Weak adverb or adjective	'as desired', 'at last', 'either', 'eventually', 'if appropriate', 'if desired', 'in case of', 'if necessary'	'по желанию', 'наконец', 'либо', 'в конечном счете', 'если уместно', 'если желательно', 'в случае', 'если необходимо'

- Connection words dictionary – ("and", "or", "but", "however", "otherwise", "even", "although", etc.) the usage of such words not a problem in itself, but their too frequent use leads to a decrease in the quality of requirements, especially in terms of uniqueness and ambiguity.
- Negative adverbs dictionary – the negative particle is the word not used to indicate negation, denial, refusal, or prohibition. Repeated use of such words makes a sentence difficult to understand and decrease the ambiguity of the requirement.
- Anaphoric expressions dictionary – the use of expressions, the interpretation of which depends on other expressions previously encountered in the text, for example: "which", "he", "she", "it", "they", "where", "this", "that", etc. Requirements containing anaphora usually do not have characteristics of clarity and unambiguity.
- Undefined terms dictionary – In addition to connective words, the quality of requirements is significantly affected by the use of vague terms that lead to ambiguity.

As the metric for assessing ambiguity, was used the following formula:

$$Unambiguity\ \% = \left(1 - \frac{N_{ambig}}{N_{total}}\right) \times 100$$

Where N_{ambig} – the number of words in the requirement, N_{ambig} – the number of ambiguous words in the requirement.

Singularity. Statement of the requirement must relate to only one unique requirement that does not overlap with others. The presence of several modal words tells us that the requirement contains several meanings and that the statement does not have the characteristic of singularity. These words may include could, may, might, can, should, will, shall, must, would, etc. The number of connective words may also indicate the presence of several requirements within one (mentioned above). As the metric for assessing singularity, was used the following formula:

$$Singularity\ \% = \left(1 - \frac{(N_{modal} - 1) + N_{connective}}{N_{total}}\right) \times 100$$

where N_{total} – the number of words in the requirement, N_{modal} – the number of modal verbs which are not zero, $N_{connective}$– the number of connective words in the requirement.

Subjectivity. This attribute indicates the presence of perspectives, feelings, or opinions entering the decision-making process. The leading causes of subjectivity in requirements can be:

- dangerous plural with ambiguous reference,
- combination of "and" and "or" that leads to unclear associativity,
- unclear inclusion,
- passive voice,
- imprecise and inside behavior,
- negative or too broad reason.

Readability. This attribute indicates how easily requirement text can be read and understood, it can be based on the number of syllables per word and number of words per sentence. It can be calculated by Flesch-Kincaid Grade Level [37], Coleman-Liau Grade Level [38], and Smog Grade [39]. The second one was chosen:

$$Redabiliti\ CLI = 0.0588\,L - 0.296S - 15.8$$

where L – average number of letters per 100 words, S – average number of sentences per 100 words. If CLI is around 10, text is easy to read, but if CLI > 15 text is too difficult for understanding. A mapping into percentage interpretation was made (if CLI index is more than 17.5, than readability is 0%) by following formula:

$$Readability\ \% = \left(1 - \frac{|\ CLI - 12.5\ |}{5}\right) \times 100$$

Completeness. It requires that the requirement contain all necessary elements, including constraints and conditions, to enable the requirement to be implemented [18].

Example of structure template in Table 1 for this requirement:

"In the Combat Zone, an HQ Switch, which is identical to a trunk node switch, shall be given two independent links to at least two other nodes in the network".

Element	Text
Actor	an HQ switch
Conditions of Action	in the Combat Zone
Action	two independent links
Object of action	-
Constraints of action	To at least two other nodes in the net
Source of Object	To at least two other nodes in the net
Destination of Action	Which identical to a trunk node switch

Table. 1. Structural template for completeness

Completeness quality attribute was calculated by this formula:

$$Completeness\ \% = \frac{N_{filled}}{N_{total}} \times 100$$

where N_{total} – the number of elements in the structural template, N_{filled} – the number of elements from template that were identified in requirement sentence.

