Quality Assessment for Requirements based on Natural Language Processing

Mathias Soeken, Nabila Abdessaied, Arman Allahyari-Abhari, Christopher B. Harris, Ian G. Harris, Liana Musat, Georg Pelz, Andi Buzo

Rolf Drechsler

German Research Center for Artificial Intelligence

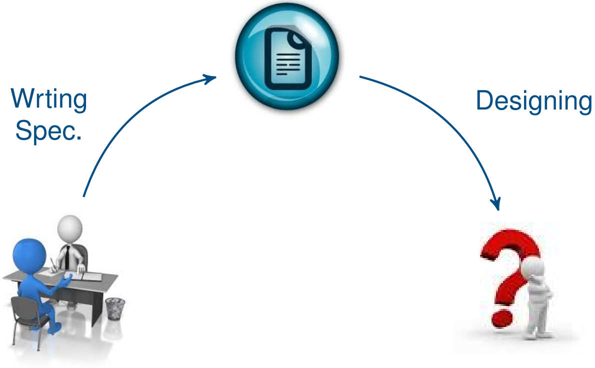
Infineon

University of Bremen

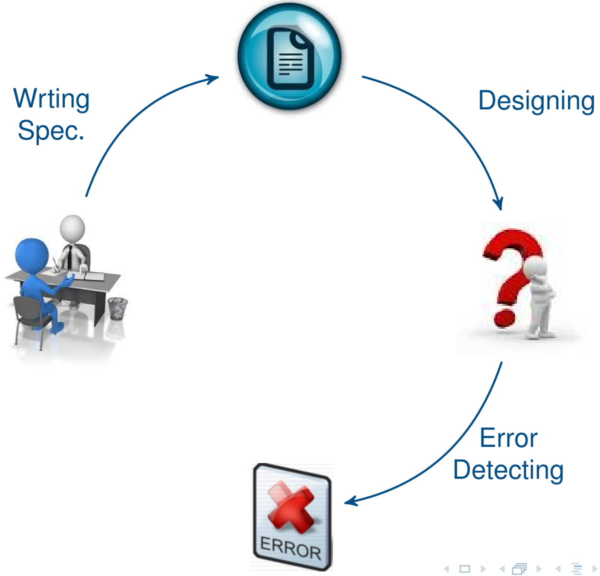
Motivation

Motivation

Motivation

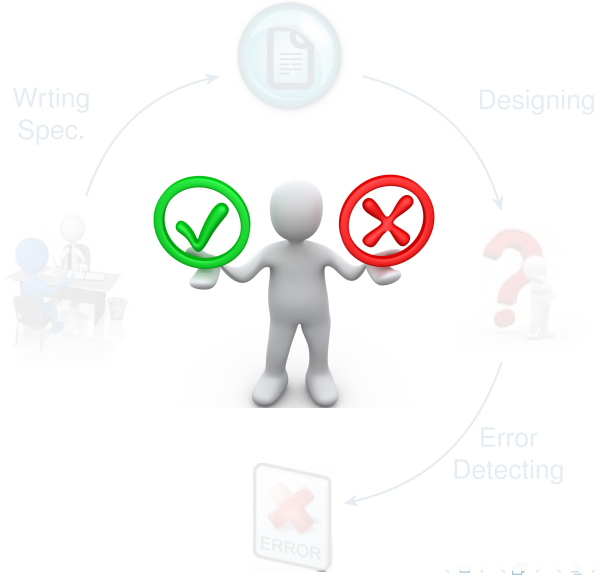


Motivation



Motivation

Motivation



Solution

Writing Specification

Solution

Writing Specification

Solution

Writing Specification

Quality Validation

Consequences

Better Writen specification will:

Consequences

Better Writen specification will:

- improve the comprehensibility of the designer

Consequences

Better Writen specification will:

- improve the comprehensibility of the designer
- enhance the quality of automatic extraction approaches

Approaches

1. Requirement quality based on:

Approaches

1. Requirement quality based on:
   - Syntactic Quality

Approaches

1. Requirement quality based on:
   - Syntactic Quality
   - Semantic Quality

Approaches

1. Requirement quality based on:
   - Syntactic Quality
   - Semantic Quality
2. Assertions translation approach

Static Sentence Analysis

Problem (Sentence quality)

Given an English sentence, the sentence quality problem asks to determine a quality measure indicating whether the sentence is good, medium, or bad in terms of comprehension.

Syntactic Quality Analysis

It is considered as directly ambivalent to the number of structural ambiguities.

Syntactic Quality Analysis

It is considered as directly ambivalent to the number of structural ambiguities.

- Calculation based on phrase structure trees

Syntactic Quality Analysis

It is considered as directly ambivalent to the number of structural ambiguities.

- Calculation based on phrase structure trees
- *Isomorphic subtrees*

Syntactic Quality Analysis

It is considered as directly ambivalent to the number of structural ambiguities.

