

Quality Measurement of Functional Requirements

David Šenkýř ^a and Petr Kroha ^b

Faculty of Information Technology, Czech Technical University in Prague, Thákurova 9, Prague, Czech Republic

Keywords: Textual Functional Requirements Specification, Text Processing, Text Mining, Domain Model, Ambiguity, Incompleteness, Inconsistency, Quality Measurement.

Abstract: In this contribution, we propose a metric to measure the quality of textual functional requirements specifications. Since the main problem of such requirements specifications is their ambiguity, incompleteness, and inconsistency, we developed textual patterns to reveal shortcomings in these properties. As a component of our analysis, we use not only the text of the requirements but also the UML model that we construct during the text analysis. Combining the results of part-of-speech tagging of the text and the modeled properties, we are able to identify a number of irregularities concerning the properties named above. Then, the text needs human intervention to correct or remove the suspicious formulations. As a measure of the requirements specification quality, we denote the number of necessary human interventions. We implemented a tool called TEMOS that can test ambiguity, incompleteness, and inconsistency, and we use its results to evaluate the quality of textual requirements. In this paper, we summarize our project results.

1 INTRODUCTION

The idea of requirements specification is not new. The famous Italian painter Raffael (1483–1520) produced his paintings to the orders of wealthy customers. They wanted to know in advance what they would pay for. Rafael described the picture in words, for example, “*In the oil painting on canvas with dimensions according to the His Highness’s wishes, there will be His Highness like a knight in armor on a white horse with a sword and spear fighting a dragon*” and added a sketch (drawing) of the painting. The customer could opt, for example, for a black horse and a helmet with a plume. Then they wrote a contract, and the image had to match its content. After five hundred years, we follow (more or less) the same practice in software engineering projects. The description of the software product to be constructed is called requirements specification.


There are two viewpoints on the quality of a software product. It has to satisfy the requirements specification, and it has to state and imply the needs of all stakeholders. The problem is that the requirements specification usually does not reflect the needs of the stakeholders completely. Many stakeholders cannot articulate the requirements or do not even know what


they want. The second reason is that the analysts, who should help them, often do not deeply understand the semantics of the application domain (e.g., molecular biology). So, mistakes are practically guaranteed. For these reasons, it is worth investing effort into formulations of textual requirements before the analysis starts. Another point of view is the software product maintenance that is the most costly part of the whole project. It is very difficult to include all corrections, enhancements, and upgrades of the requirements into existing textual requirements specification.

Textual formulated requirements specifications are necessary as a base of communication between the customer, the user, the domain expert, and the analyst. It has one main advantage – it is understandable for all of them. In the event of a lawsuit between the company and the customer, the judge has a facilitated role because he understands the assignment and can assess whether the product meets the assignment.

Unfortunately, any text suffers from ambiguity, incompleteness, and inconsistency – in our project TEMOS, we focused on checking them. In Section 3, we define a quality measure that can be used not only in the first phase of the software development but also during the maintenance for checking the new versions of updated requirements specifications.

Mainly, the inspection is used to check the quality of requirements. Experienced people remove some

^a  <https://orcid.org/0000-0002-7522-3722>

^b  <https://orcid.org/0000-0002-1658-3736>

of the problems in requirements, but some of the shortcomings remain. Some deficiencies are cleared and corrected or removed gradually during the program development in the following development steps. Some of them are discovered during testing, and some others are revealed after release by the customer. Mistakes made but not corrected during the first phase of the cycle migrate then to other phases. This fact results in a costly process because all corrections cause costs that increase multiplicative with the later discovery and correction of the bugs.

It is known that manual approaches that rely on human intelligence and the application of inspection checklists do not scale to large specifications (Dalpiaz et al., 2018). Therefore, we try to detect errors in the requirements specification during the requirements definition process. We provide defect density, i.e., we count the number of detected places in formulations that may have the meanings of a defect. A human intervention makes the decision. In this contribution, we defined the quality metrics we developed and used in the requirements' quality improvement process. To support it, we summarized our methods for checking ambiguity, incompleteness, and inconsistency of textual functional requirements specifications implemented in our tool TEMOS.