5 NATURAL LANGUAGE PROCESSING

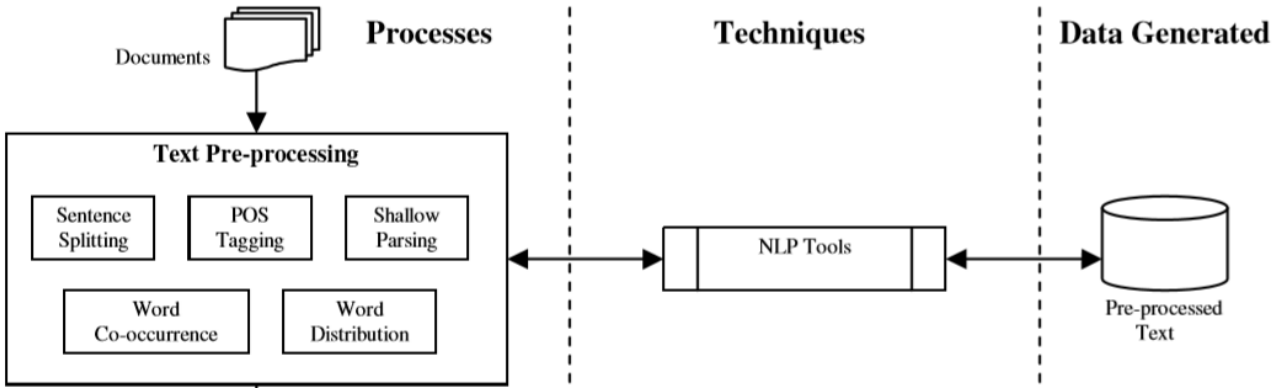

Fig. 5. NLP workflow for requirement analysis

Natural Language Processing is a field of computer science and computational linguistics that aims to analyze linguistic data from input using computational methods and techniques [33]. The natural language is very complicated, it is subject to syntactic and semantic rules. It studies the conceptual dimension that refers to "pragmatic" actions which are intended. The syntactic rules describe the major pattern of a sentence such as nouns, adjectives, and verbs [34]. The semantic rules refer to the meaning of each word in the sentence and relation between words when they are combined, which is "compositional semantic" [35].

Natural language texts are used to be analyzed in a sequential process. This process starts with lexical and structural elements. For its purpose text should be parsed in a search of the most suitable syntax tree. After that some complex techniques are applied for interpretation of the semantic content due to meaning understanding. Of course, such analysis does not allow to understand fully the content and get independent meaning of the sentence without any discrepancies.

Several techniques were used in current model workflow: splitting sentence in syntax tree, part-of-speech tagging, morphological analysis and calculationg word distribution and co-occurance by redefined dictionaries [Fig. 5]. Inputed text document with requirements should go through several text preprocessing steps: sentence splitting, POS-tagging, and phrase-based shallow parsing.

Sentence splitting. At first, the text is splitted into a set of sentences by using a sentence boundary detector.

POS-tagging. Then, for each requirement sentence, the parser based on individual words and associated phrase information that used to obtain word lemma and POS tags such as noun, verb, adjective, adverb, etc. In current model the Stanford NLP library [42] was used for this. POS tagging helped to determine so-called substituting pronouns. A detailed description of POS tagging technical details is beyond the scope of this paper, but can be found, for example, in [41]. Given a sentence in natural language text, it determines the role and function of each single word in the sentence. The output is usually a so-called tag for each word, e.g. whether a word is an adjective, a particle or a possessive pronoun. These are pronouns that do not repeat the original noun and, thus, need a human's interpretation of its dependency. A syntax tree shows the main structure of the sentence [Fig. 6], where tree's leafs are the words of the sentence and inner nodes express the sentence's composition. In example "the channel selection" forms a nominal phrase (NP), as indicated by their common parent node NP. The additional information "of the headphones" is added as prepositional phrase (PP). The noun phrase and the prepositional phrase form a new nominal phrase, which is the object of the verb "changes".

