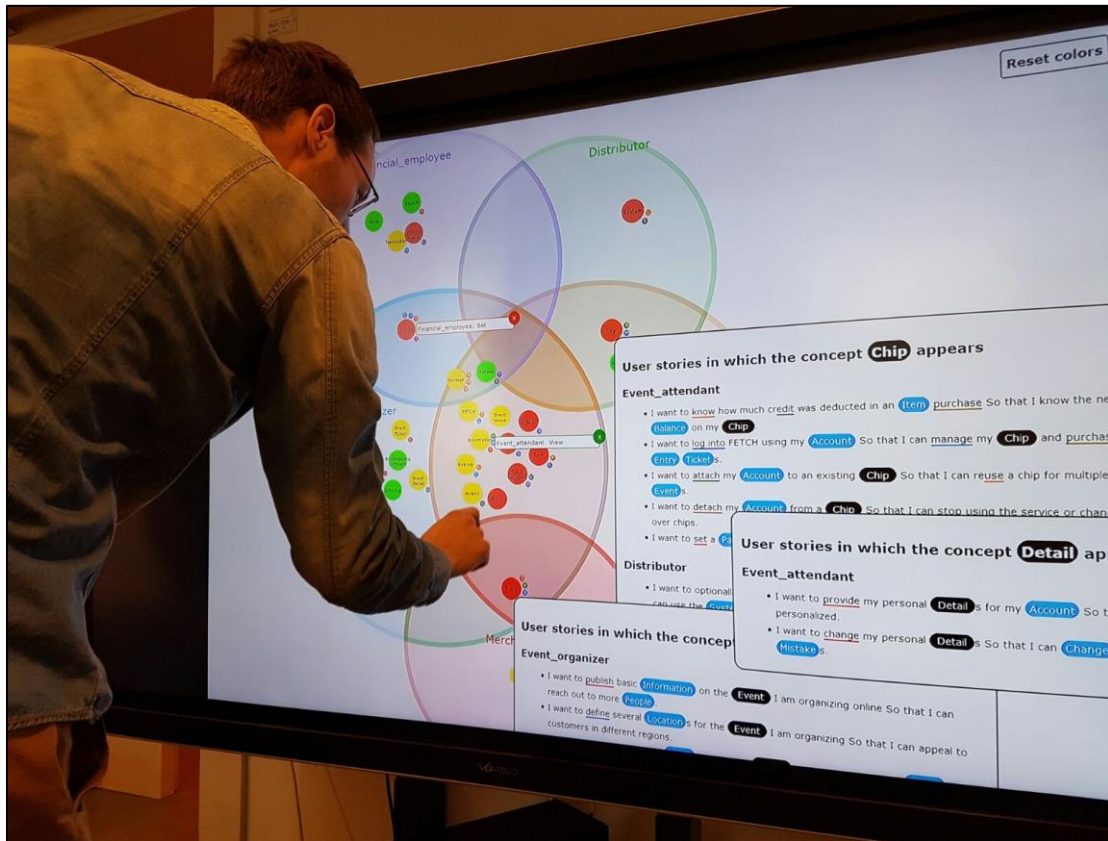


## REVV:

### A tool to create a better understanding of software requirements through Information Visualization and NLP



#### Author:

Ivor van der Schalk

#### Supervisors:

Dr. Fabiano Dalpiaz (first supervisor)

Garm Lucassen (second supervisor)

Master Business Informatics  
Dept. of Information and Computing Science

Utrecht University

2017

# List of Abbreviations

---

<b>DSF</b>	Design Science Framework
<b>IV</b>	Information Visualization
<b>NLP</b>	Natural Language Processing
<b>QUS</b>	Quality User Story
<b>RAI</b>	Requirement Ambiguity Identification
<b>REVV</b>	Requirement Engineer Verification and Validation
<b>RE</b>	Requirement Engineering
<b>SFT</b>	The Semantic Folding Theory
<b>VDAR</b>	Evaluating Visual Data Analysis and Reasoning

# List of Figures

---

<b>Figure 1</b> .....	A cave painting in Lascaux
<b>Figure 2</b> .....	The Design Science Framework
<b>Figure 3</b> .....	Starplots of assessed requirements analytics visualizations
<b>Figure 4</b> .....	Starplot of the proposed visualization
<b>Figure 5</b> .....	Conceptual framework for requirements ambiguity and incompleteness identification
<b>Figure 6</b> .....	Illustration of the details-on-demand dimension
<b>Figure 7</b> .....	Example of inspecting the user stories of a concept
<b>Figure 8</b> .....	The 7 areas (A–G) of a Venn diagram
<b>Figure 9</b> .....	Venn diagram of three viewpoints
<b>Figure 10</b> .....	The ambiguity filter range
<b>Figure 11</b> .....	Illustration of the different sizes of the visualization elements
<b>Figure 12</b> .....	Overview of the input that is received by the visualization
<b>Figure 13</b> .....	Overview of the RAI method
<b>Figure 14</b> .....	Illustration of a concept pair of which the similarity score is unknown
<b>Figure 15</b> .....	Requirements from the WebCompany data set with concepts in a correspondence state
<b>Figure 16</b> .....	Illustration of a concept pair with their context of which their semantic relatedness is unknown.
<b>Figure 17</b> .....	Overview of the correlation study independent and dependent variable.
<b>Figure 18</b> .	SPSS output of the Pearson correlation test between the ‘average human score’ and ‘concept similarity score’.
<b>Figure 19</b> .	SPSS output of the Pearson correlation test between the ‘average human score’ and ‘algorithm score’.
<b>Figure 20</b> .....	Illustration of the correlation between the ‘average human score’ and the algorithm score
<b>Figure 21</b> .....	Photo of two participants using the implemented tool on a video wall

# List of tables

---

<b>Table 1</b> .....	Literature study keyword overview.
<b>Table 2</b> .....	The 9 characteristics that determine the quality of a set of requirements
<b>Table 3</b> .....	The 6 characteristic that determine the quality of a user story
<b>Table 4</b> .....	The 13 quality criteria of the QUS framework
<b>Table 5</b> .....	The phases in the RE process
<b>Table 6</b> .....	The activities in the RE process
<b>Table 7</b> .....	The social problems causing inconsistency in software requirements
<b>Table 8</b> .....	The technical problems causing inconsistency in software requirements
<b>Table 9</b> .....	The different visualization types
<b>Table 10</b> .....	The different visualization type main groups
<b>Table 11</b> .....	The different interactive visualization types
<b>Table 12</b> .....	The Gestalt principles
<b>Table 13</b> .....	The visualization interaction techniques
<b>Table 14</b> .....	RE visualizations that employ an interactive visualization type and support the verification activity
<b>Table 15</b> .....	A description of interactive visualizations supporting the RE verification activity
<b>Table 16</b> .....	The five conceptual goals and their operational questions
<b>Table 17</b> .....	The data that is used in the proposed visualization
<b>Table 18</b> .....	Elements in the overview of the proposed visualization
<b>Table 19</b> .....	Filter features in the proposed visualization.
<b>Table 20</b> .....	Description of the shape and size employed for each element in the visualization.
<b>Table 21</b> .....	Description of the correlation study independent and dependent variable
<b>Table 22</b> .....	Overview of the data set obtained through the questionnaire
<b>Table 23</b> .....	Overview of the independent and dependent variables used in the analysis
<b>Table 24</b> .....	Overview of the ambiguities found data from the control and tool groups from session 1 & 2
<b>Table 25</b> .....	Overview of the missing user stories found data from the control and tool groups from session 1 & 2
<b>Table 26</b> .....	Overview of features missing in the tool
<b>Table 27</b> .....	Overview of the differences in the approach employed by the tool groups
<b>Table 28</b> .....	Overview of the differences between the control groups and the tool groups

# Foreword

---

The research presented in this work turned out to be an exciting yet challenging task. It gives a great feeling to build something that excites people and looks fun to use. Yet, at the same time the work, in particular the implementation, has been a challenge. Fortunately, many people stood by me throughout the research and I am very thankful to these people. In particular, I would like to thank my two supervisors. I would like to thank Fabiano Dalpiaz for his great support, feedback and excitement for the project. I would like to thank Garm Lucassen for his great feedback and support, even when he was abroad at the other side of the world for a period of time. Fabiano and Garm always made me feel I was working on something meaningful and interesting. Without their guidance and support this work would not have been possible.

Aside from the people mentioned above, I would like to thank Jan-Martijn van der Werf for providing me support in finding participants for the evaluation. I would like to thank all participants who volunteered to participate in the evaluation, and also all participants who volunteered to participate in the correlation study. Each one of them has made an important contribution to the work presented in this thesis.

Finally, I would like to thank my friends and family, who supported me throughout the research and for being the great people they are.

# Abstract

---

Current requirements elicitation techniques are sub-optimal as far as representing requirements inconsistencies and stakeholder disagreements. The literature in Requirements Engineering (RE) has shown that combining humans' cognitive and analytical capabilities with automated reasoning is an effective combination to achieve such result.

In this work we introduce a novel software tool that blends Natural Language Processing (NLP) and Information Visualization (IV) techniques with the aim of identifying potential ambiguities and missing requirements. For this purpose we have constructed a conceptual framework and built a visualization that is inspired by this framework. In addition to that, this work presents an algorithm for finding ambiguities in a set of user stories. This algorithm is evaluated in a correlation study.

The algorithm and the visualization with its incorporated state-of-the-art IV techniques are all incorporated in the implemented software tool. The usefulness of this tool for identifying ambiguities and missing requirements is assessed in an evaluation study.

# Table of Contents

1. Introduction .....	10
1.1 Problem Statement.....	12
1.2. Research Question .....	13
1.2.1 Sub research questions.....	14
1.3 Research Method .....	15
2. Literature Study .....	17
2.1 Literature Study Method.....	17
2.2 Requirement Engineering.....	18
2.2.1 Software requirements.....	18
2.2.2 The RE process.....	21
2.2.3 Consistency in RE.....	22
2.2.4 Viewpoints .....	24
2.2.5 Conceptual modeling in RE .....	25
2.3 Information Visualization .....	25
2.3.1 What is Information Visualization? .....	25
2.3.2 The Benefits of Information Visualization.....	28
2.3.3 Determining the design of a visualization .....	29
2.3.4 The state-of-the-art in Requirement Engineering visualizations .....	31
3. Identification of Requirements Ambiguity & Incompleteness via IV .....	36
3.1 A Conceptual Framework for Requirements Ambiguity & Incompleteness Identification.....	37
3.1.1 Consensus state.....	38
3.1.2 Correspondence state.....	38
3.1.3 Conflict state.....	39

3.1.4 Contrast state.....	39
<b>3.2 Interface Design Factors.....</b>	<b>39</b>
3.2.1 Data factor.....	39
3.2.2 Task factor.....	41
<b>3.3 Visualization Approach.....</b>	<b>45</b>
3.3.1 Plane.....	45
3.3.2 Color.....	46
3.3.3 Shape & size.....	46
<b>3.4 Visualization Input.....</b>	<b>47</b>
3.4.1 The tool.....	48
3.4.2 The device.....	49
<b>4. Identification of Requirements Ambiguity via NLP.....</b>	<b>50</b>
<b>4.1 The Requirements Ambiguity Identification Method.....</b>	<b>50</b>
Step 1: Create concept pairs.....	51
Step 2: Compute concept similarity.....	51
Step 3: Create concept context.....	52
Step 4: Compute concept context similarity.....	53
Step 5: Compute syntactic ambiguity score.....	53
<b>4.2 Requirements Ambiguity Identification Method Correlation Study.....</b>	<b>53</b>
4.2.1 Correlation study protocol.....	54
4.2.2 Correlation study results.....	56
<b>5. Tool Evaluation.....</b>	<b>60</b>
<b>5.1 Tool Evaluation Protocol.....</b>	<b>60</b>
5.1.1 Goal.....	60
5.1.2 Scenario.....	60
5.1.3 Evaluation question.....	60
5.1.4 Sample group.....	61



5.1.5 Data collection.....	62
5.1.6 Data preparation.....	63
5.1.7 Data analysis.....	63
5.2 Tool Evaluation Results.....	64
5.2.1 Pilot study.....	64
5.2.2 Data preparation.....	65
5.2.3 Data analysis.....	66
6. Conclusion.....	72
6.1 Conclusions of the Sub Research Questions.....	72
6.2 Conclusion of the Main Research Question.....	76
7. Discussion.....	78
7.1 Research Limitations.....	78
7.1.1 Conclusion validity.....	78
7.1.2 Internal validity.....	78
7.1.3 Construct Validity.....	79
7.1.4 External validity.....	79
7.2 Future Research.....	80
References.....	81
Appendixes.....	81
Appendix A: Detailed Explanation of Assessed Visualizations.....	86
Appendix B: The Requirements Identification Ambiguity Study Description .....	92
Appendix C: List of Concept pairs (1).....	94
Appendix D: List of Concept pairs (2).....	97
Appendix E: Correlation Study Survey.....	102
Appendix F: Evaluation Description.....	103

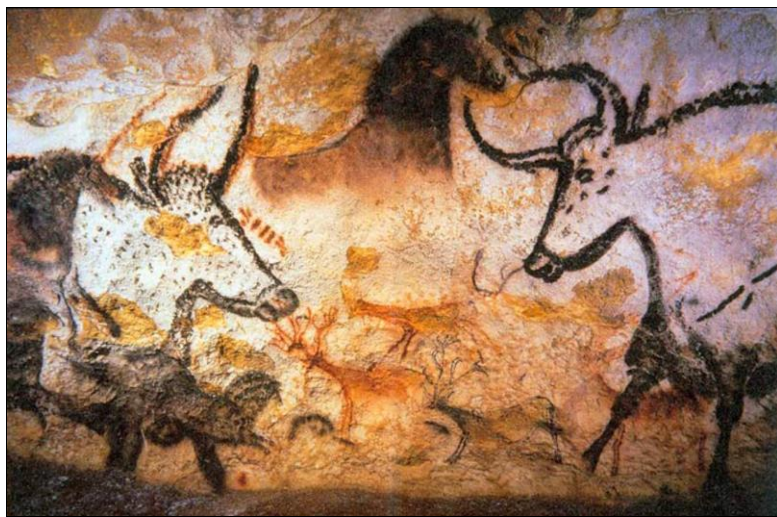
Appendix G: List of Ambiguities Tool.....	104
Appendix H: List of Ambiguities Pen & Paper .....	105
Appendix I: List of Missing User Stories .....	106
Appendix J: List of User Stories .....	107
Appendix K: List of Criteria .....	110

# Chapter 1

---

## Introduction

Since the dawn of mankind images have been used to convey information. The first of these images can be found 40,000 years ago in caves in Eurasia. These caves contain cave paintings used to visualize the world around us: it is the animals, persons and objects around us that are represented in these paintings (e.g. Figure 1).



**Figure 1** - A cave painting in Lascaux, France dating back to the prehistoric era. The painting displays an extinct species of ox that inhabited Europe, Asia and North Africa.

With the rise of civilization the world became more connected and organised. The ongoing progress of civilization demands a continuing need to express more (complex) information. At the same time new tools and methods emerge to express this information. Perhaps the greatest invention to date is written language. It is the Egyptians in 2690 BC who invented hieroglyphs: a writing system using symbols to express information through time and space. It is this system that allowed the ancient Egyptians to express instructions resulting in great artefacts such as the Pyramids and the Sphinx.

When we move forward in time to the era in which we live in now, not much has changed. When we look in the field of RE natural language is still the predominant notation that practitioners use to express requirements [1]. Software requirements are not that much different from regular requirements, except that the resulting artefact does not appear physical in the real world. It is perhaps this reason that RE is found so difficult [2]. Mankind has always used images to express what the world around him (should) look like, but creating a mental image of something that is outside the physical world is

very difficult. For this purpose many RE tools and methods exist to help establish this mental image. Much progress has been made, but at the same time many challenges remain open, and as civilization continues to progress many more challenges will follow.

In essence this thesis is the result of the seemingly never ending need to express more (complex) information through new tools and methods. This thesis focuses on creating a tool that helps to construct a mental image of a software artefact. To be more specific, various studies [3]-[4] call for the need of a new generation of visualizations to support verification and validation tasks (e.g. requirements consistency & completeness checking) in the RE process. IV allows the use of computer-supported, interactive, visual representation of abstract data to amplify cognition [5]. The purpose of IV is to offer insight [6] which makes its application well suited for RE verification and validation tasks.

User stories [7] are a popular method for representing requirements in natural language using a simple template such as: “As a {role}, I want {goal}, [so that {benefit}]” [8]. This work builds on the work of [1] who introduced a method for automatically extracting concepts and relationships from a set of user stories. These concepts and relationships are extracted through NLP: an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech [9]. In this work this information is used to transform user stories into a visualization by applying novel IV techniques. Furthermore, this work explores how NLP can be applied to find ambiguities within a set of user stories.

This research project investigates how the domain of RE can benefit from blending IV and NLP. As such, this work presents a step towards the synergistic use of NLP and human analysis in the context of user stories. This work introduces a novel software tool that blends NLP and IV techniques with the aim of identifying potential ambiguities and missing requirements in a set of user stories. This work makes four concrete contributions:

- A framework for identifying potential ambiguity and incompleteness based on RE literature about inconsistent viewpoints and conceptual modelling (**Section 3.1**);
- To aid stakeholders analyse multiple viewpoints— and identify potentially missing requirements and ambiguities— this work presents an interactive visualization that incorporates a Venn-diagram that helps to identify which concepts appearing in requirements are shared among the different viewpoints (**Section 3.2** and **Section 3.3**);
- To identify potential ambiguous requirements, this work combines state-of-the-art NLP algorithms that assess the semantic similarity between pairs of concepts in their usage context (**Chapter 4**);

- To demonstrate feasibility, this work presents a prototype based on Web 2.0 technologies and evaluates it on real-world data with requirement engineer experts (**Section 5.2**).

## 1.1 Problem Statement

In his work [10] explains “*the hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements. No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later*”. Defining a complete and consistent set of requirements is clearly a crucial step in the process of developing a software system. Requirement elicitation is a difficult process as effective communication between stakeholders is difficult to achieve and is a recurring problem in the elicitation of requirements [11]. RE Verification and validation tasks help to highlight conflicts between views, perceptions, and goals of the stakeholders, and as such helps create a mutual understanding between the stakeholders [12]–[14]. Verification and validation tasks include checking requirements for consistency and completeness [3]. Such tasks are hard because it is time-consuming for requirement engineers to analyse requirements [15]. The reason behind this can be found in the problems below:

- **Changing and unidentified stakeholders:** during a software project stakeholders often change and new stakeholders are identified leading to inconsistencies between the requirements of the stakeholders [16].
- **Changing requirements and analysts:** the terminology in which requirements are expressed often vary with the composition of team members [16]–[18].
- **Voluminous requirements:** the sheer size of a requirements document often leads to inconsistencies [16] and it quickly becomes infeasible to test its consistency [17].
- **Complex requirements:** the complexity of the domain or software specification can make it difficult to understand exactly what has been specified or how components interact. Implicit requirement dependencies often hide requirements inconsistencies [16].

IV can help to address the problems mentioned above. The usefulness of IV for identifying inconsistencies is supported by [19] who state that the application of IV can surface areas of disagreement. Furthermore, effective visualization can potentially reduce misconceptions and gaps in understanding conflicts between requirements [3]. Such knowledge discovery is possible through IV as it allows highlighting and filtering of

information that is of interest. However, manual approaches that rely solely on human intelligence are not scalable. The downside of fully automated approaches based on NLP is that they require trade-offs between precision and recall [20]–[22], also because NLP technology is still mostly at the syntactic level [23]. Combining these two approaches is therefore essential.

**Current requirements elicitation techniques are sub-optimal as far as representing requirements inconsistencies and stakeholder disagreements. Blending IV and NLP can help to identify such inconsistencies and disagreements.**

## 1.2. Research Question

The problem statement above requests a novel tool to support RE validation and verification tasks by blending IV and NLP. As such, this research is based on the following main research question:

---

*RQ: How to better understand software requirements through the application of IV and NLP?*

---

In this work we envision a tool that helps requirement engineers with RE verification and validation tasks by blending IV and NLP. As such, we name this tool the Requirement Engineer Validation and Verification (REVV) tool. This tool acts as a real-time, modern documentation tool for agile development. The input of this tool is user stories. User stories are popular among practitioners due to the fact that they are expressed in natural language, and therefore easily understood [24]. The fact that they are popular makes it easier for practitioners to start using this tool. Another reason why user stories are a good fit is because they consist of a clear and precise structure, making them suitable for NLP. This tool helps requirement engineers to perform verification and validation tasks by supporting the identification of ambiguities and missing user stories. Furthermore, this tool helps practitioners to work with a large number of user stories and facilitates effective discussion among stakeholders about software requirements. The impact of this tool will result into a set of requirements of higher quality and a better understanding of the requirements between the stakeholders. This will lead to better products overall, and ultimately drops the costs of software development. Consequently, the main goal of this thesis is to expand on the work of [1] to generate a visual interactive

representation of a set of user stories and in doing so enable more insight and better understanding between stakeholders.

### **1.2.1 Sub research questions**

The answer of the main research question will be based on the following six sub research questions. For each sub research question the relevance towards the main research question is described.

#### **SRQ 1: What problems do practitioners encounter in understanding software requirements?**

A literature study is conducted towards the problems practitioners encounter in understanding software requirements. This study provides valuable insights into the problems that should be addressed by blending IV and NLP. These problems are described in the problem statement in **Section 1.1** and further explored in **Section 2.2**

#### **SRQ 2: What types of RE visualizations already exist and what can be learned from them?**

The aim of this sub research question is to identify the state-of-the-art of RE visualizations. This literature study helps to identify useful visualization techniques for RE activities and helps to identify problems that are not yet addressed by IV. In **Section 2.3** a description is given of the state-of-the-art of IV visualizations in RE.

#### **SRQ 3: How can missing requirements be identified?**

A conceptual framework is constructed that helps to find requirements incompleteness. This framework is presented in **Section 3.1**. Furthermore, in **Section 3.2** and **Section 3.3** we explore how IV can assist in finding missing requirements.

#### **SRQ 4: How can ambiguities in requirements be identified?**

A description is given of how the framework in **Section 3.1** can be used to help to identify ambiguities in requirements. Furthermore, in **Section 3.2** and **Section 3.3** we explore how IV can help to find ambiguities in a set of requirements. Finally, a method for automatically detecting ambiguities in requirements through NLP is constructed. This method is evaluated through a correlation study. A description of this method and its evaluation are presented in **Chapter 4**.