2 RELATED WORK

Quality in software development on all levels is a topic discussed for many years, and it has been included into ISO standards, e.g., ISO/IEC/IEEE 29148:2011. Quality of requirements is a specific part of it. It is a well-established concept that is described in many papers, e.g., (Davis et al., 1993), (Fabbrini et al., 2001), (Nordin et al., 2017). In (Loucopoulos and Karakostas, 1995), the requirements engineering is described as an iterative co-operating process of improvements in the understanding gained. The survey is given in (Kocerka et al., 2018). In (Davis et al., 1993), the authors list 24 qualities of requirements. Most of them are qualitative; they can only be judged and not measured. Because of that, we selected only three of them – *ambiguity*, *incompleteness*, and *inconsistency*.

We used a mapping between requirements specification and a UML class model for plain English in our paper (Šenkýř and Kroha, 2018). In (Bugayenko, 2021), the author uses a similar concept for “controlled” English.

In (Femmer et al., 2017), the authors classify 166 rules for requirements and estimate that 53 % of them can be checked automatically with good heuristics.

They investigate the problem what cannot be checked automatically in requirements. We do not exclude human intervention because we think that the semantics may be very complex and that the mistakes caused by automated checking may be very expensive. In (Medeiros et al., 2016), requirements of agile projects are investigated.

In (Nordin et al., 2017), a study of practice is given, and it is stressed that the quality problems of requirements are an important topic. The quality evaluation is done manually during review sessions in (Saito et al., 2013). A survey of methods is given in (Kummler et al., 2018). The inspection is also described in (Takoshima and Aoyama, 2015).

It is known that most defects indicated in delivered software products are based on deficient requirements understanding (Génova et al., 2013). A complexity measure for textual requirements is described in (Antinyan et al., 2016). The measure indicates the amount of actions (and actors) and their relations in requirements. We do not investigate correctness like in (Feng et al., 2021). A semantic representation of functional requirements is investigated using methods of information retrieval in (Sonbol et al., 2020).

In (Wilson et al., 1997), the quality of textual requirements is measured using weak phrases, the size of text, text structure, hierarchical levels, and readability statistics. A similar approach is given in (Berry et al., 2006).

In (Bäumer and Geierhos, 2018), the authors developed methods of detecting quality violations in a requirements specification called *linguistic triggers*. Besides the problem of incompleteness, there is also presented an approach of ambiguity detection. First, the detection via *predicate argument analysis* in which a *semantic role labeler* assigns semantic roles such as *agent*, *theme*, and *beneficiary* to the recognized predicate. Presented illustrating example is the verb “send” that is a three-place predicate because it requires the agent (sender), the theme (sent) and the beneficiary argument (sent-to). If the beneficiary is not specified here, it is unknown whether one or more recipients are possible. The second step is compensation. Using the *similarity search component* known from the *information retrieval* (IR) domain, they try to find the potentially missing part *sent-to* based on software descriptions gathered from one software-to-download portal.

Patterns belong to the standard technology of text mining, e.g., (Bhatia et al., 2013). In (Eckhardt et al., 2016), sentence patterns have been used but for performance requirements, not for functional requirements like in our tool, and their sentence patterns are completely different from our sentence patterns.

Using NLP for requirements engineering is analyzed, e.g., in (Dalpiaz et al., 2018), (Zhao et al., 2021), (Ferrari et al., 2021). Building models from requirements is used for example in (Robeer et al., 2016). Differently to our approach, they do not use it to analyze requirements.

Ambiguity is defined in (Davis et al., 1993) as the percentage of requirements that have been interpreted in a unique manner by all its human reviewers. There are many papers about automated construction of glossaries, e.g., (Arora et al., 2017), (Gemkow et al., 2018), (Ezzini et al., 2021). In (Kose and Aydemir, 2021), a method is proposed to automatically extract a glossary from a set of models. We derived a glossary from the text of requirements. Ambiguity of words is investigated in (Gleich et al., 2010), (Arora et al., 2017), and (Kose and Aydemir, 2021). We additionally investigate the ambiguity of sentences (Section 3.1).

In (Dalpiaz et al., 2018), the authors explore potential ambiguity and incompleteness based on the terminology used in different viewpoints. They combine possibilities of NLP technology with information visualization. Their approach is completely different from our approach.

Two incompleteness metrics of input documents of the requirements specifications are described in (Ferrari et al., 2014). The second one – the *backwards functional completeness* that the paper focuses on – refers to the completeness of a functional requirements specification with respect to the input documents. Contrary to this approach, we do not measure the incompleteness using metrics and quantified results, even though it is a good idea.

In (Li, 2015), a meta-model approach is used to detect the missing information in a conceptual model. It is also an approach of the class forward functional completeness but at the level of a conceptual model.