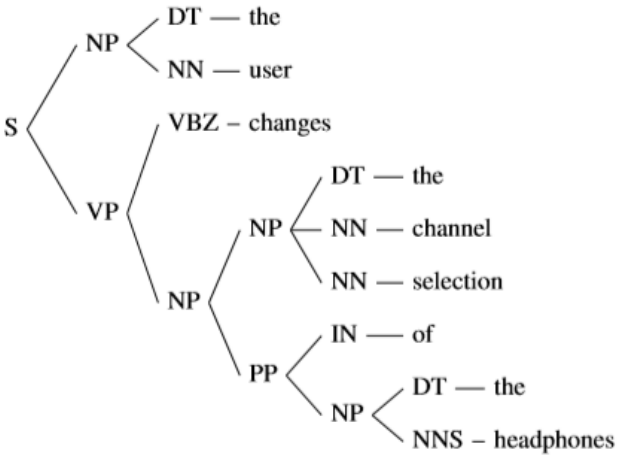

Fig. 6. Syntax tree for NLP workflow

Morphological Analysis. Based on POS-tagging more detailed analysis of text was performed that determines its inflection. This step contains identifying a verb's tense or an adjective's comparison. The main outcome of this step is analysis for usage of adverbs and adjectives in their comparative or superlative form.

Dictionaries. For describing different quality attributes were used several dictionatries with ambigious phrases and words based on quality standards and case study experience. Normalisation technique for dicitionry words called *lemmatisation* was applied, that reproduces the original form of a word. This technique is very similar to stemming, Porter Algorithm [40], that based on the POS tag as the word's morphological form instead of heuristics.