- Calculation based on phrase structure trees
- *Isomorphic subtrees*
- *Sentence length penalty*

Semantic Quality Analysis

- It is determined by its amount of semantic ambiguities

Semantic Quality Analysis

- It is determined by its amount of semantic ambiguities
- WordNet is used to determine that

Semantic Quality Analysis

- It is determined by its amount of semantic ambiguities
- WordNet is used to determine that
- A word is ambiguous if it has more than one *synsets* in the WordNet dictionary

Semantic Quality Analysis

- It is determined by its amount of semantic ambiguities
- WordNet is used to determine that
- A word is ambiguous if it has more than one *synsets* in the WordNet dictionary
- A *synsets* is a set of cognitive synonyms

Semantic Quality Analysis

let :

Semantic Quality Analysis

let :

- n be the number of nonambiguous words

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words
- $a = n/(n+m)$ be the distinct portion of the sentence

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words
- $a = n/(n+m)$ be the distinct portion of the sentence
- $b = m/(n+m)$ be the ambiguous portion of the sentence

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words
- $a = n/(n+m)$ be the distinct portion of the sentence
- $b = m/(n+m)$ be the ambiguous portion of the sentence
- k_i be the number of different synsets of the i-th ambiguous word

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words
- $a = n/(n+m)$ be the distinct portion of the sentence
- $b = m/(n+m)$ be the ambiguous portion of the sentence
- k_i be the number of different synsets of the i-th ambiguous word

Then the basic semantic quality is given by:

Semantic Quality Analysis

let :

- n be the number of nonambiguous words
- m be the number of ambiguous words
- $a = n/(n+m)$ be the distinct portion of the sentence
- $b = m/(n+m)$ be the ambiguous portion of the sentence
- k_i be the number of different synsets of the i-th ambiguous word

Then the basic semantic quality is given by:

$$q_{\text{sem}} = a + b \cdot \frac{m}{\sum_{i=1}^{m} k_i}. \tag{1}$$

Requirements Database

The requirements database is taken from:

Requirements Database

The requirements database is taken from:

- NASA hardware requirement specifications

Requirements Database

The requirements database is taken from:

- NASA hardware requirement specifications
- Intel hardware requirement specifications.

Requirements Database

The requirements database is taken from:

- NASA hardware requirement specifications
- Intel hardware requirement specifications.
- Random hardware requirement specifications

Experimental Evaluation

Quality predicate	Algo.	Subj.	Matches	Misses dist. 1	Misses dist. 2	Matching percentage
good	32	26	20	8	4	62.5 %
medium	53	41	29	24	×	54.7 %
bad	37	55	31	4	2	83.8 %
total	122	122	80	36	6	65.6 %

Guidelines Validation

Problem (Guideline checking)

Given a set of rules from guidelines how to write requirements and a natural language requirement R, the guideline checking problem asks whether R adheres to the rules.

Guidelines Validation

Requirement quality Evaluation based on:

Guidelines Validation

Requirement quality Evaluation based on:

- Rule based approach

Guidelines Validation

Requirement quality Evaluation based on:

- Rule based approach
- Stanford CoreNP
- Scala programming language

Rules

The rules are extracted from:

Rules

The rules are extracted from:

- NASA guidelines

Rules

The rules are extracted from:

- NASA guidelines
- IBM guidelines.

Rules

The rules are extracted from:

- NASA guidelines
- IBM guidelines.
- "Writing Better Requirement", I. Alexander and R. Stevens

Rules

1. Define one requirement at a time.

Rules

1. Define one requirement at a time.
2. Avoid conjunctions (and, or, with, also) that make multiple requirements.

Rules

1. Define one requirement at a time.
2. Avoid conjunctions (and, or, with, also) that make multiple requirements.
3. Use simple direct sentences.

Rules

1. Define one requirement at a time.
2. Avoid conjunctions (and, or, with, also) that make multiple requirements.
3. Use simple direct sentences.
4. Each requirement must contain a subject and a predicate.

Rules

1. Define one requirement at a time.
2. Avoid conjunctions (and, or, with, also) that make multiple requirements.
3. Use simple direct sentences.
4. Each requirement must contain a subject and a predicate.
5. Avoid let-out clauses (unless, except, if necessary, but, when, unless, although).

Rules

1. Define one requirement at a time.
2. Avoid conjunctions (and, or, with, also) that make multiple requirements.
3. Use simple direct sentences.
4. Each requirement must contain a subject and a predicate.
5. Avoid let-out clauses (unless, except, if necessary, but, when, unless, although).
6. Avoid expressing suggestions or possibilities (might, may, could, ought, should, could, perhaps, probably).