### **SRQ 5: What criteria should the proposed visualization satisfy to identify ambiguities and missing requirements?**

A design of the visualization is constructed. Its aspects are described according to the task taxonomy of [25]. The design with its rationale are given in **Section 3.2** and **Section 3.3**.

### **SRQ 6: Does the implemented tool satisfy the expectations of a requirement engineer to help identify ambiguities and missing requirements?**

The aim of this sub research question is to investigate the usefulness of the implemented tool. This is investigated by performing an evaluation study. For this purpose the proposed tool is implemented with the visualization design presented in **Section 3.2** and **Section 3.3** and the Requirement Ambiguity Identification (RAI) method presented in **Chapter 4**. The result of this evaluation is given in **Chapter 5**.

## **1.3 Research Method**

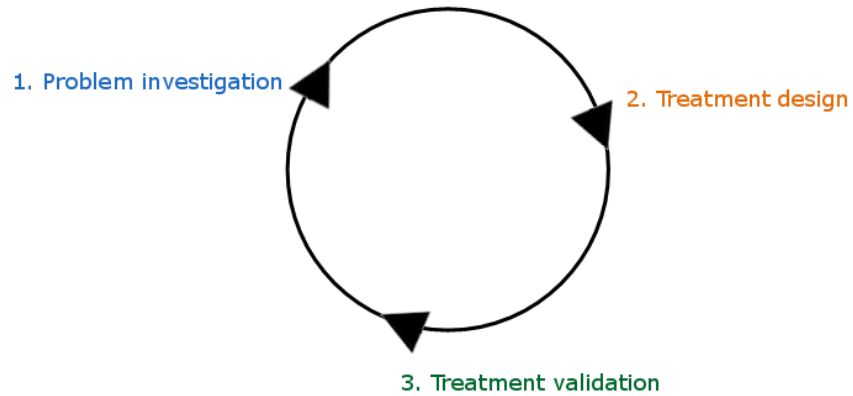
The research methods of this thesis are framed around the Design Science Framework (DSF) [26]. This framework fits well with this project since the aim of the project is to create a new artefact and acquire new knowledge. During this project a research cycle is completed in which the following tasks are conducted (Figure 2):

1. **Problem investigation:** To better understand the software requirement domain, the different problems in the software requirement domain are explored (SRQ1) through a literature study.
2. **Treatment design:** In this phase, we analyse current types of visualizations related to software requirements (SRQ2) to determine what we can learn from them. From this we learn which aspects can be incorporated in the visualization and what aspects should be added. We then conduct a literature study to investigate how ambiguities and missing user stories can be found (SRQ3 & SRQ4) and use this knowledge to construct a conceptual framework that forms the foundation of the visualization. Furthermore, to help find ambiguities we construct a method for detecting ambiguities in user stories through NLP. Once the conceptual framework and ambiguity method are constructed we define the design criteria that the proposed visualization should satisfy (SRQ5). For



determining the criteria we follow the Unified Taxonomic Framework [27]. Based on these criteria a visualization design is and developed into a working prototype.

3. **Treatment validation:** In this final phase we validate the usefulness of the RAI method through a correlation study (SRQ4). Furthermore, we validate the implemented tool with a group of experts (SRQ6). This evaluation is done through the Evaluating Visual Data Analysis and Reasoning (VDAR) scenario [28].



**Figure 2** - The Design Science Framework [26].

# Chapter 2

---

## Literature Study

The aim of this literature study is find what problems practitioners encounter in understanding software requirements (SRQ1) and also to shed light on how ambiguities in requirements can be identified (SRQ4). In addition to that we investigate what IV is and what types of RE visualizations already exist (SRQ2). This information helps to determine the criteria that the proposed visualization should satisfy to help identify ambiguities and requirements incompleteness (SRQ5), in **Chapter 3**.

**Section 2.1** presents the method that is employed in the literature study. In **Section 2.2** we dive into the world of RE. We explore what software requirements are, learn more about the process of requirement elicitation and find what problems practitioners encounter in this process. This section provides the theory on which the conceptual framework is built in **Section 3.1**. In **Section 2.3** we then go further by exploring what IV is and explore the state of the art of RE visualizations to find what we can learn from them. The knowledge that is obtained from this section is used to set the visualization criteria in **Section 3.2** and **Section 3.3**.

### 2.1 Literature Study Method

This research employs the Snowballing literature study method. In this type of literature study, articles related to the topic of research are found by reading articles and relevant citations. Google Scholar is the main source for the literature study in this research. The literature study's main focus is on answering the sub-questions needed to answer the main research question.

This research is on the intersection between IV, NLP and RE, and as such the literature study in this research is focused on these domains. In the domain of IV the literature needs to provide information on how IV can help to create an effective visualization and on how visualizations are used in the domain of RE. The literature in the domain of NLP needs to answer how NLP can be applied to help find ambiguities. Finally, the literature that is related to the RE domain should provide a better understand of the RE domain itself and answer what problems there are in the RE domain and what methods exist that can help solve such problems. Table 1 gives an overview of all the keywords that are used to find the literature in this research.

IV domain	NLP domain	RE domain
- "Information Visualization framework"	- "Natural Language Processing Requirement Engineering"	- "Requirement Engineering software requirements"
- "Information Visualization techniques"	- "Natural Language Processing software requirements"	- "Requirement Engineering ambiguity"
- "Information Visualization design"	- "Natural Language Processing user stories"	- "Requirement Engineering user stories"
- "visualization Requirement Engineering"	- "Natural Language Processing semantic similarity"	- "Requirement Engineering process"
- "visualization software requirements"	- "Natural Language Processing techniques"	- "Requirement Engineering challenges"
- "Requirement Engineering tools"		- "Requirement Engineering method"
- "software requirements tools"		

**Table 1** - Literature study keyword overview.

## 2.2 Requirement Engineering

### 2.2.1 Software requirements

This section investigates what a software requirement is. In their work [29] define requirements as followed: *"Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system"*. There are two types of software requirements [30]:

- **Functional requirements:** These requirements describe *what* developers must implement to enable users to accomplish their tasks. For example: "The passenger shall be able to print boarding passes for all flight segments for which he has checked in".
- **Non-functional requirements:** These are requirement that specify criteria that can be used to judge the operation of a system, rather than specific behaviours. They are contrasted with functional requirements that define specific behaviour or functions.

The main reason for writing software requirements is to enable an agreed understanding between stakeholders of what system is to be built [31]-[32]. The quality of a set of requirements has a major impact on the success of a software project. Various studies suggest that errors introduced during requirements activities account for 40% to 50% of all defects found in a software product [32]. There are 9 characteristics that determine the quality of a set of requirements (Table 2) [31].

Characteristic	Description
Complete	A set of requirements needs no further amplification because it contains everything pertinent to the definition of the system or system element being specified. In addition, the set contains no To Be Defined, To Be Specified, or To Be Resolved clauses.
Consistent	A set of requirements does not have individual requirements which are contradictory. Requirements are not duplicated. The same term is used for the same item in all requirements.
Affordable	A complete set of requirements can be satisfied by a solution that is obtainable/feasible within life cycle constraints (e.g., cost, schedule, technical, legal, regulatory).
Bounded	A set of requirements maintains the identified scope for the intended solution without increasing beyond what is needed to satisfy user needs.
Specific	A requirement must say exactly what is required. This includes a requirement is: <ul style="list-style-type: none"> <li>• clear i.e. that there is no ambiguity;</li> <li>• consistent i.e. that the same terminology has been used throughout the specification to describe the same system element or concept;</li> <li>• simple i.e. avoid double requirements e.g. X and Y;</li> <li>• of an appropriate level of detail.</li> </ul>
Measurable	Is it possible, once the system has been constructed, to verify that this requirement has been met?
Attainable	Is it possible to meet the defined requirement? For example a requirement that describes "the system shall be 100% reliable and 100% available" is probably not feasible;
Realisable	Is it possible to achieve the requirement given what is known about the constraints under which the system and the project must be developed? Whether the requirement is realisable depends on: <ul style="list-style-type: none"> <li>• System and physical constraints;</li> <li>• The given project resource.</li> </ul>
Traceable	Is the ability to trace (forwards and backwards) a requirement from its conception through its specification to its subsequent design, implementation and test. It is important for the following reasons: <ul style="list-style-type: none"> <li>• So that we can know and understand the reason for each requirement inclusion within the system;</li> <li>• So that we can verify that each requirement has been implemented;</li> <li>• So that modifications are made easily, consistently and completely.</li> </ul>

**Table 2** - Overview of the 9 characteristics that determine the quality of a set of requirements [31]. This work focuses on those characteristics that are colored in green.

A user story is a description containing one or more sentences that captures what a user does or needs in a system [7]. It captures the "who", "what" and "why" of a requirement in a simple, concise way. Large user stories are referred to as an epic. Epics can be split into two or more stories of smaller size. Epics typically fall into one of two categories:

- **The compound story:** A compound story is an epic that comprises multiple shorter stories;

- **The complex story:** A complex story is an epic that is inherently large and cannot easily be disaggregated into constituent stories.

In addition to epics, user stories can also be assigned to themes. A theme is a main area of focus. The INVEST framework [33] offers guidelines to create user stories of high quality. The framework contains the following characteristics: Independent, Negotiable, Valuable, Estimable, Small and Testable (Table 3).

Attribute	Description
Independent	Dependencies between user stories should be avoided. Dependencies between stories lead to prioritization and planning problems. Dependency issues can be solved by: <ul style="list-style-type: none"> <li>● Combining the dependent stories into one larger but independent story;</li> <li>● Finding a different way of splitting the stories.</li> </ul>
Negotiable	Story cards should be short descriptions of a functionality. A story that contains too many details gives the impressions of (1) false precision or completeness and (2) that there's no need to talk further about the story.
Valuable to users or customers	Stories should be valuable to either the user or customer (purchaser of the software).The best way to ensure that each story is valuable to the customer or user is to have the customer write the stories;
Estimable	Developers need to be able to estimate the size of a story or the amount of time it will take to turn a story into working code. There are three common reasons why a story may not be estimable: <ul style="list-style-type: none"> <li>● Developers lack domain knowledge;</li> <li>● Developers lack technical knowledge;</li> <li>● The story is too big.</li> </ul>
Small	Stories should have the right size because if stories are too large or too small you cannot use them in planning. Therefore epics (large stories) should be split into smaller stories while small stories should be combined into a larger story.
Testable	Stories must be written so as to be testable. Successfully passing its tests proves that a story has been successfully developed.

**Table 3** - Overview of the 6 characteristic that help determine the quality of a user story [33].

In other work [21] the authors present the Quality User Story (QUS) framework. This framework contains a set of 13 quality criteria that user story writers should strive to conform to ensure the quality of a set of user stories. A description of these quality criteria is given in Table 4.

Type	Name	Description
Syntactic	Well-formed	A user story includes at least a role and a means
	Atomic	A user story includes at least a role and a means
	Minimal	A user story expresses a requirement for exactly one feature
Semantic	Conceptually sound	The means expresses a feature and the ends expresses a rationale
	Problem-oriented	A user story only specifies the problem, not the solution to it
	Unambiguous	A user story avoids terms or abstractions that lead to multiple interpretations
	Conflict-free	A user story should not be inconsistent with any other user story
Pragmatic	Full sentence	A user story is a well-formed full sentence
	Estimable	A story does not denote a coarse-grained requirement that is difficult to plan and prioritize
	Uniform	Every user story is unique, duplicates are avoided
	Unique	All user stories in a specification employ the same template
	Independent	The user story is self-contained and has no inherent dependencies on other stories
	Complete	Implementing a set of user stories creates a feature-complete application, no steps are missing

**Table 4** - Overview of the 13 quality criteria of the QUS framework [21].

## 2.2.2 The RE process

In **Section 2.2.1** it was shown that a set of requirements needs to be complete, consistent and specific. This section describes the RE process and identifies where in this process a set of requirements is made complete, consistent and specific.

The RE process is considered to be an important phase in the software development lifecycle [34]-[35]. It is a systematic process of capturing, modelling and

documenting requirements through an interactive and cooperative approach [35]. This process contains phases and activities. The RE process contains four phases [3] (Table 5). In addition to the four phases, the RE process also contains four activities [3] (Table 6).

Context and Groundwork	Requirements Elaboration	Requirements Refinement	Requirements Specification, Management and Evolution
<ul style="list-style-type: none"> <li>- Establish the business case;</li> <li>- Scope the system;</li> <li>- Mitigate serious risks;</li> <li>- Establish process, methods, and techniques;</li> <li>- Assess feasibility.</li> </ul>	<ul style="list-style-type: none"> <li>- Prepare initial system model;</li> <li>- Document high-level organizational needs;</li> <li>- Gather stakeholder needs and constraints.</li> </ul>	<ul style="list-style-type: none"> <li>-Original artefacts are refined;</li> <li>- Interactions among diverse artefacts are identified;</li> <li>- Conflicts among requirements are resolved;</li> <li>- Stakeholders agree on a set of requirements for the system.</li> </ul>	<ul style="list-style-type: none"> <li>- Software specifications are produced from the artefacts;</li> <li>- Ensure readability and traceability of requirements;</li> <li>- Document change, or need for change is managed;</li> <li>- Modifications to accommodate corrections, environmental changes, or new objectives.</li> </ul>

**Table 5** - Overview of the phases in the RE process [3].

Elicitation, Understanding and Structuring	Modeling and Analysis	Communication and Negotiation	Verification and Validation
<ul style="list-style-type: none"> <li>- Identify stakeholders and information sources.</li> <li>- Identify system components and boundaries.</li> <li>- Perform interviews, document review, other elicitation strategies.</li> <li>- Structure requirements and RE activities.</li> </ul>	<ul style="list-style-type: none"> <li>Construct artefacts for analysis by stakeholders and developers.</li> <li>- Prepare initial models of the system, system interactions, use cases, scenarios, etc.</li> <li>- Use models and notation as drivers to prompt further elicitation.</li> </ul>	<ul style="list-style-type: none"> <li>- Document, communicate requirements based on artefact analysis.</li> <li>- Negotiate solutions to conflicts among requirements.</li> <li>- Prepare precise specifications.</li> </ul>	<ul style="list-style-type: none"> <li>- Check artefacts for consistency and completeness.</li> <li>- Ensure that requirements satisfy the intended real-world goals of the system.</li> </ul>

**Table 6** - Overview of the activities in the RE process [3]. This work focuses on the 'Verification and Validation' activity, colored in green.

### 2.2.3 Consistency in RE

In **Section 2.2.1** it was found that consistency is one of the characteristics that determines the quality of a software requirement. This section takes a closer look at this characteristic

to gain a better understanding of when and why this characteristic is violated and how it can be managed.

To maintain consistency in requirements it *“requires that no two or more requirements in a specification contradict each other”* [36]. There are several reasons why maintaining consistency between requirements of a software system is difficult. The reason behind this can be found in two types of problems [37]: (1) *social problems* and (2) *technical problems*. Table 7 gives an overview of social problems that cause inconsistency in software requirements. Table 8 gives an overview of technical problems that cause inconsistency in software requirements.

Name	Description
Conflicting stakeholder requirements	Different stakeholders have different views and interests. They often seek different requirements that cannot be mutually achieved.
Changing and unidentified stakeholder	To understand system requirements, analysts often seek new stakeholders for an ongoing project. In addition to that new stakeholders may be identified or change during a project.
Changing expectations	Stakeholders' requests and their expectations often change.

**Table 7** - Overview of social problems causing inconsistency in software requirements [16].

Name	Description
Voluminous requirements	The sheer size of a requirements document can lead to inconsistency, such as varied use of terminology. This is especially true as requirements are modified [16]. The problem of varied use of terminology is supported by [17], [18] who state the expression of requirements vary greatly in their formality and precision. Sometimes they are vague or informal while at other times they are detailed and precise. In addition to that [17] state as a set of requirements grow it quickly becomes infeasible to test their consistency.
Changing requirements and analysts	Requirements continue to evolve throughout the software development lifecycle, during this process new requirements are identified and previously stated requirements may need to be deleted [16]-[18]. In addition to that requirement concepts and their expressions vary with the composition of team members [16].
Complex requirements	The complexity of the domain or software specification can make it difficult to understand exactly what has been specified or how components interact. Implicit requirement dependencies often hide requirements inconsistencies [16].
Ill-formed requirements	Individual requirements may themselves be ill-formed or self-contradictory [17].

**Table 8** - Overview of technical problems causing inconsistency in software requirements.

Inconsistency is seen as something undesirable; however it can provide great insights. In many cases, it may be desirable to tolerate or even encourage inconsistency to facilitate collaborative working and to ensure all stakeholder views are taken into account [17]. Inconsistency also helps focus attention on problem areas, and as such may be used as a tool for learning, for directing the requirements elicitation process, and as a validation and verification tool for analysis and testing. Inconsistency contributes in highlighting conflicts between views, perceptions, and goals of the stakeholders involved in the development process [14]. Furthermore, inconsistency indicates which aspects of the



system deserve further analysis, and facilitate the exploration of alternatives in system development and the elicitation of information about the system.

On the other hand, when inconsistencies are not dealt with before the end of the requirement refinement phase, in which stakeholders agree on a set of requirements for the system, they can have severe consequences. Inconsistencies may delay and, therefore, increase the cost of the system development process, jeopardise properties related to the quality of the system (e.g. reliability, safety), and make it more difficult to maintain the system [14].

## 2.2.4 Viewpoints

**Section 2.2.3** discussed why inconsistency in software requirements should be identified. This section explores why viewpoints are useful artefacts for inconsistency identification.

Viewpoints are useful artefacts to deal with inconsistencies in software requirements [38]. A viewpoint can be seen as a combination of the idea of an “actor”, “knowledge source”, “role” or “agent” in the development process and the idea of a “view” or “perspective” which an actor maintains [38]. It is necessary to identify the different viewpoints of a system as not all requirements of a system can be identified by considering the system from a single perspective [39], [13]. Viewpoints can be distinguished into three types [13]:

- **Interactor viewpoints:** These are the viewpoint of something (human or machine) which interacts directly with the system. These are typically end-users of the system and external systems.
- **Indirect stakeholder viewpoints:** These are the viewpoint of an entity (human, role or organisation) which has an interest (stake) in the system but who does not interact with it directly. Examples include operating organisations and standards/regulatory bodies.
- **Domain viewpoints:** These represent a set of related characteristics of the domain which cannot be identified with a particular stake or interactor but which nevertheless impose requirements which are implicit in the domain. For example, requirements on a communication system may be imposed by signal propagation time in copper and optical cables.

To deal with inconsistencies the following tasks need to be performed [38]:

- **in-viewpoint checks:** check the consistency of the specification within the viewpoint;
- **inter-viewpoint checks:** check the consistency of the specification with those maintained by other viewpoints.

The inter-viewpoint check is needed as recognising different viewpoints and reconciling differences between them is an essential task for the analysis to be valid [13].

### 2.2.5 Conceptual modeling in RE

**Section 2.2.4** has shown that viewpoints are useful artefacts for identifying inconsistencies. This section investigates the usefulness of conceptual modelling for inconsistency identification.

Several researchers have conducted studies on how to extract conceptual models from requirements [1], [40]-[41]. The usefulness of conceptual models in RE is supported by [42] who states that requirements analysis can be aided by the creation of conceptual models that depict a holistic view of the system instead of relying on lengthy textual requirements documents. In their work [43] found that concept terminology and distinction play an important role to find inconsistencies in conceptual models. To help find inconsistencies in conceptual models they introduce four degrees of inconsistency categorized according to two aspects: (1) distinction referring to concepts' semantics, and (2) terminology referring to the terms in which a concept is described. Consider for example the word "bank", which can mean either a financial institution or a ground alongside a body of water. The four inconsistency states are:

1. *Consensus*: same distinction, same terminology, therefore no inconsistency. For example, the term "bank" is used twice to refer to financial institution;
2. *Correspondence*: same distinction, different terminology. For instance, both "bank" and "treasury" are used to refer to a financial institution;
3. *Conflict*: different distinction, same terminology. For instance, "bank" is used twice, once to refer to a financial institution and once to denote ground;
4. *Contrast*: different distinction, different terminology. For instance, "treasury" is used to refer to a financial institution and "bank" is used to refer to a ground.

In **Section 3.1** we extend the work of [43] by presenting a conceptual framework for ambiguity and incompleteness identification in software requirements.

## 2.3 Information Visualization





### 2.3.1 What is Information Visualization?



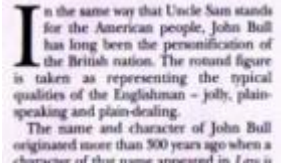
A well-known saying tells us that one picture is worth a thousand words but what exactly is IV? A visualization can be defined as "*the activity of forming a mental model of something*" [44]. A visualization uses computer supported, interactive, visual representations of data to amplify cognition of knowledge [6]. The purpose of visualization

is to offer insight [6]. The main goals of this insight are discovery, decision making and explanation. IV is useful to the extent that it increases our ability to perform these and other cognitive activities. There are different ways to visualize information. Current RE visualizations can be categorized into five generic categories of visualizations [3] (Table 9). In addition to that, visualizations can be classified into seven main groups [45] (Table 10).

Visualization type	Description	Example										
Tabular Visualizations	Typically made up of a series of intersecting columns and rows containing textual information.	<table border="1"> <thead> <tr> <th>Shoe Length</th> <th>Nbr of People</th> </tr> </thead> <tbody> <tr> <td>21 cm</td> <td>1</td> </tr> <tr> <td>24 cm</td> <td>3</td> </tr> <tr> <td>25 cm</td> <td>3</td> </tr> <tr> <td>26 cm</td> <td>2</td> </tr> </tbody> </table>	Shoe Length	Nbr of People	21 cm	1	24 cm	3	25 cm	3	26 cm	2
Shoe Length	Nbr of People											
21 cm	1											
24 cm	3											
25 cm	3											
26 cm	2											
Relational Visualizations	Consist of a collection of nodes and connectors that indicate a relationship between components, but do not describe order of operation.											
Sequential Visualizations	Similar to relational types, but intended to convey the order of operation between parts of the system or between the user and the system.											
Hierarchical Visualizations	Imply the decomposition of a system and its parts.											
Quantitative / Metaphorical Visualizations	Most commonly employed in the form of bar graphs, pie charts, or other figures conveying relative data. This type also includes more sophisticated techniques making use of visual clues such as size, shape color, or line thickness in order to convey meaning at a glance.											

**Table 9** - Overview of the different visualization types [3]. Colored in green is the visualization type that is employed in **Chapter 3**.