6 PROTOTYPE

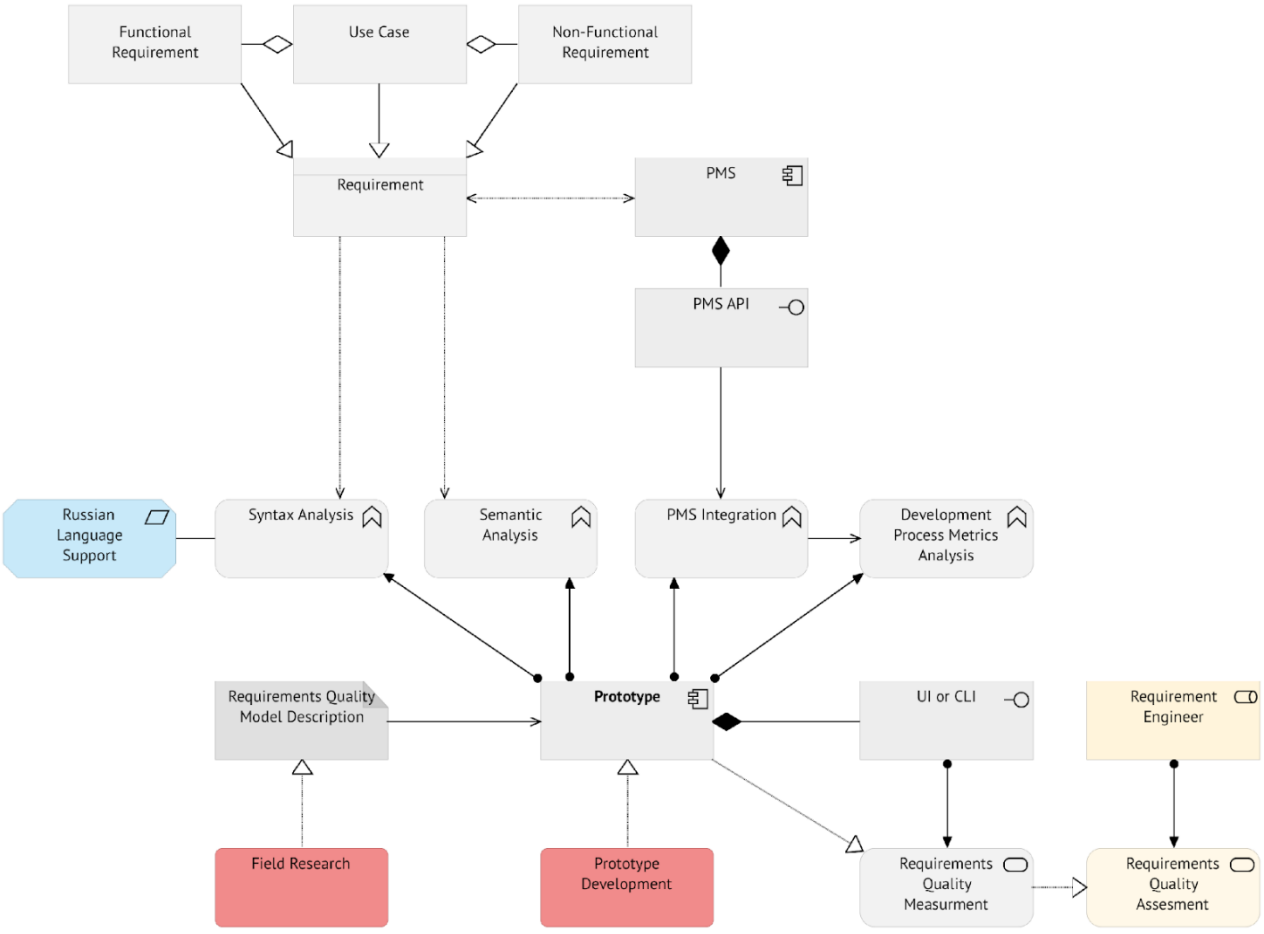

Fig. 7. Prototype scope in ArhiMate [36] notation

To fully support the extraction of metrics for all beforementioned quality attributes, the prototype should have several features [Fig. 7].

The prototype is a software tool which main goal is to perform requirements quality measurement. Requirements can be of any type expressed in the text form: functional, non-functional, use-cases. The prototype is able to perform several functions:

– Integration with project management system to gather textual requirements from it (via is API).

– Perform syntax and semantic analysis of said requirements (supporting Russian language [13][14]).

The core of the prototype is the Requirement Quality Model which contains a consistent set of requirements quality metrics and is expressed in algorithms on how to measure these metrics and how to draw conclusions (average quality of a requirement/set of requirements). The prototype provides a requirement engineer with a graphical user interface or command-line interface to obtain the results of requirements measurement.

For NLP were used custom analogues of Python libraries Wordnet [31] and Spacy [32] with Russian language support. Analysys of ambiguity was implemented based on open-source microservice OpenReqEU.

To get results of the requirements analysis the prototype provides the requirements engineer with the either graphical or command line interface. Here are some of the interface functions that are available:

- List all the requirements;
- Show quality metrics of the specific requirement;
- Show quality metric for all requirements analysed.

The dictionaries of ambigious words were translated into Russian language. Docker container for OpenReqEU microservice was rebuild and used as external API for further process of quality assessment. All ambigius words in requirements highlighted according to their category after service finished its work.

On the next step one more external API was used for evaluating readability indexes by service *readability.io* – text of every requirement was uploaded and resulting number was recieved from website.

For graphical representation of evaluated quality results Visual Paradigm Diagram was used. Spider graph and stacked histogram were chosen as the most appropriate visualisaition of collected data. All the metrics that were calculated in prototype automatically synchronized with Visual Paradigm Service and published as a web-dashboard.

7 Results

After implementing the proposed solution on requirements, it was tested on the sample requirement text. As a result, the following distribution of weak words was got:

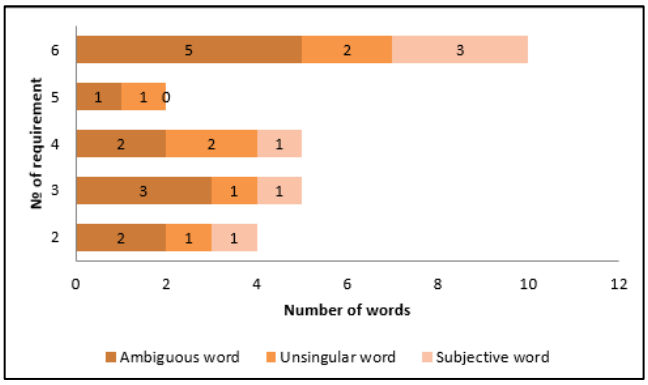

Fig. 8. Number of weak words per requirement.