Rules

7. Avoid weak phrases and undefined terms (adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely, tbd, user-friendly, versatile, robust, approximately, minimal impact, etc., and so on, flexible, to the maximum extent, as much as possible, minimal impact, in one whack, different, various, many, some of, diverse)

Rules

7. Avoid weak phrases and undefined terms (adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely, tbd, user-friendly, versatile, robust, approximately, minimal impact, etc., and so on, flexible, to the maximum extent, as much as possible, minimal impact, in one whack, different, various, many, some of, diverse)
8. Do not speculate (usually, generally, often, normally, typically).

Rules

7. Avoid weak phrases and undefined terms (adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely, tbd, user-friendly, versatile, robust, approximately, minimal impact, etc., and so on, flexible, to the maximum extent, as much as possible, minimal impact, in one whack, different, various, many, some of, diverse)
8. Do not speculate (usually, generally, often, normally, typically).
9. Avoid wishful thinking (100% reliable, safe, handle all failures, fully upgradeable, run on all platforms).

Rules

7. Avoid weak phrases and undefined terms (adequate, as a minimum, as applicable, easy, as appropriate, be able to, be capable, but not limited to, capability of, capability to, effective, if practical, normal, provide for, timely, tbd, user-friendly, versatile, robust, approximately, minimal impact, etc., and so on, flexible, to the maximum extent, as much as possible, minimal impact, in one whack, different, various, many, some of, diverse)
8. Do not speculate (usually, generally, often, normally, typically).
9. Avoid wishful thinking (100% reliable, safe, handle all failures, fully upgradeable, run on all platforms).
10. Define verifiable criteria.

Implementation

R1. Define one requirement at a time.

Implementation

R1. Define one requirement at a time.

the system is reset at start-up.

Implementation

R1. Define one requirement at a time.

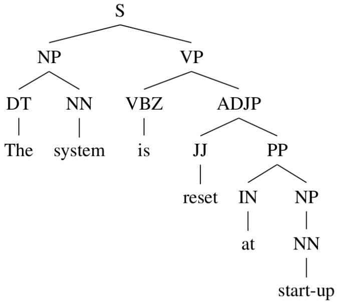
the system is reset at start-up.

```
(ROOT
  (S
    (NP (DT the) (NN system))
    (VP (VBZ is)
      (ADJP (JJ reset)
        (PP (IN at)
          (NP (NN start-up)))))
    (. .)))
```

Implementation

R1. Define one requirement at a time.

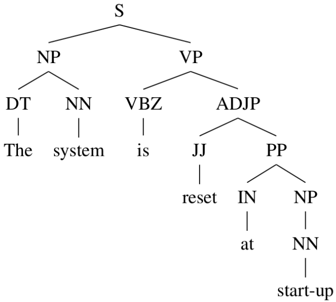
the system is reset at start-up.



Implementation

R1. Define one requirement at a time.

the system is reset at start-up.



Cheeking the phrase structure tree whether it has two sentences related directly to the root.

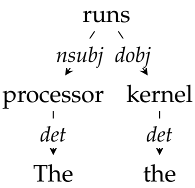
Implementation

R3. Use simple direct sentences.

Implementation

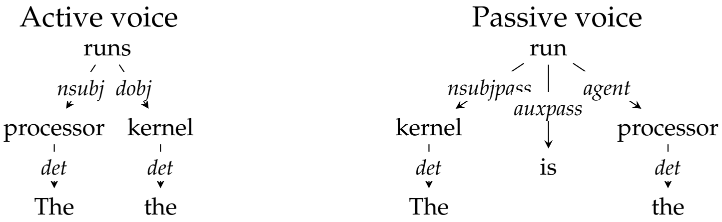
R3. Use simple direct sentences.

Active voice



Implementation

R3. Use simple direct sentences.



Requirements Database

The requirements database is the same as the one used in the first approach.

Experimental Evaluation

Rules	M. Class.		A. Class.		Classifier Evaluation					
	T	F	T	F	SA	TP	TN	FP	FN	Acc.
R1	88	15	61	42	70	58	12	30	3	67.96%
R2	84	19	89	14	80	75	5	9	14	77.67%
R3	90	13	89	14	86	81	5	9	8	83.50%
R4	102	1	92	11	93	92	1	10	0	90.29%
R5	94	9	95	8	102	94	8	0	1	99.03%
R6	92	11	85	18	92	83	9	9	2	89.32%
R7	102	1	103	0	102	102	0	0	1	99.03%
R8	103	0	103	0	103	103	0	0	0	100.00%
R9	9	17	18	8	13	7	6	2	11	50.00%
Total	826	127	772	181	811	728	83	98	44	82.48%