Visualization type main group	Description	
Sketch	<p>Sketches are atmospheric, and help to quickly visualize an idea. Sketches are effective representations to assist the group reflection and communication process by making knowledge explicit and debatable. Sketches have five strengths:</p> <ul style="list-style-type: none"> <li>• Sketches represent the main idea and key features of a preliminary study and support reasoning and arguing;</li> <li>• They are atmospheric, versatile, and universally accessible;</li> <li>• They are fast to create, and help to quickly visualize an idea;</li> <li>• They keep the attention (e.g., the use of a pen on a flipchart attracts the attention towards the communicator);</li> <li>• And they allow room for own interpretations and foster the creativity in groups.</li> </ul>	
Diagram	<p>Diagrams are abstract, schematic representations used to explore structural relationships among parts. Examples of diagrams are bar charts, tree charts and process charts; Diagrams help to:</p> <ul style="list-style-type: none"> <li>• Make abstract concepts accessible;</li> <li>• Reduce complexity;</li> <li>• Amplify knowledge;</li> <li>• Explain causal relationships;</li> <li>• Structure information;</li> <li>• and to discuss relationships</li> </ul>	
Image	<p>Images address emotions and are inspiring, appealing, motivating, and energizing. They can be impressive, expressive, or represent reality. Depending on the goal of the image it can help to:</p> <ul style="list-style-type: none"> <li>• Grab the attention (e.g., advertising);</li> <li>• Inspire (e.g., art);</li> <li>• Address emotions (e.g., advertising);</li> <li>• Improve recall (i.e., signs, visual metaphors);</li> <li>• Initiate discussions (e.g., satirical comic);</li> </ul>	
Map	<p>Maps follow cartographic conventions to reference knowledge. A map generally consists of two elements: (1) A ground layer represents the context and (2) individual elements (e.g., experts, project milestones, roads). A map helps to:</p> <ul style="list-style-type: none"> <li>• Present the overview and the details;</li> <li>• Structure information;</li> <li>• Motivate and activate employees;</li> <li>• Establish a common story;</li> <li>• Ease access to information.</li> </ul>	
Object	<p>Objects in space exploit the third dimension (3D). Objects in space are helpful to complement physical</p>	

	and digital visualizations (e.g. Augmented Reality) and to show the content from different points of view. They can help to: <ul style="list-style-type: none"> <li>• Attract recipients;</li> <li>• Support learning through constant presence;</li> <li>• Allow to integrate digital interfaces.</li> </ul>	
Interactive visualization	Interactive Visualizations allow to access, explore, and make sense of different types of information. They help to: <ul style="list-style-type: none"> <li>• Fascinate people;</li> <li>• Enable interactive collaborations across time and space;</li> <li>• Allow to represent and explore complex data;</li> <li>• Create new insights.</li> </ul>	
Story	Stories allow to transport an illustrative mental image through spoken or written language and are efficient in transferring and disseminating knowledge.	

**Table 10** - Overview of the different visualization type main groups ([45]). Colored in green is the visualization group that is employed in **Chapter 3**.

There are three types of interactive visual representations [46]. These types are classified based on the interaction a user can perform. An overview of these types can be found in Table 11.

Interactive visualization type	Description
Static representations	This type of representations contains a single, unmodifiable view in which a user is not allowed to perform any type of interaction.
Manipulable representations	In this representation a user is allowed to manipulate the process that generates the view through zooming, rotation, filtering etc.
Transformable representations	This type of representation allows a user to manipulate the input data of the representation. These manipulations usually influence and modify the visualizations that are generated.

**Table 11** - Overview of the different interactive visualization types [46]. Colored in green is the interactive visualization type that is employed in **Chapter 3**.

### 2.3.2 The Benefits of Information Visualization

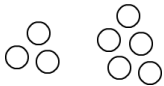
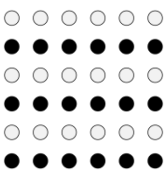
There are a number of benefits of IV compared to information expressed in Natural Language:

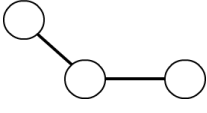
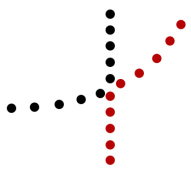
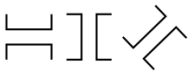

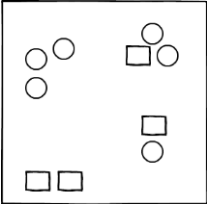
- **Comprehension of huge amounts of data:** Visualization provides the ability to comprehend huge amounts of data [47];

- **Full picture of complete data:** Visualization allows the perception of patterns or properties that would otherwise not be noticed [47]. This claim is supported by [48] who found in their study data analysts quickly discovered properties of their data that were not clearly visible or straightforwardly obtainable with other tools;
- **Reveal problems with data:** Visualization often enables problems with the data to become immediately apparent. A visualization commonly reveals things not only about the data itself but also about the way it is collected [47].
- **Facilitation of large- and small-scale features:** Visualization facilitates understanding of both large-scale and small-scale features of the data. It can be especially valuable in allowing the perception of patterns linking local features [47];
- **Facilitation of hypothesis formation:** Visualization facilitates hypothesis formation. For example a bar chart could show an upward or downward trend which could lead to questions about why the chart is showing this trend [47];
- **Integrating different perspectives:** Visualization can balance participation and reduce the dominance of certain participants [49]. It can surface areas of disagreement [19];
- **Flexibility in handling data:** it was found by [48] that data analysts appreciated being able to quickly include or exclude data for visualization, so they could concentrate on certain portions of data.

### 2.3.3 Determining the design of a visualization

The Gestalt theory explains the important principles followed by a person’s visual system when it tries to understand an image [47]. Using these principles helps to convey information that the visualization intends to express. A description of these principles is given in Table 12.

Principle	Description	Illustration
Proximity	Things that are close together are perceptually grouped together.	
Similarity	Similar elements tend to be grouped together.	
Connectedness	Connecting different graphical object by lines is a	

	powerful way of expressing that there is some relationship between them.	
Continuity	Visual elements that are smoothly connected or continuous tend to be grouped.	
Symmetry	Two symmetrically arranged visual elements are more likely to be perceived as a whole.	
Closure	A closed contour tends to be seen as an object.	
Relative size	Smaller components of a pattern tend to be perceived as objects whereas large ones as a background.	

**Table 12** - Overview of the Gestalt principles [47].

Visualization design is also influenced by the tasks that it needs to support. There are several categories of interaction techniques widely used in IV [50] (Table 13).

Interaction technique	Description
Select	When too many data items are presented on a view, or when representations are changed, it is difficult for users to follow items of interest. Allowing users to mark something as interesting makes it easier for them to keep track of interesting items even in a large data set and/or with changes in representation.
Explore	Users typically examine a subset of the data to gain understanding and insight, and then they move on to view some other data. Explore interactions do not necessarily make complete changes in the data being viewed; this allows a user to stay on the same screen while viewing new data items as others are removed.
Reconfigure	This technique allows a user to change the way data items are arranged or the alignment of data items in order to provide different perspectives on the data set.
Encode	This technique enables a user to alter the fundamental visual representation of the data including visual appearance (e.g., color, size, and shape) of each data element. Changing object size, font, color, orientation and size in a representation are examples of this technique. Completely changing how the data is represented (e.g., changing a pie chart to a histogram) is also an example of this technique.
Abstract/Elaborate	This technique provides a user with the ability to adjust the level of abstraction of a data representation. These types of interactions allow users to alter the representation from

	an overview down to details of individual data cases and often many levels in-between. An example of this technique is zooming which allows details to come more clearly into focus or fade away into context.
Filter	This technique enables a user to change the set of data items being presented based on some specific conditions. In this type of interaction, users specify a range or condition, so that only data items meeting those criteria are presented.
Connect	This technique is used to: (1) highlight associations and relationships between data items that are already represented and (2) show hidden data items that are relevant to a specified item.
Undo/redo	This technique allows a user to go backward or forward to pre-existing system states (e.g., undo, redo, history, and reset).
Change configuration	This technique allows a user to change various configurations and settings of a system.

**Table 13** - Overview of visualization interaction techniques [50].

### 2.3.4 The state-of-the-art in Requirement Engineering visualizations

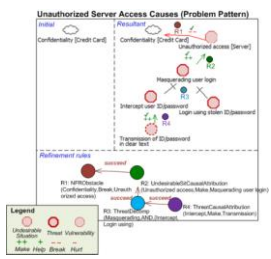
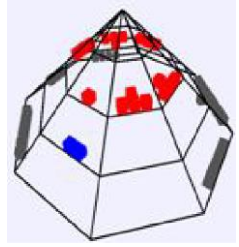
This section explores the state-of-the-art of RE visualizations. This helps to create a better understanding of why IV is useful for RE and what opportunities are still open to be addressed in this thesis. In the work of [4] the authors have categorized existing RE visualizations among different dimensions. Among these dimensions are the visualization type dimension and the visualization activity dimension. The visualization type refers to what type of visualization is employed to convey information. The visualization activity refers to what RE activity the visualization supports.

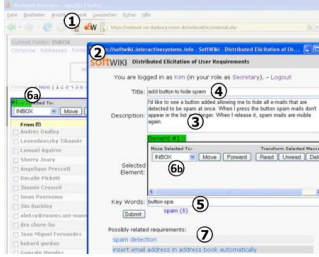
Table 14 displays an overview in which the visualization type and visualization activity dimensions are merged into one model. Since this thesis focuses on creating an interactive visualization that is used to support the RE verification & validation activity we take a closer look at the visualizations that fall into both these dimensions. From these visualizations we select three visualizations from which we believe we can learn the most in regards to the development of the visualization we present in this work. A description of each of these three visualizations is given in Table 15.



		Visualization type					
		Image Sketch	Diagram	Map	Object	Interactive visualization	Story
RE activity	Elicitation						
	Modelling						
	Communication						
	Verification					[51], [52], [53], [54], [55]	
	Evolving						

**Table 14** - RE visualizations that employ an interactive visualization type and support the verification activity. The work of [4] contains references that belong to the other cells.

Paper title	Description	Illustration
Visualizing Non-Functional Requirements Patterns [51]	A model-based tool for capturing knowledge on non-functional requirements patterns by visualizing them and the relationships between them with the goal to reuse such patterns. The notation used to express goals is relative similar to the notation used in goal-modelling [56].	
Visualizing Requirements in Distributed System Development [52]	A tool for visualizing a large number of requirements at several levels of granularity, from the perspectives of different stakeholders in a project through a 3D object. Each face of the 3D object represents a stakeholder. The color and level in which a requirement is placed indicates the priority of the requirement. This tool helps to identify whether there are conflicts in the distribution of requirements among the different stakeholders and as such verifies whether the existing requirements distribution is correct.	

<p>Involving End Users in Distributed Requirements Engineering [55]</p>	<p>An integrated web tool for enabling users to express their ideas on how the interaction with a system could be improved. The user input is contextualized, allowing for structured means to access, explore and analyse the user requirements. This tool helps to verify whether the implemented features of a system fill the needs of an end-user.</p>	
---	---	--

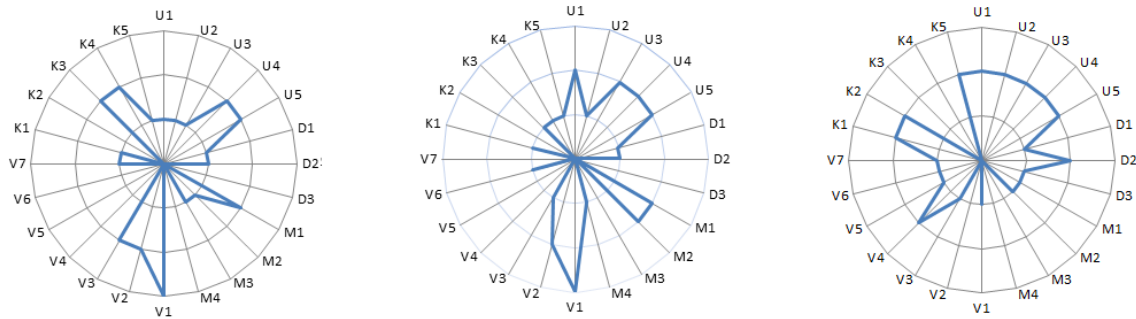
**Table 15** - A description of interactive visualizations supporting the RE verification activity.

In the work of [57] the authors present an approach to assess requirements analytic visualizations. In this approach a visualization is assessed by looking at how well it supports the conceptual goals and the associated operational questions. An overview of these conceptual goals and their associated operational questions can be found in Table 16.

User		Data		Model		Visualization		Knowledge	
U1	Multiple stakeholder roles	D1	Large-scale inputs	M1	Explicit model representation	V1	Multiple views	K1	Anomaly detection
		D2	Heterogeneous input types	M2	Automatic model construction	V2	Inter-view navigation	K2	Detailed explanation
U2	Usage without heavy pre-training	D3	Automatic pre-processing	M3	Model extension and customization	V3	Browsing	K3	Hypothesis-based reasoning
						V4	Searching		
U3	Real-time performance	D3	Automatic pre-processing	M3	Model extension and customization	V5	Query-drilling	K4	Scenario-based reasoning
						V6	Filtering		
U4	Integration into existing software development environment	D3	Automatic pre-processing	M3	Model extension and customization	V7	Annotation	K5	Actionable decision
U5	Practitioner-oriented guidelines	D3	Automatic pre-processing	M3	Model extension and customization			K5	Actionable decision

**Table 16** - The five conceptual goals and their operational questions to be addressed by a visual requirements analytics approach [57].

Each of these visualizations are assessed by using the visual requirements analytics approach [57]. Figure 3 displays a starplot for each of the three assessed visualizations. In these starplots a score is given between 0 and 3 for how well the visualization supports each operational question. An explanation on how each score is determined is given in **Appendix A**.



**Figure 3** - Starplots of the assessed requirements analytics visualizations. From left to right the starplots belong to [51], [52] and [55].

From the starplots in Figure 3 it is evident that each of the three assessed visualizations score low in the visualization area. This finding is in line with the findings of [57]. The reason behind this may be that it is labour-intensive to develop tools with highly sophisticated visualization functionalities. In recent years a number of Javascript visualization libraries have been developed [58], [59] which open the doors to create web tools with sophisticated visualization functionalities.

The model area is another area in which each of the three visualizations score low. Two of the three visualizations ([51], [55]) score low on M2 (Automatic model construction). This may be because it is difficult to create a model without the interference of a person, especially when the source of the information on which the model is build does not already exist in some way outside the mind of a person. To create a visualization that scores high on M2 it is therefore essential that the visualization is presented with a source of information that can be automatically processed by a system and transformed into a model. Each of the three visualizations also score low on M3 (Model extension and customization). Two of the three visualizations ([52], [55]) display their model mostly static. In these visualizations it is only possible to delete elements from the model. In addition to that in all of the visualizations it is not possible to customize the model by for example changing the colors or shapes of elements. The visualizations also score low on M4 (Model traceability). Each of the visualizations focus on constructing a model based on information of the present and seem to discard information from the past.

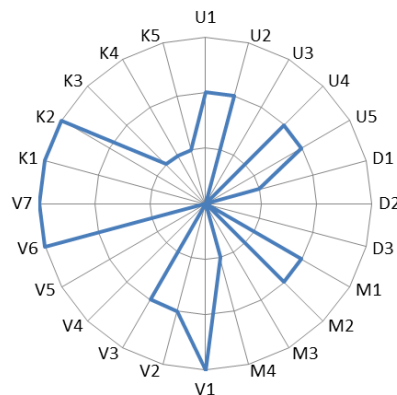
Finally, all visualization score quite low in the knowledge area. For example two of the three visualizations ([51], [52]) score 0 points on K2 (Detailed description). These two

visualizations display models without giving the user detailed information about the elements that are displayed. To score high on K2 it is therefore important to provide the user with context on what the elements displayed in the model are about. These two visualizations also score low on K5 (Actionable decision). Both visualizations give insight in the current situation but they do not actively present the user with a decision to make. The main concern with the investigated RE visualizations is that each of them present their information mostly static and do not allow to convert the displayed information into knowledge.

# Chapter 3

## Identification of Requirements Ambiguity & Incompleteness via IV

This chapter describes how the application of IV can help to identify requirements ambiguity and incompleteness from a (large) set of user stories. A list of criteria has been defined in **Appendix K** to help identify the key areas that the proposed visualization should support. This list has been set up according to the visual requirements analytics approach of [60]. Figure 4 contains a starplot based on the defined criteria that indicates to what degree the different key areas are supported by the proposed visualization. Table 16 contains a description of the key areas that are discussed in the list of criteria.



**Figure 4** – A starplot of the proposed visualization based on the criteria list given in **Appendix K**.

Based on the defined criteria the visualizations aspects are given in the following sections: **Section 3.1** presents a conceptual framework for requirements ambiguity & incompleteness identification. This framework is the foundation of the visualization aspects that are presented in this chapter. This visualization is constructed according to the guidelines of the IV taxonomic framework presented in [27]. Using this framework, **Section 3.2** presents the design criteria used in the proposed visualization. Furthermore, **Section 3.3** presents the visualization approach of the proposed visualization. Finally, **Section 3.4** describes the input that the proposed visualization receives.

## 3.1 A Conceptual Framework for Requirements Ambiguity & Incompleteness Identification

**Section 2.2.5** shows that conceptual models are useful artefacts for finding inconsistencies. This chapter presents a conceptual framework for requirements ambiguity and incompleteness identification based on the concept state theory [43]. As shown in **Section 2.2.3** reconciling differences between viewpoints (*inter-viewpoint checking*) is an important task for requirement analysis. The conceptual framework presented in this section supports this task and ultimately helps to identify potential requirements ambiguity and incompleteness.

A requirement is ambiguous when it has more than one valid interpretation [61]. Ambiguity occurs between viewpoints when a requirement set contains at least one concept in a correspondence or conflict state. The reason behind this is because requirements that contain concepts in these states are interpreted differently by the viewpoints in which they appear.

Figure 5 provides an overview of the conceptual framework (that extends [43]) that defines potential requirements ambiguity and incompleteness starting from inconsistency states. We illustrate the framework in the following sub sections with the help of the real-world user stories below from the WebCompany data set ([1]) (concepts are emphasized in the text):

- U1 As a *Visitor*, I am able to view the *media gallery* so that I can see interesting *photos* about the *event region*.
- U2 As an *Administrator*, I am able to edit existing *media elements* of a particular *gallery*, so that I can update the *content*.
- U3 As a *User*, I am able to add *content* to the selected *profile*.
- U4 As a *Visitor*, I am able to use the *contact form*, so that I can contact the *administrator*.

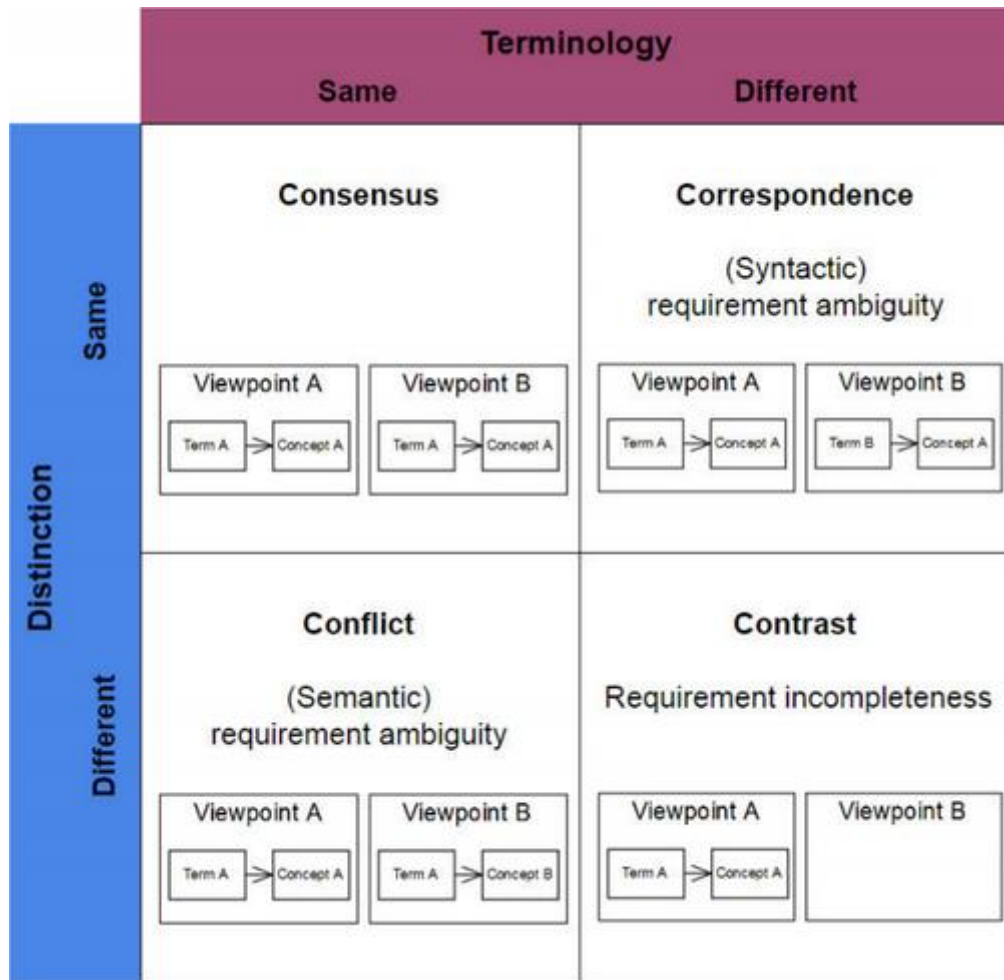


Figure 5 - Conceptual framework for requirements ambiguity and incompleteness identification (extending [43]).

### 3.1.1 Consensus state

The consensus state does not lead to any ambiguity. For example, the concept *Visitor* appears in both in U1 and U4 to refer to a visitor of the website.

Requirements that are in a consensus state are most likely not problematic. However it can be useful to identify them as it will reveal all requirements that are not in a consensus state. All requirements that are in a non-consensus state contain potential requirements ambiguity or incompleteness.

### 3.1.2 Correspondence state

Syntactic ambiguity *may* occur in the correspondence state: two different terms are used to refer to the same concept. The terms *media gallery* in U1 and *gallery* in U2 are likely to refer to the same concept, and this may make those requirements ambiguous.

Having requirements in a correspondence state makes it harder to understand different viewpoints. It is therefore important that requirements that use different terminology for different concept are changed into requirements using the same

terminology. This is necessary to make the transition from a requirement in a correspondence state to a requirement in a consensus state.