These weak words were highlighted in GUI and classified by different types of ambiguity:

REQ-2. Пользователь может выбрать любую новость компании для чтения и пр окомментировать ее по желанию.

REQ-3. Пользователь нажимает красную кнопку в углу формы и в конечном итог е переходит на следующий этап выбора товара в интернет-магазине.

REQ-4. "Система должна принимать пароль состоящий как минимум из 8 символ ов, но необязательно содержащий много цифр."

REQ-5. "Поиск в системе должен осуществляться предпочтительно по имени, ф амилии пользователя."

REQ-6. "Любой Пользователь иногда может вводить код аэропорта, который должен определяться системой, но также иногда код аэропорта должен заменят ься названием ближайшего города, таким образом пользователю не требуется знат ь код аэропорта, но система по прежнему должна знать код аэропорта."

REQ-7. "Список доступных рейсов должен содержать информация о номере поле та, времени посадки и приземления, они должны быть отсортированы по цене."

REQ-8. "В системе должна быть предусмотрена возможность оперативно заброни ровать полет, купить билет, при необходимости - зарезервировать номер в отеле и ли арендовать машину."

REQ-9. "Программа должна работать насколько возможно быстро, все ошибки должны быть устранены."

Fig. 9. Hightlighted ambiguous words in every requirement

Final evaluation about overall quality was made:

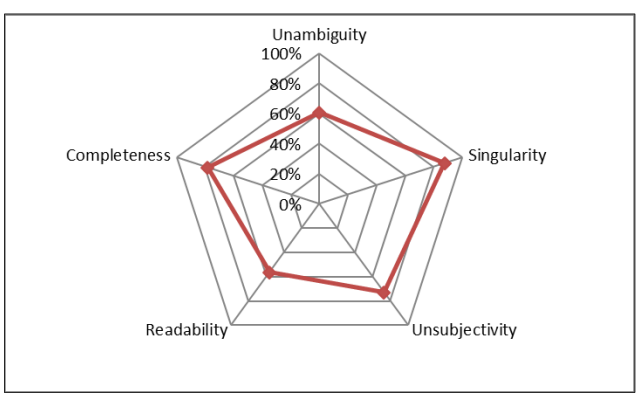

Fig. 10. Overall quality of all requirements by attributes

Conclusion and discussion

Natural language still prevails in the majority of requirement documents. Software engineers need ways to cope with the ambiguity inherent in natural language requirements. In order to minimize their side effects at the early stages of the software development lifecycle, it is important to develop scalable automated solution to detect potential nocuous ambiguities in natural language requirement specifications.

The usage of quality metrics in a software development lifecycle requires considering three important aspects. Firstly, obtaining all mentioned-above measurements by hand would be misleading, therefore automated tools become required. Secondly, an automated prototype implementation should avoid the refusal of requirements engineers – this tool is created due to help in improvement of requirements elicitation process, but not for punishment and identifying failures. Finally, decisions about which attributes and metrics to apply should be wisely and gradually made: “Not everything that can be counted counts, and not everything that counts can be counted” – Albert Enshtein.

Despite the fact that quantitative measurement is one of the foundations of modern empirical science, they should be used with caution and wisdom. Assessing the quality of requirements demands human judgment. This judgment can be assisted, but not replaced, by objective measurements. Automated tool that provides low-level quality indicators can provide valuable hints to improve high-level quality features of requirements.

In this paper an automated approach for characterizing and identifing potentially nocuous ambiguities was described. Given a natural language requirements document, ambiguous instances contained in the sentences were first extracted. Identified ambiguities can be the reason of misunderstanding among different readers. The implementation can be usable by requirements analysts and will allow them to experiment with iterative identification of potential ambiguity moments in requirement documents.