3 OUR APPROACH TO QUALITY OF REQUIREMENTS

We developed our tool TEMOS to test whether there are some problems of ambiguity, incompleteness, and inconsistency in the requirements under consideration. Then we found that we can use it to measure the quality of the requirements as a “side effect”.

Our approach is based on a concept that is well-known in traditional publishing houses. During the editing process of manuscripts, all editor’s corrections can be qualified, collected and evaluated to the quality of the manuscript. We used the same method.

Each positive test (i.e., a test revealing a potential

problem) is evaluated, and the value becomes part of the quantitative description using metrics. The best quality (zero problems) is achieved when our algorithms do not find any suspicious formulations in the sense of ambiguity, incompleteness, and inconsistency. While checking, our system TEMOS always generates warning messages when it reveals some suspicious irregularities. We have defined the quality metrics of functional requirements as the value computed from the numbers of the generated warning messages. For clarity, it is structured according to the topics (ambiguity, incompleteness, inconsistency). In some topics, we can compute the relative quality, which is the number of the generated warning messages related to the number of sentences in the requirements. The proposed quality measure formula is

$$Q\text{-Req} = w_1 \cdot AW + w_2 \cdot AS + w_3 \cdot ISen + w_4 \cdot ISc + w_5 \cdot CGS + w_6 \cdot DSR$$

where variables w_i are weights that can be individually set. It uses the following components. **Ambiguity:** *AW* – the number of ambiguous words found, *AS* – the number of ambiguous sentences found. **Incompleteness:** *ISen* – the number of incomplete sentences found, *ISc* – the number of incomplete scenarios found. **Inconsistency:** *CGS* – the number of contradictions in groups of sentences, *DSR* – the number of necessary enrichments in the sense of the default consistency rules.

As we already mentioned in Section 1, our quality measurement can be (and should be) used during the maintenance to master all the inserted corrections, enhancements, and upgrades of the requirements into existing textual requirements specification.

Below, in separate sections, we present topics connected to each problem area mentioned in Section 1 – *ambiguity*, *incompleteness*, and *inconsistency*. Due to the scope of this paper, it is not our goal to recall here all details, so we only briefly mention the problems we solved in our previous work and we refer to our previous papers.

3.1 Problem of Ambiguity

We investigate the problem of ambiguity in our paper (Šenkýř and Kroha, 2019a). In our metric, we include the following.

The number of ambiguous words (AW). Ambiguity arises whenever a word or an expression can be interpreted in more than one way; the reason is that natural languages use: *homonyms*, i.e., words which are spelled alike but have different meanings, *synonyms*, i.e., different words that have the same meaning.

The number of ambiguous sentences (AS). Regarding the sentences, we can focus on:

- *syntactic methods* – to check a different meaning that depends on the position in the parsing tree of a sentence (example: “The button next to the warning box with the red border...” – Does the button have the red border? Or does the warning box have the red border?),
- *semantic methods* – the semantic meaning includes coreferences and a model created during requirements processing or a model imported from an external ontology source; for example, the model of sentences “Our delivery contains product XYZ-123 in container X-50. Its weight is 50 kg.” can contain classes `delivery`, `product`, and `container` and all these classes can have the attribute `weight`.

3.2 Problem of Incompleteness

We investigate the problem of incompleteness in our papers (Šenkýř and Kroha, 2019b) and (Šenkýř and Kroha, 2020). In our metric, we include the following.

The number of incomplete sentences (ISen). On the level of a sentence, we can check the *usual usage of words* – in the sense that we expect usual text fragments – for example, we define a common noun and preposition collocation set – if the kind of usage in the textual requirements (e.g., *a list*) does not correspond to the kind of usage in the vocabulary (e.g., *a list of*) then a warning is counted; *acronyms definition* – all acronyms should be defined (in an attached vocabulary); *semantic model* (as mentioned in the previous section) – here, we can check described *actions*, i.e., verbs in sentences of queries are investigated in the sense whether the action that they describe, can be performed in the existing model (e.g., sorting without a key).

The number of incomplete scenarios (ISc). For recognized enumerations (e.g., of attribute values), we can check if the recognized scenarios cover all known values of the specific enumeration. We can also check the generated class model extracted from the requirements – warnings can be counted when we notice “empty” classes without attributes or classes without relation to any other class.

3.3 Problem of Inconsistency

We investigate the problem of incompleteness in our papers (Šenkýř and Kroha, 2021), (Šenkýř and Kroha, 2021a), and (Šenkýř and Kroha, 2021b). In our metric, we include the following.