### **3.1.3 Conflict state**

Semantic ambiguity *may* occur in the conflict state: the same term is used to refer to different abstractions. In U2 the term *content* refers specifically to a *media element* while in U3 the term *content* refers to either a *text, description, image, video* or *audio fragment*.

Having requirements in a conflict state makes it harder to interpret different viewpoints correctly. It is therefore important that requirements that contain the same terminology to refer to different concepts are changed into requirements with different terminology for different concepts. This is necessary to make the transition from a requirement in a conflict state to a requirement in either a consensus state or a contrast state.

### **3.1.4 Contrast state**

Incompleteness *may* occur in the contrast state, i.e., when one viewpoint refers to a concept that does not appear in another viewpoint. U4 includes a *contact form* that the *visitor* uses to contact the *administrator*. However, there is no user story that specifies how the *administrator* can respond to this action.

Having concepts in a contrast state are problematic when other viewpoints contain (unknown) requirements that depend on these concepts as it makes the execution of these requirements impossible. For this reason it is important to add such concepts in viewpoints in which they are necessary but do not appear. This is necessary to make the transition from a requirement in a contrast state to a requirement in a consensus state.

## **3.2 Interface Design Factors**

The interface design factors for the proposed visualization that this section presents are influenced by the conceptual framework presented in **Section 3.1**. There are a number of factors that need to be considered in regards to the design aspects of the intended visualization [27]. In the following sections we discuss the data and the task factor.

### **3.2.1 Data factor**

This section describes the data that is used for creating the visualization. It is important to have an understanding of the data as it helps to know how it can be used to visualise the



information that we intend to display. This step is necessary to transform the data in the visualization into useful knowledge.

Table 17 gives an overview of the data types and relationships that are used in the proposed visualization. From this table it is clear that the data the proposed visualization uses consists of three different data types (user story, actor concept and non-actor concept) and three different data relationships (association, concept similarity and user story relatedness).

	Example	Description
<b>Data type</b>		
User story	As a <i>Visitor</i> , I am able to view the <i>media gallery</i> so that I can see interesting <i>photos</i> about the <i>event region</i> .	In his book Cohn (2004) explains a user story is a description that captures what a user does or needs in a system. A set of user stories is the data input for the tool proposed in this work.
Actor concept	Visitor	The “actor” concept represents the role in a user story. The role indicates <i>who</i> can execute a user story.
Non-actor concept	Media Gallery	The “non-actor” concept represents all concepts in a user story that do not describe a role. These concepts usually indicate <i>what</i> is used when executing a user story.
<b>Data relationship</b>		
Association	view(visitor, media gallery)	Association relationships are relationships between two concepts. This type of relationship indicates who can use what.
Concept similarity	similarity(media gallery, gallery)	Concept similarity relationships are relationships between two concepts. The relationship indicates how similar two concepts are. The relationship is expressed in a similarity score. The higher the score the more similar the two concepts are.
User story relatedness	relatedness( As a <i>Visitor</i> , I am able to view the <i>media gallery</i> so that I can see interesting <i>photos</i> about the <i>event region</i> , As an <i>Administrator</i> , I am able to edit existing <i>media elements</i> of a particular <i>gallery</i> , so that I can update the <i>content</i> )	Requirement relatedness relationships are relationships between two user stories. The relationship indicates how related two user stories are. The relationship is expressed in a relatedness score. The higher the score the more related the two user stories are.

**Table 17** - Overview of the data that is used in the proposed visualization.

### 3.2.2 Task factor



This section presents a description of the tasks that the proposed visualization supports. A task describes what the user aims to achieve and how the user achieves it using all or part of an interface's functionality [27]. A visualization can be described according to seven task dimensions [25]:



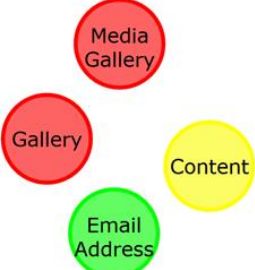
- Overview;
- Zoom;
- Filter;
- Detail on demand;
- Relate;
- History;
- Extract.

Each of the task dimensions that are supported by the proposed visualization are discussed in the following sub sections.

#### 3.2.2.1 Overview dimension

The overview dimension describes the view of the total collection [27]. Table 18 contains an overview of all the elements that can be found in the overview of the proposed visualization. In this table a description and illustration of each element is given. The description also indicates whenever one of the Gestalt theory principles, as described in **Section 2.3.3** is applied.

Overview task name	Description	Example
Overview of role concepts	For a requirement engineer it is important to know what roles appear in a set of user stories. The proposed visualization therefore displays all role concepts as role nodes in the overview. Each role node is displayed as a large circle and has a unique color and text label.	
Overview of non-role concepts	For a requirement engineer to gain an understanding of what a set of user stories is about it is important to know what concepts are used within the user stories. The proposed visualization therefore displays all non-role concepts as concept nodes in the overview. Each concept node is displayed as a circle and has a unique text label. These concepts also have a color based on the ambiguity score that they share with other non-role concepts.	

<p>Overview of association relationships</p>	<p>For a requirement engineer to gain an understanding of the relationship between a role and a concept it is important to know what action the stakeholder can execute in relation to the concept.</p> <p>The proposed visualization therefore displays all association relationships as association icons in the overview. An association icon is displayed as a small circle. To convey what association relationship an icon represents the visualization employs the similarity principle: each icon contains the color and text label (containing the first character of the association relationship name) that is related to the association relationship that it represents. In the given example on the right the association icon is colored blue, containing the character 'V' representing the association relationship 'View'. To convey to which concept an association icon belongs to we apply the proximity principle: association icons are always displayed next to the concept they belong to.</p>	
<p>Overview of viewpoints</p>	<p>For a requirement engineer to gain an understanding of the view a role has on the system it is important to know what concepts and relationships are part of the view of the associated stakeholder.</p> <p>The proposed visualization therefore employs the closure principle: it positions all concepts that are related to the role within the role concept. The combination of all the concepts nodes and their association icons within the role node represents a viewpoint.</p>	
<p>Overview of ambiguous concepts</p>	<p>For a requirement engineer to gain an understanding of which concepts in a set of user stories are syntactic ambiguous it helps to know which concepts are semantic similar.</p> <p>The proposed visualization therefore employs the similarity principle: every concept node is colored green, yellow or red color. More information about the meaning of the colors can be found in <b>Section 3.3.2</b>. In the example given in the right the concepts <i>Gallery</i> and <i>Media Gallery</i> are highly ambiguous, while the concept <i>Content</i> shares some ambiguity with other concepts and the concept <i>Email Address</i> does not share any ambiguity with other concepts.</p> <p><b>Note:</b> The ambiguity score is computed by using the Requirement Ambiguity Method (RAI) method presented in <b>Chapter 4</b>.</p>	
<p>Movable view</p>	<p>For a requirement engineer it is useful to browse through the view and center on specific elements of interest. Moving the view can be done by using the mouse (while holding the left mouse button) or by using the hand on a touch-screen display.</p>	

**Table 18** - Elements in the overview of the proposed visualization

### 3.2.2.2 Zoom dimension

Users typically have an interest in some portion of the data set [25]. In the proposed visualization the user therefore has the ability to zoom in and out on items by using the

mouse scroll wheel or by using the zoom in and out functionality on a touch-screen display.

### 3.2.2.3 Filter dimension

Filtering out uninteresting items in the data set is one of the key ideas in IV [62]. This section describes how the proposed visualization can be used to remove unwanted items from the displayed data set. All filter features can be found in the filter menu of the proposed visualization. This menu can be found in the proposed visualization on the left of the screen. Table 19 contains an overview of all the filter features that can be used in the proposed visualization. In this table a description and illustration of each feature is given.

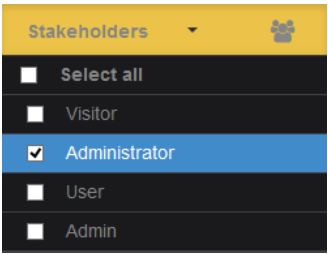
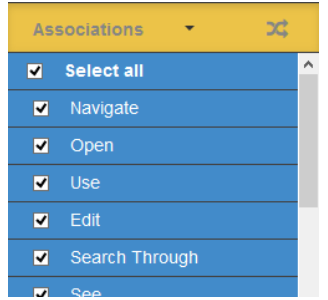

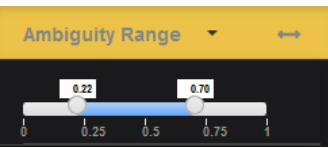
Overview task name	Description	Illustration
Viewpoint filter	Filtering viewpoints is useful as it allows to focus on viewpoints of interest. In the filter menu the requirement engineer can choose whether a viewpoint should be filtered by checking the associated check box.	
Association relationship filter	Filtering association relationships helps to focus on specific association relationships. In the filter menu the requirement engineer can choose whether association relationships should be filtered by selecting the associated check box.	
Concept state filter	Filtering concepts that belong to a certain concept state helps to find ambiguous and missing user stories as these user stories contain concepts that appear in a certain state (Section 3.1). In the filter menu the requirement engineer can choose whether concepts in a certain concept state should be filtered by selecting the associated check box.	
Concept ambiguity filter	The concept ambiguity range can be used to filter concepts that do not fall within the given ambiguity score range. This is useful as it allows to focus on concepts that share a certain ambiguity score. In the filter menu the requirement engineer can select the ambiguity range by moving the left button to indicate the minimum ambiguity score a concept should have with other concepts. The right button can be used to indicate the maximum ambiguity score a concept should have with other concepts.	

Table 19 - Filter features in the proposed visualization.

### 3.2.2.4 Detail on demand dimension

This section describes what features can be used to get the details of a selected group, sub-group or item. Within the proposed visualization details are requested by using the mouse to click on an element, or by using the finger to touch an element on a touch-screen display.

Figure 6 illustrates how the association relationship and concept ambiguity details are displayed. On the left side, the association relationship inspection reveals that the icon represents the ‘view’ association relationship, and that this association relationship belongs to the Visitor viewpoint. On the right side, the ambiguity score of the concept *media gallery* is inspected, revealing that high ambiguity is due to the concept *gallery*.

Figure 7 exemplifies detailing the user stories of a concept, showing a pop up that contains all user stories in which the concept *gallery* appears. Within the pop up the concept that is selected is colored in black and made bold. This makes it easy to identify where within the context the selected concept appears. All other concepts are colored blue and can be clicked on. Clicking on such a concept displays a new pop up of the clicked on concept. This makes it easy to compare the contexts of different concepts.

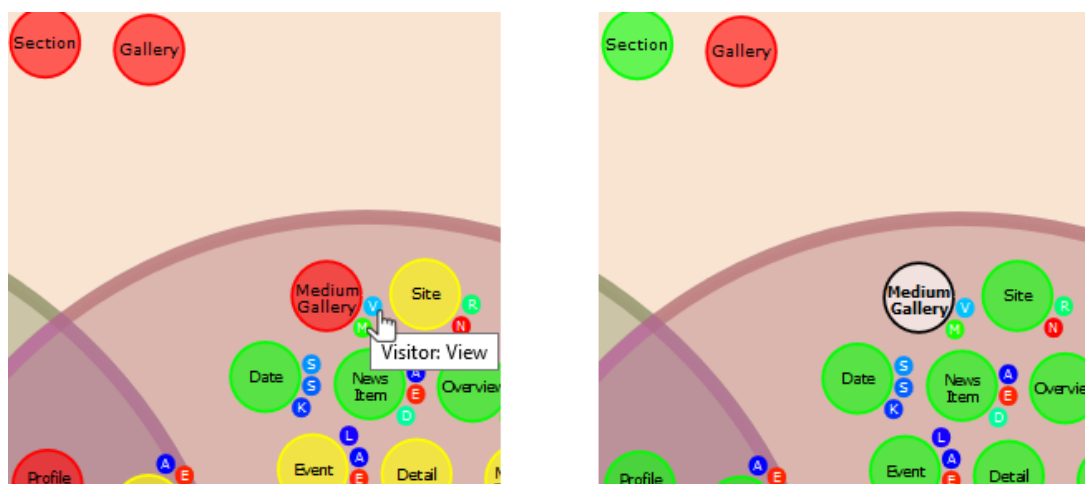


Figure 6 - Illustrations of details-on-demand for the association relationship view (left) and ambiguity details of the concept *media gallery* (right).

Gallery
X

### User stories in which the concept **Gallery** appears

**Administrator**

- I'm able to remove existing **Media Elements** of a particular **Gallery** So that I can keep the **Album** up to date
- I'm able to edit existing **Media Elements** of a particular **Gallery** So that I can update the **Content**
- I'm able to add new **Media Elements** to the select **Gallery**

Figure 7 - An example of inspecting the user stories of a concept.

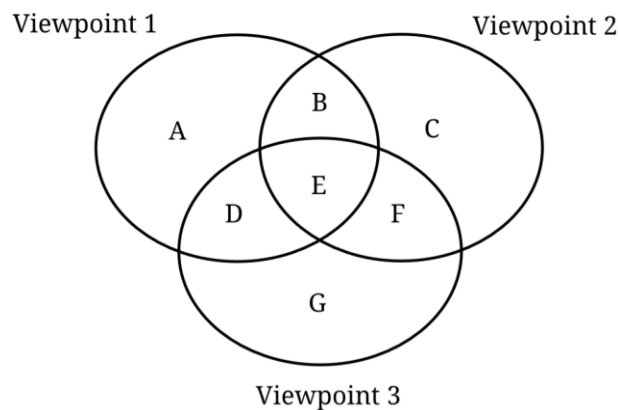
### 3.3 Visualization Approach

This section discusses the IV techniques that are employed to present the requirement engineer with information. The proposed visualization conveys its information as text and graphics. A hybrid is most suitable for our use case as text representation allows a high level of definition while pictorial representation allows the ability of pattern spotting and visual analysis [27]. There are eight aspects to describe a display on a device [27]. In the following sections we describe those aspects that play a significant role in the proposed visualization.

#### 3.3.1 Plane

The plane aspect describes the coordinates that identify the position of displayed components [27]. The visualization that is presented in this work is based on a Venn diagram [63]. A Venn diagram is well suited to display a set of data containing overlapping elements as it applies the proximity and closure principles to group elements. This makes it well suited for our use case since representing viewpoints through a Venn diagram enables a requirements engineer to examine how the concepts of multiple unique stakeholders interrelate. This section describes how the use of a Venn diagram determines the position of a concept element and how it helps to recognize its concept state.

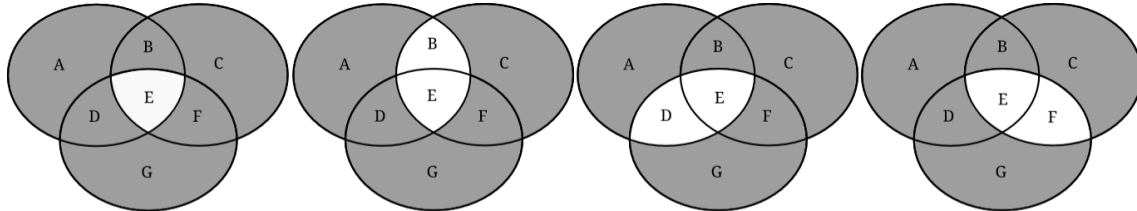
Consider the Venn diagram in Figure 8 made up of three unique stakeholder viewpoints, whose intersection produces 7 areas (A–G). Areas A, C, G includes concepts that appear in a single viewpoint. Area E contains the concepts that are shared by all three viewpoints. Areas B, D, F include the concepts that appear in exactly two viewpoints.



**Figure 8** - The 7 areas (A–G) of a Venn diagram representing three viewpoints.

As shown in Figure 9, concepts that appear in areas shared by different viewpoints are in either a consensus or conflict state. Similarly, concepts that appear in areas not

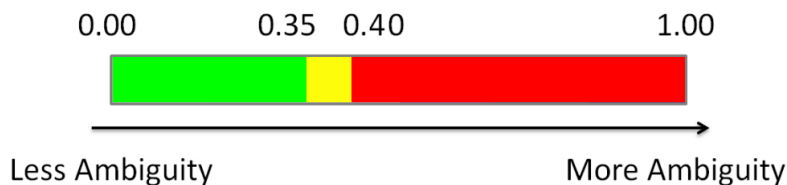
shared by different viewpoints are in either a correspondence or contrast state. More information on how the concept state of a concept is determined can be found in **Section 3.1**. When comparing all three viewpoints, the concepts in area E are in either a correspondence or conflict state while the concepts in area ABCDFG are either in a correspondence or contrast state. When comparing viewpoints 1 2, 1 3 and 2 3, the concepts in correspondence or contrast state are those in areas ACDFG, ABCFG and ABCDG, while the concepts in the other areas are in in either a consensus or conflict state.



**Figure 9** - Venn diagrams of three viewpoints in which 7 unique areas can be identified. The white areas denote either consensus or conflict state. The dark areas denote either correspondence or contrast state.

### 3.3.2 Color

This section describes the colors that are employed in the visualization. Each role node is displayed with a unique color. Each Concept node is displayed in a red, yellow or green color. A red color indicates concepts are highly ambiguous. A yellow color indicates the concepts are somewhat ambiguous. A green color indicates the concepts are not ambiguous. The color is determined by the ambiguity score obtained from the RAI method, described in **Chapter 4**. The ambiguity filter range illustrated in figure 8 is based on the results obtained from **Section 4.2** in which the RAI method is evaluated through a correlation study.

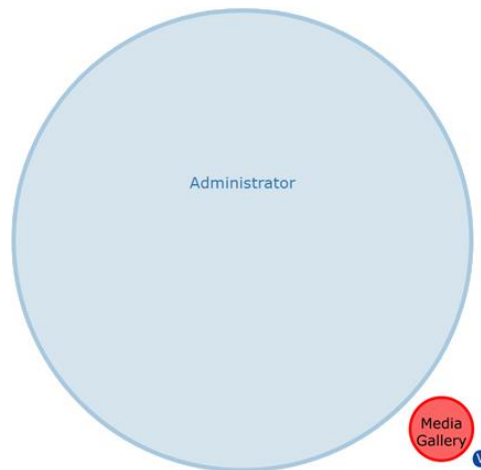


**Figure 10** - The ambiguity filter range: the closer the score of a concept pair is to 1, the more ambiguous they are (see **Chapter 4**). Scores below 0.35 are green (unlikely ambiguity), scores between 0.35 and 0.40 are yellow (possible ambiguity), and scores above 0.40 are red (probable ambiguity).

### 3.3.3 Shape & size

This section discusses the shape and size of the elements that are displayed within the proposed visualization. The elements that are displayed in the visualization are the: role node, concept node and association icon. To convey that the different types of elements represent different things the proposed visualization applies the relative size principle, as

described in **Section 2.3.3** More details about the shape and size of different type of elements are given in Table 20. **Error! Reference source not found.** gives an impression of each element’s size relative to the other elements.



**Figure 11** – Illustration of the different sizes of the elements displayed in the visualization. From left to right: the role node, concept node and association icon.

Element name	Shape	Size	Description
Role node	Circle	Relative	A role node is represented as a large circle. Its diameter is determined by a built-in algorithm in the Venn.js library [64]. This algorithm determines the optimal size of a role node based on the size of the screen and the number of concept nodes that belong to a role node. When the visualization is opened the combination of all the role nodes always fit the entire screen.
Concept node	Circle	Absolute	A concept node is represented as a circle with a diameter of 38 pixels, and is thus absolute. A circle with a diameter of 38 pixels is well suited since it is large enough to display two words with a font size that is readable on a PC and video wall.
Association icon	Circle	Absolute	An association icon is represented as a small circle with a diameter of 10 pixels, and is thus absolute. A circle with a diameter of 10 pixels is well suited since it is large enough to display 1 character with a font size that is readable on a PC and video wall.

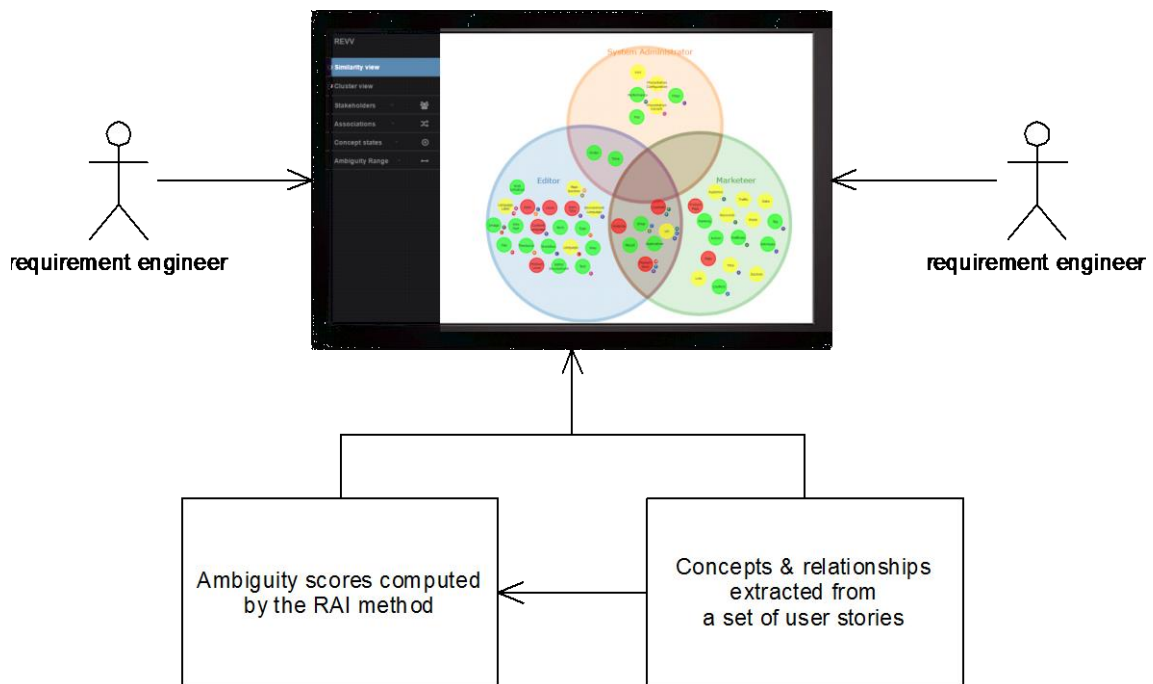
**Table 20** - A description of the shape and size employed for each element in the visualization.