Acknowledgment

Author wish to thank the organizers of CASE in Tools International Hackathon and every single person who contributed to the success of this event. This research would not have been conducted without efforts of Konstantin Valeev, the challenge owner, who shed more light on grey areas of this project and provided this research with enough resources.

References

1. IEEE, “Systems and Software Engineering – Life Cycle Processes – Requirements Engineering,” ISO/IEC/IEEE 29148:2018(E) (2018)

2. Khurana, Diksha & Koli, Aditya & Khatter, Kiran & Singh, Sukhdev. (2017). Natural Language Processing: State of The Art, Current Trends and Challenges.

3. Basili, Victor; Gianluigi Caldiera; H. Dieter Rombach, The Goal Question Metric Approach, Basili,Victor; GianluigiCaldiera,1994

4. Solingen, Rini & Berghout, Egon. (1999). The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development.

5. Rosanez. (2017). A.Semi-Automatic Checklist-Based Quality Assessment of Natural Language Requirements. Campinas.

6. Génova, G., Fuentes, J. M., Llorens, J., Hurtado, O., & Moreno, V. (2011). A framework to measure and improve the quality of textual requirements. Requirements Engineering, 18(1), 25–41. doi:10.1007/s00766-011-0134-z

7. Daniel Jurafsky & James H. Martin. (2006). Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition.

8. Génova, Gonzalo & Fuentes, José M. & Llorens, Juan & Hurtado, Omar & Moreno, Valentín. (2011). A Framework to Measure and Improve the Quality of Textual Requirements. Requirements Engineering. 18. 10.1007/s00766-011-0134-z.

9. Bokhari, Mohammad & Siddiqui, Shams. (2011). Metrics for Requirements Engineering and Automated Requirements Tools.

10. Chantree, F. & Nuseibeh, B. & De Roeck, Anne & Willis, Alistair. (2006). Identifying Nocuous Ambiguities in Natural Language Requirements. Proceedings of 14th IEEE International Requirements Engineering Conference (RE'06). 59 - 68. 10.1109/RE.2006.31.

11. Massey, Aaron & Rutledge, Richard & Antón, Annie & Swire, Peter. (2014). Identifying and classifying ambiguity for regulatory requirements. 2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings. 83-92. 10.1109/RE.2014.6912250.

12. Completeness and Consistency Checking of System Requirements: An Expert Agent Approach,” Expert Systems with Applications 11, no. 3 (1996): 263–276

13. Кирилл Игоревич Гайдамака, “Характеристики и индикаторы качества требований для русскоязычной инженерной среды,” in Конференция «технологии разработки информационных систем» (ФГАОУ ВО "Южный федеральный университет", 2017)

14. Виктор Константинович Батоврин and Кирилл Игоревич Гайдамака, “Некоторые особенности оценки характеристик требований к системам,” Информатизация и связь, no. 4 (2017): 191–196.

15. Y. Pisan, “Extending requirement specifications using analogy,” in ICSE’00: 22nd Int. Conf. on Software Engineering. Limerick, Ireland: ACM, 2000, pp. 70–76.

16. F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, “The linguistic approach to the natural language requirements quality: Benefit of the use of an automatic tool,” in SEW’01: 26th Annual NASA Goddard Software Engineering Workshop. IEEE Computer Society, 2001, pp. 97–105.

17. N. Power, “Variety and quality in requirements documentation,” in REFSQ’01: 7th Int. Workshop on Requirements Engineering : Foundation for Software Quality, 2001, pp. 165–170.