The number of contradictions in groups of sentences (CGS). The linguistic sources of the inconsis-

tency include using *antonyms*, *a negation*, or a combination of *synonyms* together with *changed roles of subject and object*, *passive voice*, and *negation*, e.g., “the user can edit a document” *contra* “the user cannot correct a document in this mode” (here, we suppose that the verbs “to edit” and “to correct” have the same meaning) *contra* “a document cannot be corrected by the user”. Using *numerically different data*, e.g., “you will start the function by double click” *contra* “you will start the function by one click”. Using *factive contradiction* in the sense of attributes of the subject. Using *lexical contradiction*, e.g., “to obtain results stay joined and wait” *contra* “to obtain results restart the application” *contra* “to obtain results restart the system”. Using *world knowledge* to indicate the contradiction, e.g., “there is public access to your private data”.

Additionally, some words may change, influence, or limit the sense of the sentence, e.g., *but*, *except*, *however*, *instead of*, *when*, *so that*, *that*.

The number of necessary enrichments in the sense of the default consistency rules (DSR). In our paper (Šenkýř and Kroha, 2021b), we define the default consistency rules as the omitted part of the textual requirements that we can try to extend by processing sentences from external sources. For that purpose, we can cluster existing triplets (subject—predicate—object) from the requirements, and each clustered triplet compare with sentences matching the same triplet (or at least a part of the triplet) from external sources.

4 DATA AND EXPERIMENTS

To evaluate our proposed methods, we have prepared a data set¹. We have collected textual requirements from four different sources: **(1)** a set of requirements from the PUBlic REquirements Documents (PURE) data set² (prefixed with uppercase *P* in the evaluation table), **(2)** a set of requirements collected by (Hayes et al., 2019) (prefixed with uppercase *H* in the evaluation table), **(3)** a set of requirements provided by (Dalpiaz et al., 2019) (prefixed with lowercase *g* in the evaluation table), **(4)** a custom collection of requirements found in the Internet (prefixed with uppercase *C* in the evaluation table).

In Table 1, for each input requirements text (case) from the aforementioned data set, we record (a legend of the table):

- ARI – automated readability index – value ob-

¹<https://zenodo.org/record/7897601>

²<https://zenodo.org/record/7118517>

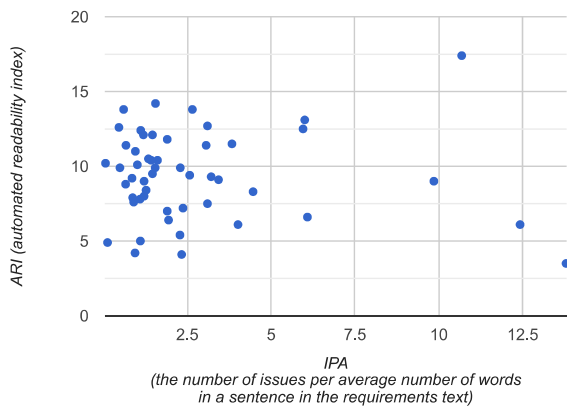


Figure 1: Correlation comparison of IPA and ARI.

tained using the *Free Text Complexity Analyzer* online tool³,

- the number of all recognized issues (warnings) in the sense of the proposed quality measure formula Q-Req defined in Section 3, where we use all weights w_i equivalent and equal to 1,
- EN – the number of recognized entities in the requirements text,
- IEN – the number of issues per one recognized entity in the requirements text,
- RN – the number of recognized relations in the requirements text,
- IRN – the number of issues per one recognized relation in the requirements text,
- AN – the number of recognized attributes in the requirements text,
- IAN – the number of issues per one recognized attribute in the requirements text,
- IPW – the number of issues per word in the requirements text,
- IPS – the number of issues per sentence in the requirements text,
- IPA – the number of issues per average number of words in a sentence in the requirements text.

In Figure 1, we compare the correlation of IPA (the number of issues per average number of words in a sentence in the requirements text) and ARI (automated readability index). According to the used data set, it cannot be claimed that the requirements texts with the most generated warnings are the most complex texts according to ARI at the same time.