### 3.4 Visualization Input

This section discusses the input that is associated with the proposed visualization. Figure 12 illustrates the input that the proposed visualization receives. An interactive visualization typically supports user interaction. The proposed visualization is meant to be used by two or more requirement engineers. These users are able to give the proposed visualization input by using the zoom, filter and detail-on-demand features as described in **Section 3.2** The proposed visualization updates the information displayed on the screen according to the input that it receives from its users.



There are two factors that impact the input process: the Tool and the Device [27]. These two factors are now discussed in more detail.



**Figure 12** – Illustration of the input that is received by the proposed visualization.

### 3.4.1 The tool

This section discusses the software mechanisms that are implemented that allow interaction with the display. As is displayed in Figure 12 the tool receives a structured set of data containing concepts and relationships that are extracted from a set of user stories. The set of data is converted by the tool as a JSON object. The tool then employs the RAI method, as described in **Chapter 4**, to compute the ambiguity scores between concepts. To convert the data from the JSON object to a visualization the tool employs the D3js library [58]. This library is used to display the different elements, to enable the dragging of elements and to enable the zoom functionality. In addition to that, the tool employs the Venn.js library [64] to display the data as a Venn diagram. The Venn.js library contains algorithms that determine the size and position of elements. Furthermore, the filter and details-on-demand features are implemented by using the jQuery library [65]. Finally, to display the visualization properly on different screen sizes the tool uses the Bootstrap library [66]. This library contains a framework for creating responsive tools.

### 3.4.2 The device

This section discusses the device on which the visualization is displayed. The visualization is built to be used on a touch-screen video wall. As is described in **Section 1.1** effective communication between stakeholders is difficult to achieve and is a recurring problem in the elicitation of requirements. In previous work it was found that a shared display can foster collaboration between group members [49]. As such we expect that a video wall will improve communication between the requirement engineers who use the visualization. Another advantage of having such a videowall is that it allows to display a large amount of data. This is useful as set of requirements can quickly grow large containing many different concepts and relationships. Using a video wall also has the advantage that it enables the user to use touch-screen features to drag and inspect elements of interest, and even to draw on the display. We expect that such interaction further improves the communication between the users.

# Chapter 4

---

## Identification of Requirements Ambiguity via NLP

As is described in **Section 2.2.1** to keep a set of requirements consistent it is important to resolve ambiguities between requirements. **Section 3.1** introduced a conceptual framework that can be used to identify syntactically and semantic ambiguous concepts in requirements by comparing different viewpoints. This chapter explores how the application of NLP can help to find syntactically ambiguous concepts in user stories. To automatically detect such concepts this work relies on NLP tools that calculate the semantic distance between two concepts: a numerical representation of how close or distant two terms are in their meaning [67]. Current state of the art NLP tools, such as Word2Vec, employ word statistics based on in which contexts a word is used to establish their semantic relatedness on a 0-1 scale [68]. The higher the score, the more likely it is that the two concepts share the same meaning. The Semantic Folding Theory (SFT) is an alternative novel method that uses a neuroscience based mechanism of distributional semantics [69]. **Section 4.1** presents the method for detecting syntactical ambiguous concepts in user stories. This method is evaluated through a correlation study of which the results are presented in **Section 4.2**

### 4.1 The Requirements Ambiguity Identification method

This section presents the RAI method for identifying syntactically ambiguous concepts in user stories. An overview of this method can be found in Figure 13. The input of the method is a structured data file of concepts and relationships extracted from a set of user stories using the Visual Narrator tool [24]. The output of the method is a list of concept pairs with their ambiguity score. This method contains five steps which are each discussed in the following sub sections.

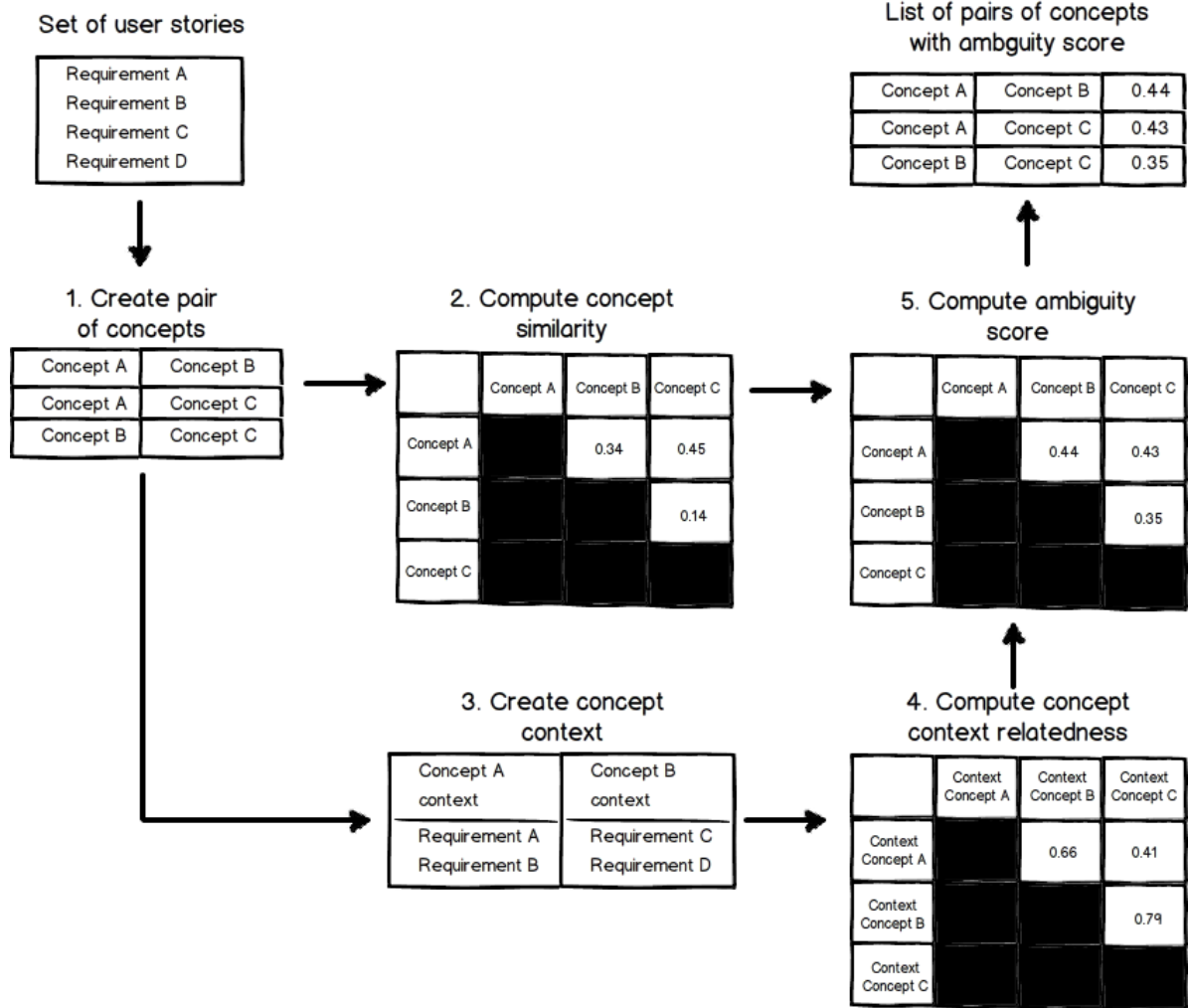


Figure 13 - Overview of the RAI method containing 5 steps.

### Step 1: Create concept pairs

In the first step of this method all concepts are extracted from the input file. In this work the concepts are extracted by using the tool of [1]. Once the concepts are extracted each concept is paired with each other concept.

### Step 2: Compute concept similarity

The similarity score is unknown for each concept pair (Figure 14). In this step the similarity score is computed between each concept pair. This score indicates how similar a concept pair is. The similarity score is a score that has a range between 0 and 1. The higher the score, the more similar a concept pair is.

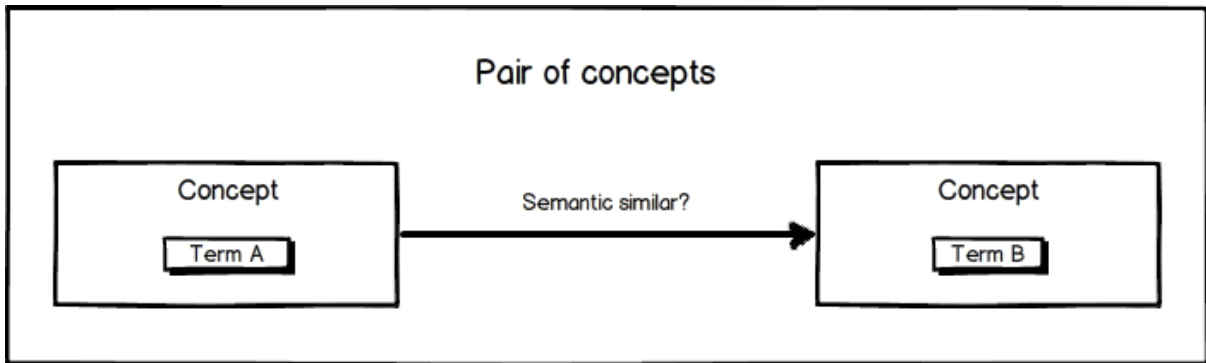


Figure 14 - Illustration of a concept pair of which the similarity score is unknown.

### Step 3: Create concept context

Requirements that share concepts that are in a correspondence state are more likely to appear in the same context and are therefore more likely to be syntactically ambiguous. For example, requirements that are related to an *album* appear in the context of an album. This example is illustrated in Figure 15. Within the requirements found in this figure we can find that an album contains *media elements* and *media*. It is likely that these two terms refer to the same concept.

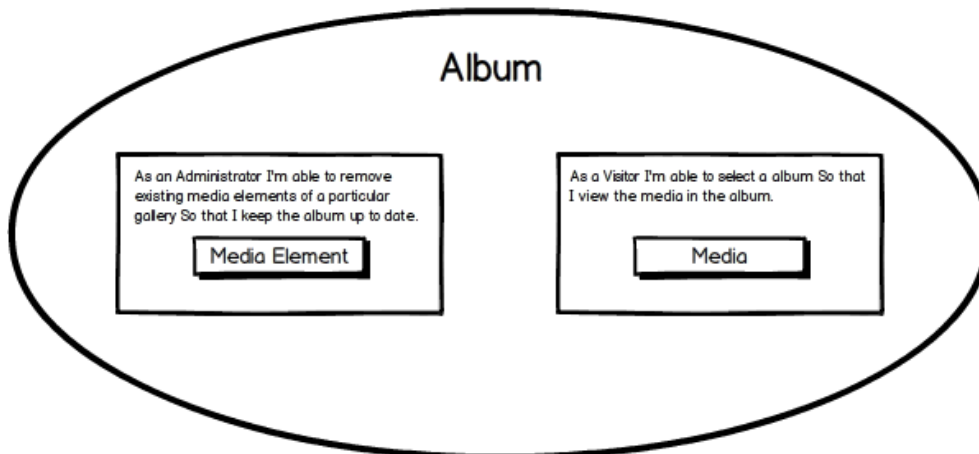


Figure 15 - Illustration of two requirements from the WebCompany data set [1] that contain concepts that are in a correspondence state as they appear in the context of 'album'.

We define the context of a concept as all requirements in which a concept appears. As third step in our method we therefore extract all the requirements in which a concept appears. It should be noted that it is unlikely for concepts to be in a correspondence state when they both appear in the same requirement. For each concept pair we therefore remove all requirements from the concept contexts in which both concepts appear. For example, the concepts *email address* and *password* share the same context as they both appear in user story 5 (U5), however they do not share the same meaning.

U5 As a User, I can login using my *email address* and *password*, so that I get access to the user -only features of the website.

#### Step 4: Compute concept context similarity

In this step the cosine relatedness score is computed between the contexts for each concept pair (Figure 16). The higher the score, the more related the context of a concept pair is.

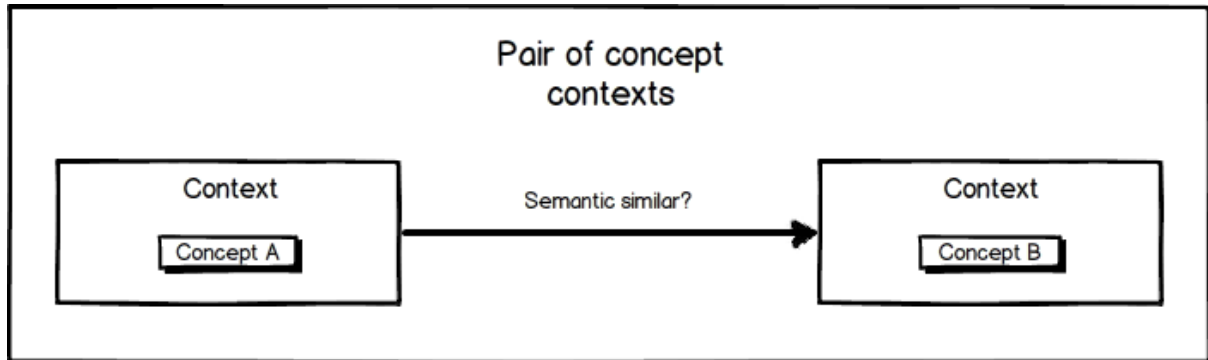


Figure 16 - Illustration of a concept pair with their context of which their semantic relatedness is unknown.

#### Step 5: Compute syntactic ambiguity score

In the final step of the RAI method the ambiguity score for each concept pair is computed. This score is computed by multiplying the concept similarity score (derived from step 2) with 2 and adding the context relatedness (derived from step 4) to the sum. This sum is then divided by three which results into the ambiguity score. The algorithm can be found below:

$$aScore = \frac{simScore(c_i, c_j) \times 2 + relnScore(ctx_i, ctx_j)}{3}$$

## 4.2 Requirements Ambiguity Identification method correlation study

This section presents the correlation study that is conducted to investigate the usefulness of the RAI method presented in the **Section 4.1**. This study helps to evaluate how suitable the method is for detecting ambiguities in user stories. **Section 4.2.1** presents the correlation study protocol. **Section 4.2.2** presents the results that are obtained from the correlation study.

## 4.2.1 Correlation study protocol

This section describes the correlation study protocol that is employed to evaluate the RAI method. The purpose of this study is to test the accuracy of this method. In the context of the thesis this study answers SRQ4. This study contains a pilot study that is followed up by the actual correlation study. The following sections describe the correlation study protocol in more detail.

### 4.2.1.1 Hypothesis

In this correlation study the hypothesis is defined as followed:

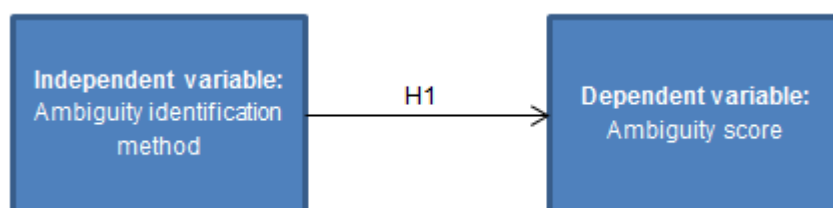
**H1: The accuracy of the RAI method for identifying syntactically ambiguous requirements has a significant positive correlation with the accuracy of Information Science students identifying syntactically ambiguous requirements.**

### 4.2.1.2 Design

The **independent variable** in this study is the method that is being used for identifying syntactically ambiguous requirements. There are two different methods: (1) human knowledge on the topic of ambiguity and (2) the RAI method presented in **Section 4.1**. The **dependent variable** in this study is the ambiguity score given to the concept pairs. Table 21 gives a description of the independent and dependent variables in this study. Figure 17 presents an overview of these variables.

<b>Independent variable</b>	Human knowledge on ambiguity	RAI method
<b>Dependent variable</b>	Ambiguity score given to the concept pairs <i>(based on the total score / number of students)</i>	Ambiguity score given to the concept pairs <i>(based on the requirements ambiguity identification method)</i>

**Table 21** - Description of the correlation study independent and dependent variable.



**Figure 17** - Overview of the correlation study independent and dependent variable.

#### **4.2.1.3 Data collection**

The sample group in this correlation study contains 8 Information Science students. This group is selected based on the convenience sampling technique [70]. This sample group uses *human knowledge on ambiguity* to determine the ambiguity score for each concept pair. Once the data for each participant is collected, the average score for each concept pair is calculated (total score / number of participants). This results in a list of concept pairs with scores based on human knowledge on ambiguity. For this correlation study, data is also collected for each concept pair by using the RAI method presented in **Section 4.1**. This results in a list of concept pairs with their ambiguity score based on the RAI method.

In addition to quantitative research we also perform qualitative research by presenting each participant a small survey (**Appendix E**). A survey can be used to collect information from people that will help to explain their decision [70]. This is well suited for this use case as it allows to collect information from the participants that will help explain why a participant considers a requirement to be ambiguous. The survey questions in this study are based on the guidelines for designing survey questions as presented by [71].

#### **4.2.1.4 Procedure**

The following steps are defined to conduct the correlation study:

1. In the first step the RAI method is executed on the Web Company data set [1]. This creates a list of concept pairs with the requirements in which they appear and their ambiguity score.
2. We randomly select 8 concept pairs with an ambiguity score above 0.6, randomly select 8 concept pairs with a score between 0.4 and 0.6 and randomly select 8 concept pairs with a score below 0.4. This results in a total number of 24 concept pairs.
3. The participant receives a description which includes a short introduction and explanation on how to find ambiguous requirements (**Appendix B**). The participants then receives a list of 12 concept pairs with the requirements in which they appear (**Appendix C**). The participant needs to identify which concept pairs are ambiguous by using his knowledge on ambiguity. This process is repeated for each participant in the sample group.
4. Once the participant has assigned an ambiguity score to each concept pair the participant is given a small survey. In this survey the participants is asked to pick 2 concept pairs of which he thinks that they are ambiguous and is asked why he



thinks so. The same is done for two concept pairs of which he thinks are not ambiguous. This survey is audio recorded.

5. Once the data is collected a Pearson correlation test is conducted on the ambiguity scores given by the ambiguity method and on the scores given by the participants to find whether there is a significant positive correlation.

#### ***4.2.1.5 Analysis***

Once the data is collected a Pearson correlation test is conducted on the dependent variable to test the hypothesis. In this work the desired outcome is that the correlation study shows that the RAI method shares a significant positive correlation with the human knowledge on ambiguity method. The qualitative data obtained from the survey is used to gain a better understanding behind the reasoning of a person in regards to ambiguity. This may reveal potential information that can be useful to improve the RAI method.

#### **4.2.2 Correlation study results**

This section presents the results that are obtained from the correlation study. **Section 4.2.2.1** presents the data preparation. The data analysis is given in **Section 4.2.2.2**

##### ***4.2.2.1 Data preparation***

Table 22 contains a list which summarizes the data set that is obtained through a questionnaire with 8 participants with a background in Information Science. This list contains a total number of 24 concept pairs with their computed similarity scores. The list is ordered from high to low by the score computed by the algorithm. This makes it convenient to analyse how the algorithm performed compared to the score given by the participants.

A concept pair can be considered to be certain ambiguous when the score in the 'average human score' column is 3.5 or higher. When the score is between 2.5 and 3.5 a concept pair can be considered to be likely ambiguous. A score between 1.5 and 2.5 indicates a concept pair is unlikely to be ambiguous. And finally, when the score is below 1.5 a concept pair is considered to be impossible to be ambiguous.

Concept A	Concept B	S_1	S_2	S_3	S_4	Average human score	Concept similarity score	Context relatedness score	Algorithm score
Media Gallery	Gallery	4	4	1	4	3.25	0.361	0.648	0.428
Profile Page	Profile	2	4	3	4	3.25	0.367	0.622	0.427
Password	Password Reset	4	1	2	2	2.25	0.38	0.535	0.416
Location	Event Location	4	3	4	4	3.75	0.357	0.528	0.397
Event Plot	Plot	3	2	4	4	3.25	0.355	0.49	0.387
Plot	Plot Location	3	2	3	2	2.5	0.359	0.477	0.386
Media	Media Gallery	1	3	3	1	2	0.361	0.405	0.371
Media Category	Category	4	3	4	4	3.75	0.36	0.359	0.36
Description	Text	4	3	3	2	3	0.177	0.889	0.343
Page	Content	3	2	1	2	2	0.262	0.547	0.329
Information	Link	1	3	3	3	2.5	0.25	0.555	0.321
Description	Image	3	1	2	1	1.75	0.128	0.864	0.3
Event Plot	Plot Location	4	1	4	3	3	0.176	0.633	0.283
Content	Profile	2	2	1	1	1.5	0.183	0.612	0.283
Information	Profile	2	2	4	3	2.75	0.177	0.612	0.278
Media Gallery	Media Category	3	3	1	2	2.25	0.175	0.413	0.231
Detail	Plot	2	1	1	1	1.25	0.128	0.441	0.201
Video	Profile Card	2	1	1	1	1.25	0.074	0.538	0.183
Text	Plot	2	3	2	1	2	0.104	0.381	0.168
Password Reset	Video	1	1	1	1	1	0.053	0.368	0.127
Password	Story	1	1	1	1	1	0.067	0.239	0.107
Contact Form	Audio Fragment	1	1	1	1	1	0.081	0.133	0.093
Category	Text	1	1	1	1	1	0.037	0.248	0.086
News Section	Description	1	2	2	1	1.5	0.022	0.272	0.08

**Table 22** - Overview of the data set obtained through the questionnaire. The variables in the columns S\_1, S\_2, S\_3 and S\_4 represent the ambiguity scores given to the requirements of a concept pair by each participant. The variables in the average human score column are calculated by calculating the sum of S\_1, S\_2, S\_3 and S\_4 and dividing it by 4 for each concept pair. The variables in the column algorithm score are calculated by applying the algorithm presented in **Section 5.1.5**. The input for this algorithm is the variables in the columns concept similarity score and context relatedness score.

#### 4.2.2.2 Data analysis

This section presents the analysis by using SPSS on the data presented in the previous section. A Pearson correlation test is performed on the variables 'average human score' and 'concept similarity score', and on the variables 'average human score' and 'algorithm score'.

Figure 18 presents the computation of the Pearson correlation between the variables 'average human score' and 'concept similarity score'. From this result we find there is a significant positive correlation between the variables,  $r = .773, p = \leq .001$ .

		Average_Human_score	Similarity_score
Average_Human_score	Pearson Correlation	1	.773**
	Sig. (2-tailed)		.000
	N	24	24
Similarity_score	Pearson Correlation	.773**	1
	Sig. (2-tailed)	.000	
	N	24	24

\*\* . Correlation is significant at the 0.01 level (2-tailed).