Experimental Evaluation for Infenion

Rules	M. Class.		A. Class.		Classifier Evaluation					
	T	F	T	F	SA	TP	TN	FP	FN	Acc.
R1	52	32	54	30	76	49	27	3	5	90.48%
R2	68	16	35	49	45	32	13	36	3	53.57%
R3	31	53	19	65	62	14	48	17	5	73.81%
R4	76	8	76	8	80	74	6	2	2	95.24%
R5	77	7	75	9	80	74	6	3	1	95.24%
R6	65	19	61	23	78	60	18	5	1	92.86%
R7	82	2	80	4	82	80	2	2	0	97.62%
R8	84	0	84	0	84	84	0	0	0	100.00%
R9	26	8	27	7	19	19	0	7	8	55.88%
Total	589	201	541	249	680	510	170	79	31	84.28%

Notations

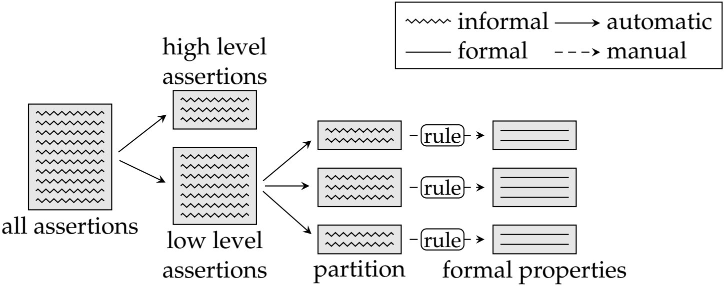
- SA= Number of the Same annotated requirement
- TP+FP= Number of manual True annotated requirements
- TN+FN= Number of manual false annotated requirements
- Acc=$\frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$

Automating the Translation of Assertions

Proposed flow

Automating the Translation of Assertions

Proposed flow



Algorithm

1. Abstraction Level Classification

Algorithm

1. Abstraction Level Classification
2. Partitioning based on Sentence Similarity

Algorithm

1. Abstraction Level Classification
2. Partitioning based on Sentence Similarity
3. Assertion Generation

Abstraction Level Classification

Using A SPARQL query to partition the assertion into

Abstraction Level Classification

Using A SPARQL query to partition the assertion into

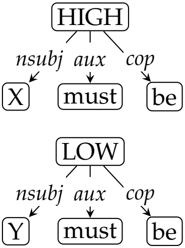
- high abstraction level subsets

Abstraction Level Classification

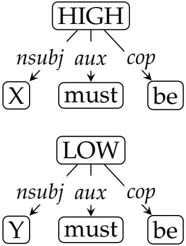
Using A SPARQL query to partition the assertion into

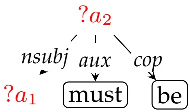
- high abstraction level subsets
- low abstraction level subsets

Partitioning based on Sentence Similarity



Partitioning based on Sentence Similarity





Assertion Generation

1. an assertion template is generated for each cluster

Assertion Generation

1. an assertion template is generated for each cluster
2. the assertion template is populated for each natural language assertion in the cluster.

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

- AMBA 3 AXI Protocol Checker user guide containing 145 assertions

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

- AMBA 3 AXI Protocol Checker user guide containing 145 assertions
- Random hardware assertion list

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

- AMBA 3 AXI Protocol Checker user guide containing 145 assertions
- Random hardware assertion list

Requirement quality Evaluation based on:

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

- AMBA 3 AXI Protocol Checker user guide containing 145 assertions
- Random hardware assertion list

Requirement quality Evaluation based on:

- Stanford CoreNP

Experimental Evaluation

The algorithm is applied to the assertions is taken from:

- AMBA 3 AXI Protocol Checker user guide containing 145 assertions
- Random hardware assertion list

Requirement quality Evaluation based on:

- Stanford CoreNP
- JENA API for the triple store based information extraction

Experimental Evaluation

1. Abstraction Level Classification

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions
2. Partitioning based on Sentence Similarity

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions
2. Partitioning based on Sentence Similarity
   - 11 clusters are found

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions
2. Partitioning based on Sentence Similarity
   - 11 clusters are found
3. Partitioning based on Sentence Similarity

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions
2. Partitioning based on Sentence Similarity
   - 11 clusters are found
3. Partitioning based on Sentence Similarity
   - 11 SystemVerilog assertion templates were generated

Experimental Evaluation

1. Abstraction Level Classification
   - From 145 assertions 100 have been classified as low level assertions
2. Partitioning based on Sentence Similarity
   - 11 clusters are found
3. Partitioning based on Sentence Similarity
   - 11 SystemVerilog assertion templates were generated
   - Each is applied to all elements of the corresponding cluster

Summary

Two Requirement evaluation approaches

Summary

Two Requirement evaluation approaches

- Static Sentence Analysis

Summary

Two Requirement evaluation approaches

- Static Sentence Analysis
- Guidelines Validation

Summary

Two Requirement evaluation approaches

- Static Sentence Analysis
- Guidelines Validation

Assertions translation approach