In Figure 2, we compare the correlation of EN (the number of recognized entities in the requirements

³<https://www.lumoslearning.com/llwp/free-text-complexity-analysis.html>

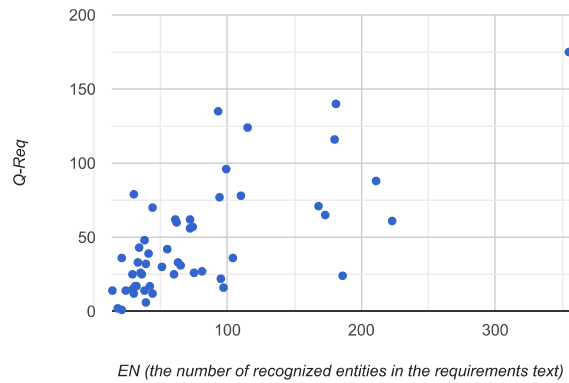


Figure 2: Correlation comparison of EN and Q-Req.

text) and Q-Req (the number of generated warnings). In this case, as might be intuitively expected, the increasing size of the generated model in the sense of recognized entities is reflected in the number of generated warnings due to the need for a more complex specification.

5 CONCLUSIONS

In this contribution, we described a method to check the quality of textual functional requirements. We used the linguistic patterns we originally developed to build a UML class model from textual requirements. Our implemented system TEMOS analyses the *ambiguity*, *incompleteness*, and *inconsistency* of textual functional requirements and indicates a set S of recognized suspicious irregularities. This set is used to generate warning messages that inform the analyst about the problems found.

To measure the quality of requirements, we have defined a simple formula that uses the number of elements in the set S – see Section 3. We compute the formula and declare this result to be the value of our quality metric. In the evaluation (Section 4), we compare the result value of this metric with the readability index ARI, recognized parts of the generated model from the requirements, and number of words and sentences of the requirements.

We assume the analysts remove some of the problem sources according to the generated warnings. Then they iterate in quality checking as long as there are no more problems found or the problems can be explained by the imperfection of our system TEMOS. As we declare above, our methods do not guarantee that they reveal all problems concerning ambiguity, incompleteness, and inconsistency. The semantics of natural languages is enormously complex, and it is based not only on the written text but also on educa-

Table 1: The evaluation of the recognized issues (warnings) using Q-Req.