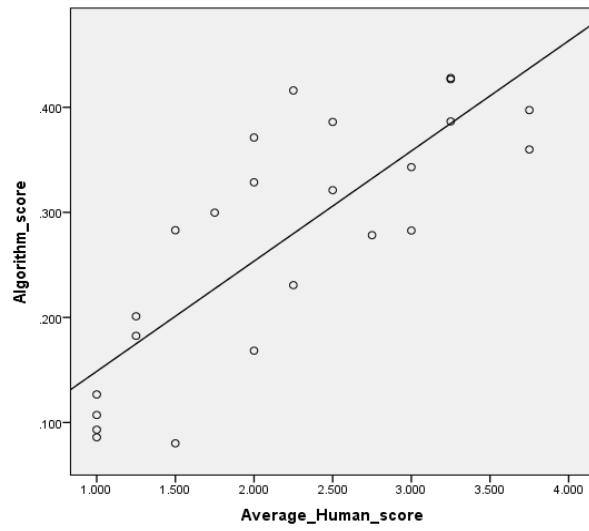
**Figure 18** - SPSS output of the Pearson correlation test between the 'average human score' and 'concept similarity score'.

Figure 19 presents the computation of the Pearson correlation between the variables 'average human score' and the 'algorithm score'. From this result we find there is a significant positive correlation between the variables,  $r = .806, p = \leq .001$ . This result shows that the inclusion of the context relatedness in the algorithm leads to better results in detecting syntactically ambiguities in user stories. Figure 17 shows an illustration of the correlation between the two variables.

		Average_Human_score	Algorithm_score
Average_Human_score	Pearson Correlation	1	.806**
	Sig. (2-tailed)		.000
	N	24	24
Algorithm_score	Pearson Correlation	.806**	1
	Sig. (2-tailed)	.000	
	N	24	24

\*\* . Correlation is significant at the 0.01 level (2-tailed).

**Figure 19** - SPSS output of the Pearson correlation test between the 'average human score' and 'algorithm score'.



**Figure 20** - An illustration of the correlation between the 'average human score' and the algorithm score in which the similarity score is given a weight of 2 and the context relatedness score is given a weight of 1.

# Chapter 5

---

## Tool Evaluation

To determine the usefulness of the proposed tool the conceptual framework from section 3.1, the visualization design aspects from **Section 3.2** and **Section 3.3**, and the RAI method from **Chapter 4** have been implemented as a software tool [72]. The usefulness of this tool is evaluated through an evaluation scenario. **Section 5.1** describes the evaluation protocol that is employed in the evaluation of the implemented tool. The results of the evaluation are presented in **Section 5.2**.

### 5.1 Tool Evaluation Protocol

#### 5.1.1 Goal

The type of scenario should fit with the goals of the evaluation. The goals of this evaluation are defined as followed:

1. Determine how useful the implemented tool is to find ambiguities in user stories compared to how useful pen and paper is to find ambiguities in user stories.
2. Determine how useful the implemented tool is to find missing user stories compared to how useful pen and paper is to find missing user stories.

#### 5.1.2 Scenario

The scenario that is employed in this evaluation is the VDAR scenario [28]. This scenario is focused on evaluating how effective a tool is to support users in generating actionable and relevant knowledge in their domain. This scenario type is well suited for this use case since it fits well with the defined goals.

#### 5.1.3 Evaluation question

The evaluation question of this scenario is defined as followed:

*“How useful is the implemented tool for a requirement engineer for the application of knowledge discovery compared to using pen and paper?”*

To answer the evaluation question the following hypotheses are defined:

**H1: Requirement engineers who use the implemented tool produce a significant higher precision ratio for finding ambiguities in a set of user stories than requirement engineers who use pen & paper.**

**H2: Requirement engineers who use the implemented tool produce a significant higher recall ratio for finding ambiguities in a set of user stories than requirement engineers who use pen & paper.**

**H3: Requirement engineers who use the implemented tool produce a significant higher precision ratio for finding missing user stories than requirement engineers who use pen & paper.**

**H4: Requirement engineers who use the implemented tool produce a significant higher recall ratio for finding missing user stories than requirement engineers who use pen & paper.**

#### **5.1.4 Sample group**

To answer the evaluation question of this scenario an evaluation is conducted which consists of two sample groups. Sample group A uses the implemented tool and is from this point on referred to as the tool group. Sample group B uses Pen & Paper and is from this point on referred to as the control group. The participants that participate in this evaluation have a background in Information Science and are originally part of teams of 3 students from a Software Architecture course. In this course each of these teams has created a set of user stories. The context of these user stories is the same as they are all about an event ticket system.

The user story sets that originate from this Software Architecture course only contain a limited number of user stories. The implemented tool is meant to be used on large sets of user stories. For this reason we use the user story sets of team 6 and 13, and combine these user stories into one user story set. Duplicate user stories are removed from the combined data sets.

When creating the teams for the evaluation it is made sure that the teams do not contain team members who were in the same group during the Software Architecture course. Each participant is assigned to a Participants Poole. Participants in 'Participants Poole A' use Pen & Paper while participants in 'Participants Poole B' use the tool.

### 5.1.5 Data collection

In this evaluation each team is asked to find ambiguities and missing user stories by using the assigned method.

A **missing user story** is defined as followed:

*A user story is missing when its absence inhibits the realization of another user story.*

An **ambiguity** is defined as followed:

*An ambiguity occurs when at least two user stories contain a concept pair that is described in different terminology but shares the same meaning.*

In this evaluation the participants need to perform the following steps:

1. **Introduction** (20 minutes):
  - 1.1 In the room with the video wall team A and team B are asked to read the Evaluation Description (**Appendix F**: Evaluation Description).
  - 1.2 Team A and Team B receive a short presentation about:
    1. What their goal is (finding ambiguities & missing requirements);
    2. What ambiguity is;
    3. How to find ambiguities in requirements;
    4. How to find missing requirements.
  - 1.2 Team B is asked to wait 5 minutes in a different room;
  - 1.3 Team A receives a demo about the implemented tool.
2. **Finding ambiguities & missing requirements** (20 mins):
  - 2.1 Team A & team B are asked by using the assigned method to simultaneously:  
(1) find ambiguities in user stories and to note them down on the list of ambiguities (**Appendix G**) and (2) to find missing user stories and to note their results down on the list of missing user stories (**Appendix I**).
3. **Evaluating the results** (20 mins):
  - 3.1 Team B is asked to come back to the room with the video wall;
  - 3.2 Team A and team B are asked to discuss the found ambiguities to determine whether they agree on the identified ambiguities.
  - 3.3 Team A and team B are asked to discuss the missing user stories to determine whether they agree on the identified missing user stories.

### 5.1.6 Data preparation

With the generated output from the evaluation the data can be prepared for the analysis. The data collection has generated the following data:

- For each team the **number of TP ambiguities** and the **number of FP ambiguities** found by using the **implemented tool**;
- For each team the **number of TP ambiguities** and **number of FP ambiguities** found by using **pen & paper**;
- For each team the **number of TP missing user stories** and **number of FP missing user stories** found by using the **implemented tool**;
- For each team the **number of TP missing user stories** and **number of FP missing user stories** found by using **pen & paper**.

The precision and recall ratio for finding ambiguities is calculated for each team as followed (Note that the # sign represents ‘the number of’):

$$\mathbf{aPrecisionRatio} = \#TP \text{ ambiguities} / (\#TP \text{ ambiguities} + \#FP \text{ ambiguities})$$

$$\mathbf{aRecallRatio} = \#TP \text{ ambiguities} / \text{total } \#TP \text{ ambiguities}$$

The precision and recall ratio for finding missing user stories is calculated for each team as followed:

$$\mathbf{mPrecisionRatio} = \#TP \text{ missing user stories} / (\#TP \text{ missing user stories} + \#FP \text{ missing user stories})$$

$$\mathbf{mRecallRatio} = \#TP \text{ missing user stories} / \text{total } \#TP \text{ missing user stories}$$

### 5.1.7 Data analysis

Now that the data is prepared it is possible to conduct the analysis. Table 23 contains an overview of the independent and dependent variables that are used in the analysis.

Independent variable	Implemented tool (sample group A)	Pen & paper (sample group B)
Dependent variable	aPrecisionRatio	aPrecisionRatio
	aRecallRatio	aRecallRatio
	mPrecisionRatio	mPrecisionRatio
	mRecallRatio	mRecallRatio

**Table 23** - An overview of the independent and dependent variables used in the analysis.



To determine whether hypothesis 1 is significant an independent samples t-test is conducted on the following variables:

- the **aPrecisionRatio** of **sample group A**;
- the **aPrecisionRatio** of **sample group B**.

To determine whether hypothesis 2 is significant an independent samples t-test is conducted on the following variables:

- the **aRecallRatio** of **sample group A**;
- the **aRecallRatio** of **sample group B**.

To determine whether hypothesis 3 is significant an independent samples t-test is conducted on the following variables

- the **mPrecisionRatio** of **sample group A**;
- the **mPrecisionRatio** of **sample group B**.

To determine whether hypothesis 4 is significant an independent samples t-test is conducted on the following variables:

- the **mRecallRatio** of **sample group A**;
- the **mRecallRatio** of **sample group B**.

## 5.2 Tool Evaluation Results

The evaluation of the implemented tool is conducted as described in the protocol given in **Section 5.1**. The evaluation was first conducted in a pilot study which is described in **Section 5.2.1**. **Section 5.2.2** describes the preparation of the data that is obtained from the evaluation. Finally, in **Section 5.2.3** the analysis is given of the prepared data.

### 5.2.1 Pilot study

The evaluation was first conducted during a pilot study in which four people with a background Information Science participated. After evaluating the results from the pilot study we slightly changed the evaluation set up. For example, initially we made the concepts bold in the user story list (**Appendix J**) that is used by the control group. However, since this is not how a list of user stories would generally appear we decided to make the font-weight of the concepts normal.

Another change we made is in the list of ambiguities (**Appendix G**). We found that for the control group it would be useful to let them document in which user stories they found an ambiguity. This addition optimizes step 3 of the evaluation, in which the groups discuss whether their finding are true or false positives, as it allows the control group to

recall to which user stories a found ambiguity applies. The list of ambiguities for the control group can be found in **Appendix H**.

During the pilot study one group consisted of Dutch native speakers while the other group consisted out of people who didn't speak the same native language. Since this could influence the results we decided to mix the groups with non-native speakers whenever possible.

## 5.2.2 Data preparation

The data obtained from the evaluation is prepared according to the data preparation description given in **Section 5.1.6**. Table 24 presents an overview of the obtained data in regards to ambiguity identification. Table 25 presents an overview of the obtained data in regards to missing user story identification. In both tables the variable '*# Total TP*' refers to the total number of *unique* True Positives (TP) found by both the control group and tool group. The variable '*#TP*' refers to the number of True Positives found by each team. The variable '*#FP*' refers to the number of False Positives (FP) found by each team. The variable '*#TP + #FP*' refers to the sum of the variables '*#TP*' and '*#FP*' for each team. The precision ratio for each team is computed by dividing the '*#TP*' variable with the '*#TP + #FP*' variable. The recall ratio is calculated by dividing the '*#TP*' variable with the '*#Total TP*' variable.

	#Total TP	#TP	#FP	#TP + #FP	Precision ratio	Recall ratio
<b>Session 1</b>						
Control group	28	8	1	9	0.888	0.285
Tool group		23	4	27	0.851	0.821
<b>Session 2</b>						
Control group	12	3	4	7	0.428	0.25
Tool group		9	0	9	1	0.75

**Table 24** - Overview of the ambiguities found data from the control and tool groups from session 1 & 2.

	#Total TP	#TP	#FP	#TP + #FP	Precision ratio	Recall ratio
<b>Session 1</b>						
Control group	9	4	1	5	0.8	0.444
Tool group		5	2	7	0.714	0.555
<b>Session 2</b>						
Control group	5	2	2	4	0.5	0.4
Tool group		3	2	5	0.6	0.6

**Table 25** - Overview of the missing user stories found data from the control and tool groups from session 1 & 2.

### 5.2.3 Data analysis

This section presents the data analysis of the evaluation. The evaluation contains both quantitative and qualitative results. Each of them is discussed in the following sub sections.

#### 5.2.3.1 Quantitative result

Below is the result of the independent t-tests that are conducted on the hypotheses described in **Section 5.1.3**. It should be noted that due to the small sample size more evaluations should be conducted with a larger sample size to ensure the validity of the quantitative analysis.

For H1, there is **not a significant** difference in the scores for the precision ratio using the implemented tool for finding ambiguities ( $M=.925$ ,  $SD=.105$ ) and using Pen & Paper for finding ambiguities ( $M= .658$ ,  $SD= .325$ ) conditions;  $t(-1.106)= -1.208$ ,  $p = .442$ . Further, Cohen's effect size value ( $d = 1.1$ ) suggested low practical significance.

For H2, there is a **significant** difference in the scores for the recall ratio using the implemented tool for finding ambiguities ( $M=.785$ ,  $SD=.05$ ) and using Pen & Paper for finding ambiguities ( $M= .267$ ,  $SD= .024$ ) conditions;  $t(-13.088)= 1.459$ ,  $p = .017$ . Further, Cohen's effect size value ( $d = 13.2$ ) suggested high practical significance.

For H3, there is **not a significant** difference in the scores for the precision ratio using the implemented tool for finding missing user stories ( $M=.657$ ,  $SD=.08$ ) and using Pen & Paper for finding missing user stories ( $M= .65$ ,  $SD= .212$ ) conditions;  $t(-.044)= 1.283$ ,  $p = .971$ . Further, Cohen's effect size value ( $d = 0.04$ ) suggested low practical significance.

For H4, there is a **significant** difference in the scores for the recall ratio using the implemented tool for finding missing user stories ( $M=.577$ ,  $SD=.031$ ) and using Pen & Paper for finding missing user stories ( $M= .422$ ,  $SD= .031$ ) conditions;  $t(-4.941)= 1.999$ ,  $p = .039$ . Further, Cohen's effect size value ( $d = 4.999$ ) suggested high practical significance.

#### 5.2.3.2 Qualitative results

This section presents the qualitative results obtained from the evaluation. These results were obtained by observing the participants and by interviewing the participants from the tool groups after the evaluation. In this short interview the participants were asked about their experience and whether they were missing features.

In regards to the user experience the participants from the tool groups indicated that they were missing a number of features in the tool. An overview of these missing

features can be found in Table 26. **Section 2.3.4** described the different operational questions, taken from [57], that can be used to assess a visualization. In the table it is indicated for each feature to what operational question it relates to. This helps to gain insight in what the areas of the visualization of the tool are that are open for improvement.

Feature name	Description	Operational question
'Close all' button	The participants found that when they often inspect elements of interests the display gets crowded with pop-ups. To close each of these pop-ups is time-consuming; therefore a 'close all button' could be useful. The participants indicated that the position of this button would be suitable next to the 'reset color button'.	M3 (Model extension and customization)
Search field for concepts	The participants also indicated that a search field in which you could search for a concept would be useful. The reason behind this is that the participants want to be able to instantly find back concepts that they discussed before but of which they forgot the position of. This is especially the case after the participants use the viewpoint filter to add or remove viewpoints as this changes the position of the concepts.	V4 (searching)
Underline association relationships	When inspecting the user stories of a concepts the participants found that it is time-consuming to recognize where in the user stories the associated relationships are located. Underlining these association relationships could be useful	V7 (Annotation)
More help icons	The participants indicated additional help icons could be added especially to help explain the approach one can take to use the tool for finding ambiguities and missing user stories. Such icons may make a demo of the tool obsolete.	U2 (Usage without heavy training)
Option to change/remove elements	The participants indicated it could be useful to be able to remove elements from the display if they are not interesting or already analysed. In addition to that the participants indicated that changing elements, for example changing the name of a concept when it is found ambiguous with another concept can also be useful (e.g. changing the concept 'picture' to 'photo'). This all helps to keep the information on the display up to date and to allow the requirement engineer fully focus on elements that have not yet been investigated.	M3 (Model extension and customization)

**Table 26** - Overview of features missing in the tool.

### Differences between the tool groups

During each session the tool groups were observed and the differences in how the participants of each group interacted with each other and the tool were documented.

Table 27 displays an overview of these differences. These differences are now discussed in more details.

Difference 1 (D1) indicates that the groups employed a different approach for using the tool on the video wall. For example, the participants of the tool group from the first session both stood near the video wall. The participants from the tool group from the second session took a different approach as one participant stood near the video wall while the other participant observed the video wall from a distance. The latter approach appears more useful as it gives the requirement engineers a different point of perspective on the displayed data. Figure 21 illustrates the 'near the wall' approach that two participants employed during the pilot study.

Difference 2 (D2) indicates that two requirement engineers who use the video wall simultaneously may lead to conflicting actions. For example, in session one each participant of the tool group wanted to use the video wall to drag two elements simultaneously. Instead of interpreting these actions as dragging actions the tool used the zooming functionality as it interpreted the dragging actions from both participants as a single zooming action.

Difference 3 (D3) indicates that the participants who employed the 'near the wall' approach use less time to inspect elements. For example, the participants from this group indicated that, when looking for ambiguities, they mostly focus on the concepts, and did not focus much on inspecting their associated user stories. On the other hand, the group that employed the 'Near the wall + observer' approach indicated that they did inspect the user stories frequently when searching for ambiguities. This is reflected in the quantitative results where the tool group from the first session obtained a precision ratio of .851 while the group from the second session obtained a precision ratio of 1. This result indicates that inspecting the requirements of a concept pair helps to determine whether they are truly ambiguous. However, inspecting requirements is more time-consuming as is clearly reflected in the recall ratio. The tool group from the first session obtained a recall ratio of .821 while the tool group from the second session obtained a recall ratio of .75. Further research is necessary to validate whether these findings apply specifically for the participants in these tool groups or whether they can be generalised.

ID	Near the wall	Near the wall + observer
D1	- Same point of perspective. When both requirement engineers stand near the video wall it only allows them to focus on a small portion of the displayed data.	- Different point of perspective: <ul style="list-style-type: none"> <li>o the requirement engineer who is standing near the video wall is able to inspect the data from close by and can use the tool's features to focus on elements of interest;</li> <li>o the requirement engineer that observes the data from a distance has a complete overview of the displayed data.</li> </ul>
D2	- Using the video wall simultaneously leads to conflicting actions	- Only the requirement engineer near the video wall is able to use the tool's functionalities. This prevents conflicting actions. On the other hand, to inspect and focus on elements of interest the requirement engineers solely rely on the requirement engineer that is standing near the video wall.
D3	- Inspected elements infrequently	- Inspected elements frequently

**Table 27** - Overview of the differences in the approach employed by the tool groups. A green color indicates the participant experienced the characteristic of their approach as positive. A red color indicates the approach was experienced as negative. An orange color indicates the participant experienced the approach as something between positive and negative.



**Figure 21** - Two participants using the implemented tool on a video wall using the 'near the wall' approach during the pilot study.

### Differences between the tool groups and control groups

During the evaluation the groups were observed and the differences in how the participants of each group used their assigned method for finding ambiguities and missing user stories were documented. Table 28 displays an overview of these differences. These differences are now discussed in more details.

Difference 4 (D4) indicates that both methods do not require much training. To make the demo of the tool obsolete one of the tool groups indicated that more help icons that describe how the tool can be used for finding ambiguities and missing user stories could help.

Difference 5 (F5) indicates that a requirement engineer is forced to go through all the details. When using the tool the requirement engineer can choose between the most appropriate option by either only viewing a short description or by inspecting the details. This may explain why H2 and H4 are significant. A user story list is simply not a feasible method for finding a large number of ambiguities and missing user stories in a large data set in a short time as it is too time-consuming to go through all the details. On the other hand, an advantage of being forced to go through all the details is that the requirement engineer is forced to take all the information into account while this is not the case when a requirement engineer uses the tool. This may explain why H1 and H3 are not significant as it seems that to determine whether something is a true ambiguity or missing user story it is important to read the associated details.

Difference 6 (F6) indicates that in a user story list the information is not organized in a logical way and that it is difficult to find information back. This may explain why most of the ambiguities that the control groups found are in user stories that are near to each other. Scanning through information that is not organized in a logical way and not being able to easily find information back is time-consuming and thus further explains why H2 is significant.

ID	Control group	Tool group
D4	<ul style="list-style-type: none"> <li>- For the control groups it was immediately clear without any training how they had to use the user story list.</li> </ul>	<ul style="list-style-type: none"> <li>- The tool group understood how to use the tool without much training. The 5 minute demo of the tool that the tool groups were given was sufficient to get comfortable with its use. While the device (a video wall) on which the tool was used is uncommon in daily practice it was not an obstacle for the participants as they instantly understood how to use it.</li> </ul>
D5	<ul style="list-style-type: none"> <li>- When using a user story list the requirement engineer is forced to read the details as reading the entire user story is mandatory. Although this is an advantage when the undertaken action requires all the details it is a major disadvantage when the undertaken action only requires a short description as it is unnecessarily time-consuming to go through all the details when it is not required.</li> </ul>	<ul style="list-style-type: none"> <li>- When using the tool the requirement engineer has the option to choose between a short description and detailed information. Having these options is an advantage as it allows a requirement engineering to pick the most appropriate option and in doing so save time. However when the wrong option is chosen it can influence the results.</li> </ul>

D6	<p>- When using a user story list the requirements are clustered together depending on the role that they belong to. Other than that the requirements are not organized in a logical way and there are no means to find information back other than scanning through the entire document.</p>	<p>- The requirements in the tool are organized in a logical way and information can be easily found back by focusing on elements of interest.</p>
----	---	--

**Table 28** - Overview of the differences between the control groups and the tool groups. A green color indicates the participant experienced the characteristic of their method as positive. A red color indicates the participant experienced the characteristic of their method as negative. An orange color indicates the participant experienced the characteristic of their method as something between positive and negative.



# Chapter 6

---

## Conclusion

The objective of the conducted research was to determine how the application of IV and NLP could help to create a better understanding of software requirements. This objective was triggered by the observation that current requirements elicitation techniques are sub-optimal as far as representing requirements inconsistencies and stakeholder disagreements. This chapter summarizes the research findings that provide answers to all of the sub questions of this research. Based on these different findings a final conclusion is drawn. This conclusion provides the answer to the main question in this research, which is based on the research objective described above.

### 6.1 Conclusions of the Sub Research Questions

The answer of the main research question is divided over the six sub research questions. This section provides the answers to these six sub research questions.