18. S. J. K¨orner, M. Landh¨außer, and W. F. Tichy, “Transferring research into the real world: How to improve RE with AI in the automotive industry,” in AIRE’14: 1st Int. Workshop on Artificial Intelligence for Requirements Engineering, Aug. 2014

19. K. Verma and A. Kass, “Requirements analysis tool: A tool for automatically analyzing software requirements documents.” in Int. Semantic Web Conf., ser. Lecture Notes in Computer Science, A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, Eds., vol. 5318. Springer, Oct. 2008, pp. 751–763

20. A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, “Identifying and measuring quality in a software requirements specification,” in 1st Int. Software Metrics Symposium, May 1993, pp. 141–152

21. W. Wilson, L. Rosenberg, and L. Hyatt, “Automated analysis of requirement specifications,” in ICSE’97: 19th Int. Conf. on Software Engineering, May 1997, pp. 161–171

22. Yang H, Willis A, De Roeck A, Nuseibeh B. Automatic detection of nocuous coordination ambiguities in natural language requirements. InProceedings of the IEEE/ACM international conference on Automated software engineering 2010 Sep 20 (pp. 53-62). ACM.

23. Ambriola, V., and Gervasi, V. 1997. Processing natural language requirements. In Proceedings of the 12th international conference on Automated software engineering 36-45.

24. Goldin, L., and Berry, D. M. 1994. Abstfinder, a prototype abstraction finder for natural language text for use in requirements elicitation: design, methodology, and evaluation. In Proceedings of the First International Conference on Requirements Engineering, 18–22.

25. Lee, B. S., and Bryant, B. R. 2004. Automation of software system development using natural language processing and two-level grammar. Radical Innovations of Software and Systems Engineering in the Future. Springer, Heidelberg 219–233.

26. Kamsties, E., Berry, D., and Paech, B. 2001. Detecting ambiguities in requirements documents using inspections. In Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01), 68-80.

27. Mich, L., and Garigliano, R. 2000. Ambiguity measures in requirement engineering. In Proceedings of international conference on software—theory and practice (ICS2000), 39– 48.

28. Kiyavitskaya, N., Zeni, N., Mich, L., and Berry, D. M. 2008. Requirements for tools for ambiguity identification and measurement in

natural language requirements specifications. Requirements Engineering Journal 13, 207–240.

29. Abbot R. Program design by informal English descriptions. Comm. ACM 26(11): 882-894, 1983.

30. Femmer H, Fernández DM, Juergens E, Klose M, Zimmer I, Zimmer J. Rapid requirements checks with requirements smells: Two case studies. InProceedings of the 1st International Workshop on Rapid Continuous Software Engineering 2014 Jun 3 (pp. 10-19). ACM.

31. (2017). ru-wordnet. GitHub repository. Retrieved from https://github.com/jamsic/ru-wor

32. Baburov, Y. (2018). spacy-ru. GitHub repository. Retrieved from https://github.com/buriy/spacy-ru

33. Rosanez. (2017). A.Semi-Automatic Checklist-Based Quality Assessment of Natural Lan-guage Requirements. Campinas.

34. Génova, G., Fuentes, J. M., Llorens, J., Hurtado, O., & Moreno, V. (2011). A framework to measure and improve the quality of textual requirements. Requirements Engineering, 18(1), 25–41. doi:10.1007/s00766-011-0134-z

35. Daniel Jurafsky & James H. Martin. (2006). Speech and Language Processing: An introduc-tion to natural language processing, computational linguistics, and speech recognition.

36. Pubs.opengroup.org. (2019). *ArchiMate® 3.1 Specification*. [online] Available at: https://pubs.opengroup.org/architecture/archimate3-doc/ [Accessed 20 Nov. 2019].

37. Kincaid, J.P., Fishburne, R.P., Rogers, R.L., & Chissom, B.S. (1975). Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formu-la) for Navy enlisted personnel. Research Branch Report 8–75. Chief of Naval Technical Training: Naval Air Station Memphis.

38. Coleman, Meri; and Liau, T. L. (1975); A computer readability formula designed for ma-chine scoring, Journal of Applied Psychology, Vol. 60, pp. 283–284

39. McLaughlin, G. Harry (May 1969). "SMOG Grading — a New Readability Formula" (PDF). Journal of Reading. 12 (8): 639–646

40. M. Porter. An algorithm for suffix stripping. Program: electronic library and information systems, 198

41. D. Jurafsky and J. H. Martin. Speech and Language Processing. Pearson Education, 2014

42. K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2003.