Case	ARI	Q-Req	EN	IEN	RN	IRN	AN	IAN	IPW	IPS	IPA
g02-federalspending	8.4	27	81	0.33	51	0.53	0	0.00	0.01	0.28	1.27
g03-loudoun	14.2	43	34	1.26	23	1.87	1	43.00	0.03	0.75	1.55
g04-recycling	10.1	25	29	0.86	17	1.47	0	0.00	0.02	0.49	1.01
g05-openspending	12.6	14	38	0.37	26	0.54	0	0.00	0.01	0.26	0.46
g08-frictionless	11.8	48	38	1.26	29	1.66	1	48.00	0.03	0.73	1.90
g10-scrumalliance	9.9	60	62	0.97	45	1.33	1	60.00	0.02	0.61	2.29
g11-nsf	9.0	25	36	0.69	21	1.19	0	0.00	0.01	0.34	1.21
g12-camperplus	10.5	36	21	1.71	15	2.40	0	0.00	0.03	0.68	1.34
g13-planningpoker	10.4	39	41	0.95	34	1.15	1	39.00	0.03	0.74	1.42
g14-datahub	11.0	26	35	0.74	33	0.79	2	13.00	0.01	0.39	0.95
g16-mis	12.1	33	63	0.52	62	0.53	1	33.00	0.02	0.49	1.46
g17-cask	11.4	17	42	0.40	45	0.38	2	8.50	0.01	0.27	0.67
g18-neurohub	9.4	56	72	0.78	62	0.90	0	0.00	0.03	0.55	2.57
g19-alfred	7.2	42	55	0.76	42	1.00	2	21.00	0.02	0.30	2.37
g21-badcamp	12.1	32	39	0.82	26	1.23	0	0.00	0.02	0.46	1.19
g22-rdadmp	12.4	30	51	0.59	48	0.63	0	0.00	0.01	0.36	1.11
g23-archivesspace	7.6	14	14	1.00	9	1.56	0	0.00	0.02	0.25	0.90
g24-unibath	13.8	17	31	0.55	21	0.81	1	17.00	0.01	0.33	0.60
g25-duraspace	9.5	31	65	0.48	67	0.46	4	7.75	0.02	0.31	1.46
g26-racdam	9.9	33	33	1.00	23	1.43	0	0.00	0.02	0.33	1.54
g27-culrepo	13.8	77	94	0.82	64	1.20	1	77.00	0.02	0.64	2.65
g28-zooniverse	9.2	15	29	0.52	17	0.88	0	0.00	0.01	0.25	0.85
P01. Blit	5.0	12	30	0.40	34	0.35	1	12.00	0.02	0.25	1.10
P02. CS179G...	8.0	22	95	0.23	83	0.27	6	3.67	0.02	0.33	1.21
P03. eProcurement	11.4	62	72	0.86	78	0.79	0	0.00	0.04	0.69	3.06
P04. Grid 3D	4.9	2	18	0.11	13	0.15	0	0.00	0.01	0.18	0.12
P05. Home 1.3	6.4	25	60	0.42	74	0.34	2	12.50	0.02	0.29	1.94
P06. Integrated...	8.3	124	115	1.08	149	0.83	1	124.00	0.06	1.57	4.46
P07. Inventory	3.5	116	180	0.64	317	0.37	8	14.50	0.02	0.23	13.79
P08. KeePass...	4.2	12	44	0.27	32	0.38	0	0.00	0.03	0.33	0.94
P09. Mashbot	8.8	14	24	0.58	28	0.50	0	0.00	0.02	0.54	0.66
P10. MultiMahjong	9.3	62	61	1.02	66	0.94	1	62.00	0.04	0.70	3.21
P11. Nenios	6.6	70	44	1.59	59	1.19	2	35.00	0.07	0.85	6.08
P12. Pontis 5.0...	11.5	71	168	0.42	267	0.27	3	23.67	0.02	0.32	3.83
P13. Public Health...	17.4	140	181	0.77	244	0.57	0	0.00	0.05	1.27	10.68
P14. Publications...	9.1	96	99	0.97	196	0.49	3	32.00	0.04	1.57	3.43
P15. Puget Sound...	7.0	36	104	0.35	123	0.29	1	36.00	0.02	0.39	1.90
P16. Tactical...	7.5	61	223	0.27	204	0.30	1	61.00	0.01	0.21	3.10
P17. Tarrant...	13.1	79	30	2.63	33	2.39	4	19.75	0.04	0.59	6.00
P18. X-38 Fault...	12.5	88	211	0.42	233	0.38	4	22.00	0.02	0.25	5.95
H01. CCHIT	12.7	65	173	0.38	205	0.32	4	16.25	0.03	0.58	3.10
H02. CM1	10.2	1	21	0.05	13	0.08	0	0.00	0.00	0.03	0.06
H03. InfusionPump	10.4	24	186	0.13	185	0.13	2	12.00	0.01	0.10	1.61
H04. Waterloo	9.0	175	355	0.49	1070	0.16	3	58.33	0.01	0.26	9.85
C01. Amazing...	5.4	78	110	0.71	251	0.31	0	0.00	0.02	0.84	2.28
C02. EU Rent	4.1	26	75	0.35	97	0.27	6	4.33	0.05	0.60	2.33
C03. FDP Expand...	9.9	6	39	0.15	29	0.21	1	6.00	0.01	0.15	0.49
C04. Library System	6.1	57	74	0.77	109	0.52	1	57.00	0.03	0.45	4.01
C05. Nodes Portal...	7.8	16	97	0.16	259	0.06	1	16.00	0.01	0.10	1.09
C06. Online Nat...	6.1	135	93	1.45	137	0.99	3	45.00	0.04	0.51	12.42
C07. Restaurant...	7.9	17	32	0.53	65	0.26	0	0.00	0.02	0.37	0.87

tion and experience, i.e., each human being has (or can have) a different model of the world in its head.

ACKNOWLEDGEMENTS

This research was supported by the grant of Czech Technical University in Prague No. SGS23/206/OHK3/3T/18.