#### **SRQ 1: What problems do practitioners encounter in understanding software requirements?**

**Section 2.2** presented a literature study which investigated the problems practitioners encounter in understanding software requirements. From the literature it was found that effective communication between stakeholders is difficult to achieve. As a result a set of requirements becomes inconsistent and incomplete as the set of requirements does not reflect the true needs of the different stakeholders. Defining a consistent and complete set of requirements is a crucial step in software development. Checking requirements for consistency and completeness is hard because it is time-consuming for requirement engineers to analyse requirements. The reason behind this can be found in the following problems: (1) a set of requirements keeps changing due to changing needs of (newly introduced) stakeholders, (2) due to the complexity of the domain it can be difficult to understand the expressed software requirements and (3) as a set of requirements grows it quickly becomes infeasible to analyse it.

## **SRQ 2: What types of RE visualizations already exist and what can be learned from them?**

**Section 2.3.4** investigated three RE visualizations that employ interactive visualization techniques and support RE verification & validation tasks. It was found that these visualizations do not employ sophisticated IV techniques. The reason behind this may be that it is labour-intensive to develop tools with highly sophisticated visualization functionalities. When creating a tool with sophisticated IV techniques it is therefore useful to use a visualization library (e.g. [58]-[59]). The investigated RE visualizations also show little support for automatic model creation. To support automatic model creation it is important that the tool that incorporates the visualization is able to automatically create the model from a source of data. Furthermore, none of the investigated RE visualizations can be manipulated, for example by allowing the user to change element's colors or shapes. Such manipulations help to adapt the visualization to the personal needs of the user. Finally, for each of the investigated visualizations there is also room for improvement in presenting the user with more and in-depth knowledge. For example, the investigated visualizations do not or hardly support the option to inspect detailed information on elements and hardly present the user with actionable decisions. The main concern with the investigated RE visualizations is that each of them present their information mostly static and do not allow to convert the displayed information into knowledge.

## **SRQ 3: How can missing requirements be identified?**

**Section 3.1** presented a conceptual framework that is constructed to help to find requirements incompleteness. This framework helps to find missing requirements by comparing viewpoints with each other which allows to identify the state of a concept. The contrast state indicates a concept exists in one viewpoint but is missing in another. Knowing that a concept is in a contrast state allows the requirement engineer to wonder whether such a concept should also appear in other viewpoints. If a concept is missing in a viewpoint it means that there are associated requirements missing in which this concept should appear.

## **SRQ 4: How can ambiguities in requirements be identified?**

The constructed framework from **Section 3.1** also helps to find ambiguities in requirements. This framework helps to find ambiguities in requirements by comparing

viewpoints with each other which allows to identify the state of a concept. The correspondence state helps to identify syntactic ambiguities in requirements. A requirements is syntactic ambiguous when it contains concepts that are in a correspondence state. A concept is in a correspondence state when it exists in two or more viewpoints but is described in different terminology.

The conflict state helps to identify semantic ambiguous requirements. A requirement is semantic ambiguous when it contains concepts that are in a conflict state. A concept is in a conflict state when it exists in two or more viewpoints but these viewpoints assign a different meaning to the associated concept.

In addition to the conceptual framework this work has presented the RAI method in **Section 4.1**. This method contains an algorithm which computes the ambiguity score between concept pairs. Concepts that have a high ambiguity score are likely to be syntactic ambiguous while concept pairs that share a low ambiguity score are unlikely to be syntactic ambiguous. The accuracy of this algorithm has been evaluated in a correlation study. The correlation study shows a significant positive correlation between the scores computed by the algorithm and given by humans,  $r = .806, p = \leq .001$ .

#### **SRQ 5: What criteria should the proposed visualization satisfy to identify ambiguities and missing requirements?**

To help identify ambiguities and missing requirements a list of criteria which the proposed visualization should satisfy has been defined in **Appendix K**. The framework for this list originates from the work of [60]. Figure 4 contains a starplot based on the defined criteria that indicates to what degree the different key areas are supported by the proposed visualization. Table 16 contains a description of the dimensions that are discussed in the list of criteria. The main criteria of this list are described below.

Regarding the user dimension, the visualization should foster collaboration between requirement engineers. For this purpose the visualization is displayed on a video wall. A video wall contains a large shared display which allows requirement engineers to interact with the displayed elements through touch-screen interactions and fosters collaboration between them [49]. Furthermore, a video wall allows to display a high number of elements compared to a regular computer display which is useful as a set of requirements can contain many concepts and relationships. In addition to that, the visualization should be incorporated into an existing software development environment as this allows requirement engineers to incorporate the visualization in their daily work without much effort. For this purpose the tool is meant to be used in agile software development. To be more precise, the input of the tool is a set of user stories. Furthermore,

the visualization is based on existing theories for finding ambiguities and missing requirements as it supports in-viewpoint checking as well as inter-viewpoint checking.

Regarding the data dimension, it is important that the tool is able to process large-scale input. A set of user stories can contain many requirements and thus many concepts and relationships between them. The tool should be able to process all this data without much effort.

Regarding the model dimension, the visualization should allow in-viewpoint checking by allowing the requirement engineer to inspect different viewpoints in the model separately. Furthermore, the visualization should also support inter-viewpoint checking by allowing the requirement engineer to compare viewpoints with each other. In addition to that, the model should be generated automatically by the tool as it is too time-consuming for a requirement engineer to create a model manually from a (large) set of user stories.

Regarding the visualization dimension, the visualization should support filtering features to allow the requirement engineer to focus on elements of interest. Furthermore the visualization should use labels to indicate what the displayed elements are about.

Regarding the knowledge dimension, it is important that the displayed information can be truly turned into knowledge for the requirement engineer. As such, the tool should support detailed explanations to help allow the requirement engineer to inspect the details of elements of interest. Furthermore, the requirement engineer should be able to gain new insights by inspecting the colors of concept nodes to find ambiguities within requirements and by inspecting the areas in which concept nodes appear to determine the concept state of a concept which ultimately helps to identify ambiguities and missing requirements.

#### **SRQ 6: Does the implemented tool satisfy the expectations of a requirement engineer to help identify ambiguities and missing requirements?**

To determine whether the implemented tool is effective for identifying ambiguities and missing requirements an evaluation has been conducted as is described in **Chapter 5**. Although it was not possible to conduct the evaluation with real requirement engineers due to time issues; the evaluation was conducted with students who have experienced what is like to act as a requirement engineer during a Software Architecture course that they followed.

The evaluation showed there is a significant difference in the scores for the recall ratio using the implemented tool for finding ambiguities ( $M=.785$ ,  $SD=.05$ ) and using Pen & Paper for finding ambiguities ( $M= .267$ ,  $SD= .024$ ) conditions;  $t(-13.088)= 1.459$ ,  $p =$

.017. Further, Cohen's effect size value ( $d = 13.2$ ) suggested high practical significance. In addition to that the evaluation also showed there is a significant difference in the scores for the recall ratio using the implemented tool for finding missing user stories ( $M=.577$ ,  $SD=.031$ ) and using Pen & Paper for finding missing user stories ( $M= .422$ ,  $SD= .031$ ) conditions;  $t(-4.941)= 1.999$ ,  $p = .039$ . Further, Cohen's effect size value ( $d = 4.999$ ) suggested high practical significance. These findings indicate that the implemented tool is more useful for finding a larger number of ambiguities as well as missing requirements within a determined period of time compared to using pen & paper.

The evaluation did not provide evidence that the implemented tool achieves a higher precision ratio for finding ambiguities and missing requirements compared to pen & paper. It should be noted that due to the small sample size future evaluations are necessary to verify the evaluation results presented in this work.

## 6.2 Conclusion of the Main Research Question

The sub research questions provided the research findings which are needed to answer the main research question within this thesis. The main research question was defined as followed:

---

**RQ:** *How to better understand software requirements through the application of IV and NLP?*

---

This question investigates how the field of RE can benefit from the application of IV and NLP. To better understand a set of software requirements one needs to be able to analyse it. IV allows to gain new insights [6] and allows knowledge discovery through highlighting and filtering of information that is of interest [3]. To allow such tasks one needs to be able to inspect a model that represents a given set of software requirements. Such a model cannot be constructed manually since it contains too much information and complexity for a human to construct within a reasonable amount of time. For this purpose the implemented tool in this work employs the NLP method from [1] to automatically extract concepts and relationships from a set of user stories. Once the concepts and relationships are loaded the implemented tool automatically constructs a model of the associated set of requirements.

To help a requirement engineer make sense of the model several IV techniques have been incorporated in the tool. A set of requirements quickly becomes too large for a requirement engineer to grasp [16]. For this purpose the implemented tool presents an

overview of the information that is extracted from the set of requirements. The model is presented as a Venn diagram. Within this Venn diagram each element and the relationships between them is displayed in a structured way. This structure is created by positioning and sizing elements in the model in a logical way. This allows a requirement engineer to easily explore a set of requirements by following the structure of the model. In the model constructed in this work the position of elements help to recognize whether a requirement is missing or contains ambiguities.

The colors of elements also play an important role in the overview. The color of an element conveys important information as this information helps a requirement engineer to determine whether or not an element is of interest. The colors of the concept nodes are based on the ambiguity scores computed by the RAI method. Employing NLP techniques, such as the RAI method, for ambiguity identification is necessary as it is too time-consuming to solely rely on human intelligence to identify a large number of ambiguities in a (large) set of requirements. The colors of association icons help to recognize what requirements exist in the set of requirements and help to recognize missing requirements.

Furthermore, detailed information can be viewed by inspecting elements. Details on demand are an important aspect of the visualization as the model would otherwise be overcrowded with information. Finally, the implemented tool has a number of filters that can be used to focus on elements of interest. Filtering helps to create a better understanding of requirements as it allows the requirement engineer to fully use its analytical capabilities on the information that he is interested in.

# Chapter 7

---

## Discussion

The implemented tool with the incorporated IV and NLP techniques in this research has provided several valuable insights. There are however some limitations to the conducted research. These limitations are addressed in this chapter. Besides these research limitations, an overview of future research possibilities is provided.

### 7.1 Research Limitations

As indicated above there are some limitations within this research. One of the major limitations can be found in the rather small sample size of the evaluation study, presented in **Chapter 5**. Another major limitations is the fact that the participants of the evaluation study and the correlation study, presented in **Section 4.2** all have a background in Information Science. The reason behind these limitations is related to time issues. These limitations and others are now described in more detail below according to the four types of validity: conclusion validity, internal validity, construct validity and external validity [70].

#### 7.1.1 Conclusion validity

Conclusion validity concerns the statistical analysis of results and the composition of subjects. A major concern of the evaluation study regarding conclusion validity is the fact that the evaluation study resulted in a low number of samples as only two evaluations were conducted. As such it is difficult to determine how effective the implemented tool truly is for finding ambiguities and missing user stories and which factors may have influenced the results. For example, the collaboration technique that the participants had chosen could have had an impact on the results. In future research more samples are necessary to isolate such factors and by doing so help to determine how effective the implemented tool truly is.

#### 7.1.2 Internal validity

Internal validity concerns matters that may affect the independent variable with respect to causality, without the researcher's knowledge. There are a number of limitations regarding the internal validity of the conducted correlation study that was employed to

assess the RAI method. First, the selection of the participants was not random. The participants were selected based on the convenience sampling technique [70]. Second, the correlation study only used one set of user stories. The set of user stories could influence the results as each set of user stories is different in terms of complexity and number of ambiguities.

Regarding the evaluation study of the implemented tool the sample size was rather small. More evaluations with a larger sample size are necessary to confirm the evaluation results presented in this work. In addition to that the sets of user stories that were used in this evaluation originate from a Software Architecture course. As a result the results that were obtained from this evaluation may not apply to a set of user stories that originates from a software engineering environment, such as a software company.

### **7.1.3 Construct Validity**

Construct validity concerns generalisation of the experiment result to concept or theory behind the experiment. A major concern of the evaluation study that was conducted to evaluate the implemented tool is that it is not clear which aspects of the tool are effective for finding ambiguities and missing requirements. The tool with the incorporated conceptual framework, RAI method and visualization aspects has been evaluated as a whole. The usefulness of the conceptual framework that is presented in **Section 3.1** has not been evaluated with a sample group. It is therefore unknown which aspects that are incorporated in the tool make it effective for finding ambiguities and missing user stories. It could for example be that the conceptual framework is useful for finding ambiguities and missing user stories but that the added visualization aspects do not add any value for this purpose.

### **7.1.4 External validity**

External validity concerns generalisation of the experiment result to other environments than the one in which the study is conducted. There are a number of limitations regarding the external validity of the conducted correlation study that was employed to assess the RAI method. First, the sample group consisted of participants who all have a background in Information Science. Since this particular group could have a different definition of when a concept pair is ambiguous more studies with a more diverse sample group should be conducted to confirm whether the results are generalizable. To be more specific, the RAI method should be evaluated in a software development environment, such as a software company, to determine the usefulness of the method in the field of software engineering. The same applies to the evaluation that was conducted to evaluate the



implemented tool. This evaluation study was conducted with students and not at a company. In future research the implemented tool should be evaluated in a situation that reflects software development in daily practice.

## 7.2 Future Research

The performed research as described in this research developed into several future research opportunities. The implemented tool currently visualizes the viewpoints of the different roles that one can find in a set of user stories. However in future research one could investigate whether visualizing the viewpoints of user story writers is useful for identifying ambiguities and missing user stories. It is possible that different user story writers describe requirements in different terminology. It would be interesting to determine whether the conceptual framework presented in section would be suitable for this task and to adapt the implemented tool for this use case. Such a tool could act more as a project management tool as it would be used by team members to determine whether they agree on the existence of requirements and the terminology in which they are described.

Another research opportunity that could be investigated in the future is in regards to global software development. Communication between stakeholders is one of the reasons why requirement elicitation is so difficult [11]. In global software development communication between stakeholders is even more difficult [73]. As a result it may be that there is an even greater need for a tool that helps to identify and solve conflicts between stakeholders in global software development. The tool presented in this work is meant to be used by requirement engineers who are present in the same physical location and time, and is therefore not suitable to be used in global software development. In future research it could be investigated what the criteria are for a tool that helps to identify ambiguities and missing requirements in global software development.

Finally, the tool presented in this work focuses specifically on user stories. However, it may also be interesting to investigate whether requirements that are described in a different notation are also suitable for a tool such as the one presented in this work. For example, it could be interesting to investigate whether requirements described in plain text without a structure can be visualized in the same way as we visualize user stories in this work. Or perhaps a different kind of visualization for requirements in a notation without a structure would be more suitable.

# References

- [1] M. Robeer, G. Lucassen, J. M. van der Werf, F. Dalpiaz, and S. Brinkkemper, "Automated Extraction of Conceptual Models from User Stories via NLP," *Proc. 24th IEEE Int. Requir. Eng. Conf.*, 2016.
- [2] S. R. Faulk, "Software requirements: A tutorial," *Development*, pp. 1–33, 1995.
- [3] J. J. R. Cooper, S. W. Lee, R. A. Gandhi, and O. Gotel, "Requirements Engineering Visualization : A Survey of the State-of-the-Art," *Proc. Int. Work. Requir. Eng. Vis.*, pp. 46–55, 2009.
- [4] Z. Shakeri, H. Abad, and G. Ruhe, "Requirements Engineering Visualization : A Systematic Literature Review," *Requir. Eng. Conf.*, vol. IEEE 24th, no. 1, pp. 6–15, 2016.
- [5] S. K. Card, J. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. San Francisco: Morgan Kaufmann, 1999.
- [6] A. R. Teyseyre, "3D Requirements Visualization," *J. Comput. Sci. Technol.*, vol. 3, no. 2, pp. 45–51, 2003.
- [7] M. Cohn, *User Stories Applied: For Agile Software Development*. Boston: Addison Wesley, 2004.
- [8] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "The use and effectiveness of user stories in practice," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9619, pp. 205–222, 2016.
- [9] G. Chowdhury, "Natural language processing," *Annu. Rev. Inf. Sci. Technol.*, vol. 37, pp. 51–89, 2003.
- [10] F. P. J. Brooks, "No silver bullet-essence and accidents of software engineering," *Proc. IFIP Tenth World Comput. Conf.*, pp. 1069–1076, 1986.
- [11] J. Coughlan and R. D. Macredie, "Effective Communication in Requirements Elicitation: A Comparison of Methodologies," *Requir. Eng.*, vol. 7, no. 2, pp. 47–60, 2002.
- [12] S. Easterbrook and B. Nuseibeh, "Using ViewPoints for Inconsistency Management," *Softw. Eng. J.*, vol. 11, no. 2, pp. 31–43, 1996.
- [13] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," *Ann. Softw. Eng.*, vol. 3, no. 1, pp. 101–130, 1997.
- [14] G. Spanoudakis and A. Zisman, "Inconsistency management in software engineering: Survey and open research issues," *Handb. Softw. Eng. Knowl. Eng.*, vol. 1, pp. 329–380, 2001.
- [15] M. Soltani and E. Knauss, "Challenges of Requirements Engineering in AUTOSAR ecosystems," *2015 IEEE 23rd Int. Requir. Eng. Conf. RE 2015 - Proc.*, pp. 294–295, 2015.
- [16] W. N. Robinson and S. D. Pawlowski, "Managing requirements inconsistency with development goal monitors," *IEEE Trans. Softw. Eng.*, vol. 25, no. 6, pp. 816–835,

1999.

- [17] B. Nuseibeh, S. Easterbrook, and A. Russo, "Making inconsistency respectable in software development," *J. Syst. Softw.*, vol. 58, no. 2, pp. 171–180, 2001.
- [18] D. Zowghi and V. Gervasi, "On the interplay between consistency, completeness, and correctness in requirements evolution," *Inf. Softw. Technol.*, vol. 45, no. 14, pp. 993–1009, 2003.
- [19] J. Whyte, B. Ewenstein, M. Hales, and J. Tidd, "Visualizing Knowledge in Project-Based Work," *Long Range Plann.*, vol. 41, no. 1, pp. 74–92, 2008.
- [20] D. Berry, R. Gacitua, P. Sawyer, and S. Tjong, "The Case for Dumb Requirements Engineering Tools," *Proc. Int. Conf. Requir. Eng. Found. Softw. Qual.*, 2012.
- [21] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requir. Eng.*, vol. 21, no. 3, pp. 383–403, 2016.
- [22] M. Bano, "Addressing the Challenges of Requirements Ambiguity: A Review of Empirical Literature," *5th Int. Work. Empir. Requir. Eng. Emp. 2015 - Proc.*, no. July, pp. 21–24, 2016.
- [23] E. Cambria and B. White, "Jumping NLP Curves: A Review of Natural Language Processing Research," *IEEE Comput. Intell. Mag.*, vol. 9, no. 2, pp. 48–57, 2014.
- [24] G. Lucassen, M. Robeer, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Extracting conceptual models from user stories with Visual Narrator," *Requir. Eng.*, 2017.
- [25] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," *Proc. 1996 IEEE Symp. Vis. Lang.*, pp. 336–343, 1996.
- [26] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.
- [27] D. Pfitzner, V. Hobbs, and D. Powers, "A Unified Taxonomic Framework for Information Visualization," *Proc. Asia-Pacific Symp. Inf. Vis.*, vol. 24, pp. 57–66, 2003.
- [28] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, "Seven Guiding Scenarios for Information Visualization Evaluation Seven Guiding Scenarios for Information Visualization Evaluation," *Tech. Rep. DCS Univ. Calgary*, p. 17, 2011.
- [29] I. Sommerville and P. Sawyer, *Requirements engineering: a good practice guide*. John Wiley & Sons, 1997.
- [30] K. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.
- [31] IEEE Computer Society, Software Engineering Standards Committee, and IEEE-SA Standards Board, "Systems and software engineering — Life Cycle Processes — Requirements engineering (ISO/IEC/IEEE 29148)," pp. 1–83, 2011.
- [32] A. Davis, *Just enough requirements management: where software development meets marketing*. Davis, A. (2013). Just enough requirements management: where software development meets marketing. Addison-Wesley, 2013.
- [33] B. Wake, "INVEST in Good Stories, and SMART Tasks," 2003. [Online]. Available: <http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>. [Accessed:

18-Feb-2015].