REFERENCES

- Antinyan, V., Staron, M., Sandberg, A., and Hansson, J. (2016). A Complexity Measure for Textual Requirements. In *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, pages 148–158, Los Alamitos, CA, USA. IEEE Computer Society.
- Arora, C., Sabetzadeh, M., Briand, L., and Zimmer, F. (2017). Automated Extraction and Clustering of Requirements Glossary Terms. *IEEE Transactions on Software Engineering*, 43(10):918–945.
- Bäumer, F. S. and Geierhos, M. (2018). Flexible Ambiguity Resolution and Incompleteness Detection in Requirements Descriptions via an Indicator-Based Configuration of Text Analysis Pipelines. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, pages 5746–5755.
- Berry, D., Bucchiarone, A., Gnesi, S., Lami, G., and G., T. (2006). A New Quality Model for Natural Language Requirements Specifications. In *Proceedings of the 12th International Working Conference on requirements Engineering: Foundation of Software Quality (REFSQ-06)*.
- Bhatia, J., Sharma, R., Biswas, K., and Ghaisas, S. (2013). Using Grammatical Knowledge Patterns for Structuring Requirements Specifications. In *IEEE Third International Workshop on Requirements Patterns (RePa)*.
- Bugayenko, Y. (2021). Combining Object-Oriented Paradigm and Controlled Natural Language for Requirements Specification. In *Proceedings of the 1st ACM SIGPLAN International Workshop on Beyond Code: No Code*, BCNC 2021, page 11–17, New York, NY, USA. Association for Computing Machinery.
- Dalpiaz, F., Ferrari, A., Franch, X., and Palomares, C. (2018). Natural Language Processing for Requirements Engineering: The Best Is Yet to Come. In *Proceedings of the Workshop NLPRE'18*, pages 70–77. IEEE.
- Dalpiaz, F., van der Schalk, I., Brinkkemper, S., Aydemir, F. B., and Lucassen, G. (2019). Detecting terminological ambiguity in user stories: Tool and experimentation. *Information and Software Technology*, 110:3–16.
- Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledebner, G., Reynolds, P., Sitaram, P., Ta, A., and Theofanos, M. (1993). Identifying and Measuring Quality in a Software Requirements Specification. In *Proceedings First International Software Metrics Symposium*, pages 141–152, Los Alamitos, CA, USA. IEEE Computer Society.
- Eckhardt, J., Vogelsang, A., Femmer, H., and Mager, P. (2016). Challenging Incompleteness of Performance Requirements by Sentence Patterns. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 46–55, Beijing, China. IEEE.
- Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., and Briand, L. C. (2021). Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 1485–1497, Los Alamitos, CA, USA. IEEE Computer Society.
- Fabbrini, F., Fusani, M., Gnesi, S., and Lami, G. (2001). An Automatic Quality Evaluation for Natural Language Requirements. In *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ)*, volume 1.
- Femmer, H., Unterkalmsteiner, M., and Gorschek, T. (2017). Which Requirements Artifact Quality Defects are Automatically Detectable – A Case Study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops*, pages 400–406. IEEE Computer Society.
- Feng, S., Chen, X., Li, Q., and Zhao, Y. (2021). RE2B: Enhancing Correctness of Both Requirements and Design Models. In *2021 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 191–198, Los Alamitos, CA, USA. IEEE Computer Society.
- Ferrari, A., dell’Orletta, F., Spagnolo, G. O., and Gnesi, S. (2014). Measuring and Improving the Completeness of Natural Language Requirements. In *Proceedings of Requirements Engineering: Foundation for Software Quality - REFSQ 2014*. LNCS 8396, pages 23–38, Cham. Springer.
- Ferrari, A., Zhao, L., and Alhoshan, W. (2021). NLP for Requirements Engineering: Tasks, Techniques, Tools, and Technologies. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 322–323, Los Alamitos, CA, USA. IEEE Computer Society.
- Gemkow, T., Conzelmann, M., Hartig, K., and Vogelsang, A. (2018). Automatic Glossary Term Extraction from Large-Scale Requirements Specifications. In *IEEE 26th International Requirements Engineering Conference*, pages 412–417.
- Gleich, B., Creighton, O., and Kof, L. (2010). Measuring and Improving the Completeness of Natural Language Requirements. In *Proceedings of Requirements Engineering: Foundation for Software Quality - REFSQ 2010*, LNCS 6182, pages 233–247, Cham. Springer.
- Génova, G., Fuentes, J. M., Llorens, J., Hurtado, O., and Moreno, V. (2013). A Framework to Measure and Improve the Quality of Textual Requirements. *Requirements Engineering*, 18(1):25–41.

- Hayes, J. H., Payne, J., and Leppelmeier, M. (2019). Toward Improved Artificial Intelligence in Requirements Engineering: Metadata for Tracing Datasets. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, pages 256–262.
- Kocerka, J., Krzeslak, M., and Galuszka, A. (2018). Analysing Quality of Textual Requirements using Natural Language Processing: A Literature Review. In *Conference: 2018 23rd International Conference on Methods and Models in Automation and Robotics (MMAR)*.
- Kose, S. and Aydemir, F. (2021). Automated Glossary Extraction from Collaborative Requirements Models. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 11–15, Los Alamitos, CA, USA. IEEE Computer Society.
- Kummler, P., Vernisse, L., and Fromm, H. (2018). How Good are My Requirements? A New Perspective on the Quality Measurement of Textual Requirements. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 156–159, Los Alamitos, CA, USA. IEEE Computer Society.
- Li, A. (2015). Analysis of Requirements Incompleteness Using Metamodel Specification. Master's thesis, University of Tampere.
- Loucopoulos, P. and Karakostas, V. (1995). *System Requirements Engineering*. McGraw-Hill.
- Medeiros, J., Goulao, M., Vasconcelos, A., and Silva, C. (2016). Towards a Model about Quality of Software Requirements Specification in Agile Projects. In *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 236–241, Los Alamitos, CA, USA. IEEE Computer Society.
- Nordin, A., Zaidi, A., and Mazlan, A. (2017). Measuring Software Requirements Specification Quality. *Journal of Telecommunication, Electronic and Computer Engineering*, 9(3-5):123–128.
- Robeer, M., Lucassen, G., van der Werf, J. M. E. M., Dalpiaz, F., and Brinkkemper, S. (2016). Automated Extraction of Conceptual Models from User Stories via NLP. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 196–205.
- Saito, S., Takeuchi, M., Hiraoka, M., Kitani, T., and Aoyama, M. (2013). Requirements Clinic: Third Party Inspection Methodology and Practice for Improving the Quality of Software Requirements Specifications. In *2013 IEEE 21st International Requirements Engineering Conference (RE)*, pages 290–295, Los Alamitos, CA, USA. IEEE Computer Society.
- Šenkýř, D. and Kroha, P. (2018). Patterns in Textual Requirements Specification. In *Proceedings of the 13th International Conference on Software Technologies*, pages 197–204, Porto, Portugal. SCITEPRESS – Science and Technology Publications.
- Šenkýř, D. and Kroha, P. (2019a). Patterns of Ambiguity in Textual Requirements Specification. In Rocha, Á., Adeli, H., Reis, L. P., and Costanzo, S., editors, *New Knowledge in Information Systems and Technologies*, volume 1, pages 886–895, Cham. Springer International Publishing.
- Šenkýř, D. and Kroha, P. (2019b). Problem of Incompleteness in Textual Requirements Specification. In *Proceedings of the 14th International Conference on Software Technologies*, volume 1, pages 323–330, Porto, Portugal. INSTICC, SCITEPRESS – Science and Technology Publications.
- Šenkýř, D. and Kroha, P. (2020). Patterns for Checking Incompleteness of Scenarios in Textual Requirements Specification. In *Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering*, volume 1, pages 289–296, Porto, Portugal. INSTICC, SCITEPRESS – Science and Technology Publications.
- Šenkýř, D. and Kroha, P. (2021a). Problem of Inconsistency and Default Consistency Rules. In Fujita, H. and Pérez-Meana, H., editors, *New Trends in Intelligent Software Methodologies, Tools and Techniques – Proceedings of the 20th International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques (SoMeT 2021)*, volume 337 of *Frontiers in Artificial Intelligence and Applications*, pages 674–687. IOS Press.
- Šenkýř, D. and Kroha, P. (2021b). Problem of Semantic Enrichment of Sentences Used in Textual Requirements Specification. In Polyvyanyy, A. and Rinderle-Ma, S., editors, *Advanced Information Systems Engineering Workshops*, pages 69–80, Cham. Springer International Publishing.
- Šenkýř, D. and Kroha, P. (2021). Problem of Inconsistency in Textual Requirements Specification. In Ali, R., Kaindl, H., and Maciaszek, L. A., editors, *Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering – ENASE*, pages 213–220. INSTICC, SciTePress.
- Sonbol, R., Rebdawi, G., and Ghneim, N. (2020). Towards a Semantic Representation for Functional Software Requirements. In *2020 IEEE Seventh International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, pages 1–8, Los Alamitos, CA, USA. IEEE Computer Society.
- Takoshima, A. and Aoyama, M. (2015). Assessing the Quality of Software Requirements Specifications for Automotive Software Systems. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*, pages 393–400, Los Alamitos, CA, USA. IEEE Computer Society.
- Wilson, W., Rosenberg, L., and Hyatt, L. (1997). Automated Analysis of Requirement Specifications. In *Automated Software Engineering*, pages 161–171.
- Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E.-V., and Batista-Navarro, R. T. (2021). Natural Language Processing for Requirements Engineering: A Systematic Mapping Study. *ACM Comput. Surv.*, 54(3).