- [34] D. Leffingwell and D. Widrig, *Managing Software Requirements: a Unified Approach*. Addison-Wesley Professional, 2000.
- [35] N. Juristo, A. M. Moreno, and A. Silva, "Is the European industry moving toward solving requirements engineering problems?," *IEEE Softw.*, vol. 19, no. 6, pp. 70–77, 2002.
- [36] F. Jackson, *Software requirements & specifications: a lexicon of practice, principles and prejudices*. ACM Press/Addison-Wesley Publishing Co, 1995.
- [37] W. N. Robinson, S. D. Pawlowski, and V. Volkov, "Requirements interaction management," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 132–190, 2003.
- [38] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, "Viewpoints : A Framework for Integrating Multiple Perspectives in System Development," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 2, pp. 31–57, 1992.
- [39] G. P. Mullery, "CORE - a Method for Controlled Requirement Specification," *Proc. 4th Int. Conf. Softw. Eng.*, pp. 126–135, 1979.
- [40] N. Omar, J. R. P. Hanna, and P. McKeivitt, "Heuristics-based Entity-Relationship Modelling through Natural Language Processing by," *Artif. Intell. Cogn. Sci. Conf.*, no. September, pp. 302–313, 2004.
- [41] S. Du and D. P. Metzler, "An Automated Multi-component Approach to Extracting Entity Relationships from Database Requirement Specification Documents," *Int. Conf. Appl. Nat. Lang. to Inf. Syst.*, pp. 1–11, 2006.
- [42] J. Mylopoulos, "Conceptual modelling and Telos," *Concept. Model. Databases, Case An Integr. view Inf. Syst. Dev.*, pp. 49–68, 1992.
- [43] M. L. G. Shaw and B. R. Gaines, "Knowledge Support Systems for Constructively Channeling Conflict in Group Dynamics," *AAAI-94 Work. Model. Confl. Manag. Coop. Probl. solving*, pp. 107–116, 1994.
- [44] R. Spence, *Information Visualization*. New York: Addison-Wesley, 2001.
- [45] R. A. Burkhard, "Towards a Periodic Table of Visualization Methods for Management," *Knowl. Inf. Vis.*, pp. 238–255, 2005.
- [46] R. Mazza, *Introduction to Information Visualization*. 2009.
- [47] C. Ware, *Information Visualization: Perception for Design*. Elsevier, 2012.
- [48] V. Gonzalez and A. Kobsa, "Benefits of information visualization systems for administrative data analysts," *Proc. Int. Conf. Inf. Vis.*, pp. 331–336, 2003.
- [49] J. M. DiMicco, A. Pandolfo, and W. Bender, "Influencing group participation with a shared display," *Proc. 2004 ACM Conf. Comput. Support. Coop. Work - CSCW '04*, pp. 614–623, 2004.
- [50] J. S. Yi, Y. Kang, J. Stasko, and J. Jacko, "Toward a deeper understanding of the role of interaction in information visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 6, pp. 1224–1231, 2007.
- [51] S. Supakkul and L. Chung, "Visualizing Non-Functional Requirements Patterns,"

- Fifth Int. Work. Requir. Eng. Vis.*, pp. 25–34, 2010.
- [52] D. Savio, P. C. Anitha, A. Patil, and O. Creighton, “Visualizing requirements in distributed system development,” *2012 2nd IEEE Int. Work. Requir. Eng. Syst. Serv. Syst. RESS 2012 - Proc.*, pp. 14–19, 2012.
- [53] P. Laurent, P. Mäder, J. Cleland-Huang, and A. Steele, “A taxonomy and visual notation for modeling globally distributed requirements engineering projects,” *Proc. - 5th Int. Conf. Glob. Softw. Eng. ICGSE 2010*, pp. 35–44, 2010.
- [54] D. Calleele, E. Neufeld, and K. Schneider, “Visualizing emotional requirements,” *2009 4th Int. Work. Requir. Eng. Vis. REV 2009*, pp. 1–10, 2009.
- [55] S. Lohmann, J. Ziegler, and P. Heim, “Involving End Users in Distributed Requirements Engineering 3 Web-based Elicitation of User Requirements,” *Eng. Interact. Syst.*, pp. 221–228, 2008.
- [56] A. van Lamsweerde, “Goal-oriented requirements engineering: a guided tour,” *Proc. Fifth IEEE Int. Symp. Requir. Eng.*, pp. 249–262, 2001.
- [57] N. Niu, S. Reddivari, and Z. Chen, “Keeping requirements on track via visual analytics,” *2013 21st IEEE Int. Requir. Eng. Conf. RE 2013 - Proc.*, pp. 205–214, 2013.
- [58] M. Bostock, “D3JS Library.” [Online]. Available: <https://d3js.org/>. [Accessed: 26-May-2017].
- [59] Almende B.V., “Vis.js.” [Online]. Available: <http://visjs.org/>. [Accessed: 26-May-2017].
- [60] S. Reddivari, S. Rad, T. Bhowmik, N. Cain, and N. Niu, “Visual requirements analytics: A framework and case study,” *Requir. Eng.*, vol. 19, no. 3, pp. 257–279, 2014.
- [61] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer, 2010.
- [62] C. Ahlberg, C. Williamson, and B. Shneiderman, “Dynamic queries for information exploration,” *Proc. SIGCHI Conf. Hum. factors Comput. Syst. - CHI '92*, vol. 92, pp. 619–626, 1992.
- [63] J. Venn, “On the diagrammatic and mechanical representation of propositions and reasonings,” *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, 1880.
- [64] B. Frederickson, “Venn.js.” [Online]. Available: <https://github.com/benfred/venn.js>. [Accessed: 28-May-2017].
- [65] T. jQuery Foundation, “jQuery library.” [Online]. Available: <https://jquery.com/>.
- [66] Twitter, “Bootstrap,” 2010. [Online]. Available: <http://getbootstrap.com/>.
- [67] L. J. Rips, E. J. Shoben, and E. E. Smith, “Semantic distance and the verification of semantic relations,” *J. Verbal Learning Verbal Behav.*, vol. 12, no. 1, pp. 1–20, 1973.
- [68] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *Adv. Neural Inf. Process. Syst.*, pp. 3111–3119, 2013.
- [69] F. De Sousa Webber, “Semantic Folding Theory And its Application in Semantic Fingerprinting,” 2015.

- [70] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [71] E. D. De Leeuw, J. J. Hox, and D. A. Dillman, *International Handbook of Survey Methodology*. 2008.
- [72] I. van der Schalk, “REVV tool,” 2017. [Online]. Available: <https://github.com/RELabUU>.
- [73] D. E. Damian and D. Zowghi, “Requirements Engineering challenges in multi-site software development organizations 1,” *Requir. Eng. J.*, vol. 8, no. February, pp. 149–160, 2003.
- [74] “StarUML.” [Online]. Available: <http://staruml.sourceforge.net>.
- [75] M. H. Awaad, H. Krauss, and H. D. Schmatz, *Advanced Praise for The Unified Modeling Language Reference Manual, Second Edition*, vol. 240, no. 3. 1978.
- [76] G. Walsh, “Distributed participatory design,” *Proc. 2011 Annu. Conf. Ext. Abstr. Hum. factors Comput. Syst. - CHI EA '11*, no. January 2008, p. 1061, 2011.

# Appendix A: Detailed Explanation of Assessed Visualizations

	Visualizing Non-Functional Requirements Patterns [51]	Visualizing Requirements in Distributed System Development [52]	Involving End Users in Distributed Requirements Engineering [55]
<b>User</b>			
<b>U1</b>	When multiple users use the tool issues may arise since different users may apply different definitions for the same goal.	This tool supports multiple stakeholders quite well as it visualizes the requirements of each stakeholder in a separate area.	This tool can be used by multiple stakeholders that are geographically distributed and supports a functionality to connect words with each other that share a similar meaning to avoid ambiguity between stakeholders.
<b>U2</b>	This tool has a learning curve since it uses specific modelling notations to represent and capture non-functional requirements knowledge. To become familiar with these notations some training is necessary.	The tool requires some training to understand how it works since the type of visualization is not common in software development and contains colors and different levels of which the meaning is unknown unless it is explained.	Tests show that the developed tool with its integrated approach is quickly understood by users.
<b>U3</b>	To make the tool truly real-time the modelled goals should be linked with variables that can be measured in real-time to show their impact on the measured goals.	Requirements are imported in the tool through an XML file. To make the tool truly real-time the tool should be linked with a requirement repository and automatically update the tool with new requirements and update the (traceability) status of existing requirements.	The tool is used by end-users who discover the need for new requirements. This data is obtained from the end-user through a simple form and is then directly send to the developers. However, further integration is possible by sending the obtained requirements automatically to a requirement management tool (e.g. a SCRUM board).
<b>U4</b>	This tool is implemented as an extension to StarUML, an open source UML modeling tool [74].	The use of Requirement Management tools is common in the software development process. This tool is meant as an	This tool is developed as a plugin in a web browser. As a result this tool can be used for any type of web application.

		add-on for such tools.	
<b>U5</b>	This tool is inspired by goal modeling [56], a well-known RE modeling approach, and by UML [75] a popular notation language within RE.	Requirement traceability is a common practice in software development. This tool offers a new type of visualization to track requirements.	The approach behind the tool is based on Distributed Participatory Design [76]. This approach describes the need for involvement of physical distributed end-users in the analysis and design of interactive systems.
<b>Data</b>			
<b>D1</b>	Models need to be created manually, processing large-scale inputs is therefore labour intensive.	This tool as difficulties in handling large sets of requirements as there is only a limited amount of space available once requirements reach the top level of the model.	The tool offers filtering options which makes the processing of obtained requirements easier. Despite these filters it's likely the overview of the requirements is quickly lost, especially when the requirements are not processed.
<b>D2</b>	This tool is focused on only visualizing patterns of non-functional requirements and ignores any relationships with other requirements artefacts such as agents and functional requirements.	This tool is only focused on visualizing function requirements. Other type of data such as non-functional requirements cannot be processed by this tool.	The user fills in a form to indicate what requirements he is missing. This requirement can be either a functional- as well as a non-functional requirement.
<b>D3</b>	This tool does not support automatic pre-processing.	The tool does not support automatic pre-processing.	The tool delivers the obtained requirements with meta-data such as keywords. However much improvement can still be made in the aspect of automatic pre-processing. The end-user sends the requirement in the form of a description. For developers it is time-consuming to analyse these descriptions. A possible improvement is that descriptions that include functional requirements could be automatically transformed



			into a more useful notation for developers, such as user stories.
<b>Model</b>			
<b>M1</b>	This tool helps to create a model of non-functional requirement goals	This tool helps to model the status of requirements of different stakeholders.	This tool helps to model the context of requirements by visualizing its meta-data.
<b>M2</b>	This tool helps to capture and reuse patterns for future models and as such somewhat assists in automatic model construction.	The model is automatically created by importing a XML file. To make the model construction completely automatic the tool should import the requirements automatically without human interference.	The tool automatically creates a model by using the data obtained from the form that the end-user submits. In addition to that the tool automatically extracts the location and device of the end-user. Further improvement on the aspect of automatic model construction is possible by for example employing NLP techniques for automatically connecting similar and possible duplicate, requirements.
<b>M3</b>	This tool can be used to extend a model with additional goals and patterns.	The model is constructed by using the data it is provided. It is not possible to manipulate this data nor is it possible to customise the model by for example changing the colors of the blocks that represent the requirements.	Requirements that are similar can be manually connected with each other. Other than that the model cannot be extended or customized.
<b>M4</b>	This tool does not support model traceability.	This tool does not support model traceability.	This tool does not support model traceability.
<b>Visualization</b>			
<b>V1</b>	This tool allows to create sub patterns by modeling 'Specialization-of Relationships', 'Part-of Relationships' and 'Occurrence-of Relationships' and allows to view such sub	In this tool each face of the 3D object represents a stakeholder perspective.	This tool allows to obtain new requirements from stakeholders. The list in which these requirements appear can be sorted by the author's name as well as by

	patterns in separate views.		the role of an author. This somewhat helps a developer to gain insight in the needs of different stakeholders.
V2	This tool allows to inspect different views in separate windows. However it is not possible to compare different views in the same window.	The user can hold down the mouse button to change the point of perspective from which the 3D object is shown. This function allows the user to focus on a specific face of the pyramid.	This tool does not support inter-view navigation.
V3	This tool allows to zoom into elements of interest and to move the screen.	This tools allows to turn the 3D object and allows to view the 3D object from a top down view.	This tool somewhat supports browsing as it is possible tool browse the Google Map which displays the locations of stakeholders from who the requirements are obtained.
V4	This tool does not support searching.	This tool does not support searching.	In this tool the user is able to use a search engine to find requirements. The input of this search engine is a keyword. The search engine then checks whether this keyword occurs in the title or description of a requirement.
V5	This tool does not support query drilling.	This tool does not support query drilling.	This tool does not support query drilling.
V6	The tool does not support filtering actions.	This tool allows to filter requirements based on their implementation status.	This tool somewhat allows filtering as it allows to focus on specific keywords by using either the search engine or word cloud. As a result all requirements that do not include the given keyword are filtered out. In addition to that the Google Map can be used to zoom into specific location and as a result filter out all requirements in the Google Map that do not belong to the location on which is zoomed in.
V7	This tool uses labels to annotate the meaning of elements.	This tool does not employ annotation.	This tool uses labels to annotate the meaning of fields

			in the form that the user uses to submit a requirement and in the table overview of the developer to annotate the meaning of the data in the columns of the table.
<b>Knowledge</b>			
<b>K1</b>	This tool is used to visualize patterns and can be used to create alternative patterns with the intend to compare them which somewhat helps to detect anomalies.	This tool is used to visualize the priority of requirements from different stakeholders.	This tool supports anomaly detection as it allows to obtain requirements from end-users and thus reveals the short-comings of a system.
<b>K2</b>	This tool does not support the option to display detailed information.	This tool allows to inspect a requirement and view its dependencies with other requirements.	Each obtained requirement contains a detailed explanation from the end-user and additional meta-data.
<b>K3</b>	This tool supports hypothesis-based reasoning as it allows to change the weight towards goals which helps to view the effect that this change has.	The tool somewhat allows hypothesis-based reasoning as it allows to change the priority and/or stakeholder of a requirement (by uploading a new requirements file) and in doing so help to determine whether the sequence in which the requirements are processed is improved.	This tool does not support hypothesis-based reasoning.
<b>K4</b>	This tool supports scenario-based reasoning as it allows to create alternative views.	The tool somewhat allows scenario-based reasoning as it allows to change the priority and/or stakeholder of a requirement (by uploading a new requirements file) and in doing so create multiple visualizations which could then be compared with each other.	This tool does not support scenario-based reasoning.
<b>K5</b>	This tool ultimately helps to detect weaknesses and vulnerabilities in a system; this somewhat motivates a user to	This tool gives an overview of the priority and stakeholders that belong to a requirement. The tool somewhat motivates an user to	This tool somewhat supports developers to make actionable-decisions. The tool obtains requirements from

	tackle these problems.	take action if it appears the sequence in which requirements are processed can be improved by changing the priorities and/or stakeholder to which they belong to.	end-users. Once a requirement is obtained the developers need to decide whether it is implemented or not. Although this decision-making is done outside the tool it does motivate developers to discuss a requirement once it is obtained.
--	------------------------	---	--

# Appendix B: The Requirements Identification

## Ambiguity study description

Experimenter: Ivor van der Schalk

Affiliation: Utrecht University

**Introduction:** You are invited to participate in a research study that investigates the automatic detection of syntactically ambiguous requirements.

**Instructions:** In this study you will be given a list of 12 pairs of concepts with a number of requirements in which these concepts appear. You need to indicate how likely you think the requirements that contain these concepts are **syntactically ambiguous**. You can choose between the options 'impossible', 'unlikely', 'likely' and 'certain'.



**Explanation:** Ambiguity refers to situations in which a requirement may have more than one valid interpretation. Syntactically ambiguous requirements can be found by looking at the concepts that are found in the requirements. **If the concepts use different terminology but (possibly) share the same meaning they are syntactically ambiguous.**

**Example:** Below follows an example of a pair of concepts with syntactically ambiguous requirements.

Media Element	Media	
<ul style="list-style-type: none"> <li>- As a Visitor, I'm able to navigate through the album so that I view all <b>media elements</b>.</li> <li>- As a Visitor, I'm able to view textual information on the selected <b>media element</b>.</li> <li>As an Administrator, I'm able to add new <b>media elements</b> to the select gallery.</li> <li>- As an Administrator, I'm able to edit existing <b>media elements</b> of a particular gallery so that I update the content.</li> <li>- As an Administrator, I'm able to remove existing <b>media elements</b> of a particular gallery so that I keep the album up to date.</li> </ul>	<p>As a Visitor, I'm able to select an album so that I view the <b>media</b> in the album.</p>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;"><input type="radio"/> impossible</div> <div style="text-align: center;"><input type="radio"/> unlikely</div> <div style="text-align: center;"><input type="radio"/> likely</div> <div style="text-align: center;"><input checked="" type="radio"/> certain</div> <div style="text-align: center;"><input type="radio"/> don't know</div> </div>






In this example it is likely the concepts 'Media' and 'Media Element' are syntactically ambiguous because they share the same meaning. When looking at the requirements we find an album contains *media elements*. We also find an album contains *media*. Since the concept media is not used in any other context we can assume they refer to the same thing. We therefore indicate that the requirements in which these concepts appear are **certain** to be syntactically ambiguous.

## Appendix C: List of concept pairs (1)

Media gallery	Gallery	
<p>As a Visitor, I'm able to view the <b>media gallery</b> so that I see interesting photo's about the Event Region.</p> <p>As an Administrator, I'm able to manage the <b>media gallery</b> so that I add or remove content.</p> <p>As an Administrator, I'm able to delete an existing album so that its removed from the <b>media gallery</b>.</p>	<p>As an Administrator, I'm able to add new media elements to the select <b>gallery</b>.</p> <p>As an Administrator, I'm able to edit existing media elements of a particular <b>gallery</b> so that I update the content.</p> <p>As an Administrator, I'm able to remove existing media elements of a particular <b>gallery</b> so that I keep the album up to date.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>
Profile Page	Profile	
<p>As a Visitor, I'm able to click on a person's profile card So that I open their <b>profile page</b>.</p> <p>As a User, I'm able to open the interactive map from a person's <b>profile page</b> so that I see that particular plot location.</p> <p>As a Visitor, I'm able to open the interactive map from the person's <b>profile page</b> so that I know the location of the person's event plot.</p> <p>As a Visitor, I'm able to view the added stories (if any) on the <b>profile page</b> so that I learn more about the person.</p> <p>As a User, I'm able to add text to a person's <b>profile page</b>.</p> <p>As a User, I'm able to add an image to a person's <b>profile page</b>.</p> <p>As a User, I'm able to add a description to a person's <b>profile page</b>.</p> <p>As a User, I'm able to edit the content that I added from a person's <b>profile page</b> to update the information.</p> <p>As a User, I'm able to delete content (which I added) from a person's <b>profile page</b> So that I remove information that I no longer want to share. As an</p>	<p>As a Visitor, I'm able to search for people in the ABC database so that I review the <b>profiles</b> of the positioned people.</p> <p>As an Administrator, I'm able to manage people so that I add edit or delete <b>profiles</b>. As a Visitor, I'm able to view the <b>profile</b> of a particular person so that I identify that person and add additional content to their <b>profile</b>.</p> <p>As a Visitor, I'm able to navigate to the next and previous <b>profile</b> so that I easily scan through the search results.</p> <p>As a Visitor, I'm able to close the selected <b>profile</b> so that I return to the search results or homepage.</p> <p>As a User, I'm able to add content to the selected <b>profile</b>.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>

Administrator I'm able to delete content (which a user added) from a person's <b>profile page</b> .		
<b>Media Gallery</b>	<b>Media Category</b>	
As a Visitor, I'm able to view the <b>media gallery</b> so that I see interesting photo's about the Event Region. As an Administrator, I'm able to manage the <b>media gallery</b> so that I add or remove content. As an Administrator, I'm able to delete an existing album so that its removed from the <b>media gallery</b> .	As a Visitor, I'm able to navigate back to the <b>media categories</b> .	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> impossible   unlikely   likely   certain   don't know
<b>Category</b>	<b>Text</b>	
As a Visitor, I'm able to see an overview of available <b>categories</b> so that I select a particular <b>category</b> . As a Visitor, I'm able to see an overview of available albums within the select <b>category</b> so that I select a particular album.	As a User I'm able to add <b>text</b> to a person's profile page .	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> impossible   unlikely   likely   certain   don't know
<b>Text</b>	<b>Plot</b>	
As a User, I'm able to add <b>text</b> to a person's profile page.	As a User, I'm able to view an interactive map of the Event Region so that I view intact mass event locations listed by <b>plot #</b> .	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> impossible   unlikely   likely   certain   don't know
<b>Location</b>	<b>Event Location</b>	
As a Visitor, I'm able to open the interactive map from the person's profile page so that I know the <b>location</b> of the person's event plot.	As a User, I'm able to view an interactive map of the Event Region so that I find <b>event locations</b> . As a User, I'm able to view an interactive map of the Event Region so that I view intact mass <b>event locations</b> listed by plot #.	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> impossible   unlikely   likely   certain   don't know
<b>Video</b>	<b>Profile Card</b>	
As a User, I'm able to add an <b>video</b> (YouTube Vimeo link) to a person's profile page.	As a Visitor, I'm able to click on a person's <b>profile card</b> so that I open their profile page.	<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> impossible   unlikely   likely   certain   don't know



Event Plot	Plot Location	
<p>As a Visitor, I'm able to open the interactive map from the person's profile page so that I know the location of the person's <b>event plot</b>.</p> <p>As a Visitor, I'm able to see a map of the <b>event plot</b> so that I know where the person has been positioned.</p>	<p>As a User, I'm able to click a particular <b>plot location</b> from the map and thereby perform a search of people associated with that plot number.</p> <p>As a User, I'm able to open the interactive map from a person's profile page so that I see that particular <b>plot location</b>.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>
Media Category	Category	
<p>As a Visitor, I'm able to navigate back to the <b>media categories</b>.</p>	<p>As a Visitor, I'm able to see an overview of available <b>categories</b> so that I select a particular <b>category</b>.</p> <p>As a Visitor, I'm able to see an overview of available albums within the select <b>category</b> so that I select a particular album.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>
Password	Story	
<p>As a User, I can login using my email address and <b>password</b> so that I get access to the user -only features of the website.</p> <p>As a User, I am able to set a new <b>password</b> so that I login.</p>	<p>As a Visitor, I'm able to view the added <b>stories</b> (if any) on the profile page so that I learn more about the person.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>
Description	Image	
<p>As a User, I'm able to add a <b>description</b> to a person's profile page.</p>	<p>As a User, I'm able to add an <b>image</b> to a person's profile page.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>
Password	Password Reset	
<p>As a User, I can login using my email address and <b>password</b> so that I get access to the user -only features of the website.</p> <p>As a User, I am able to set a new <b>password</b> so that I login.</p>	<p>As a User, I can request a <b>password reset</b> so that I am still able to login whenever I forgot my password.</p>	 <p>impossible   unlikely   likely   certain   don't know</p>

## Appendix D: List of concept pairs (2)

Media	Media Gallery	
<p>As a Visitor I'm able to select a album so that I view the <b>media</b> in the album.</p>	<p>As a Visitor, I'm able to view the <b>media gallery</b> so that I see interesting photo's about the Event Region.</p> <p>As an Administrator, I'm able to manage the <b>media gallery</b> so that I add or remove content.</p> <p>As an Administrator, I'm able to delete an existing album So that its removed from the <b>media gallery</b>.</p>	<p style="text-align: center;"> <input type="radio"/>   <input type="radio"/>   <input type="radio"/>   <input type="radio"/>   <input type="radio"/> </p> <p style="text-align: center;">             impossible   unlikely   likely   certain   don't know           </p>
Page	Content	
<p>As a Visitor, I'm able to navigate the site through site wide navigation menus so that I always quickly open the <b>page</b> I'm looking for.</p> <p>As a Visitor, I'm able to navigate the site through the site wide top menu so that I always quickly open the <b>page</b> I'm looking for.</p> <p>As a Visitor, I'm able to navigate the site through the site wide footer menu so that I always quickly open the <b>page</b> I'm looking for.</p>	<p>As a Visitor, I'm able to view the profile of a particular person so that I identify that person and add additional <b>content</b> to their profile.</p> <p>As a Visitor, I'm able to see who (display name) added the <b>content</b> and when so that I know more about the contributor of the <b>content</b>.</p> <p>As a User, I'm able to add <b>content</b> to the selected profile.</p> <p>As a User, I'm able to edit the <b>content</b> that I added from a person's profile page to update the information.</p> <p>As a User, I'm able to delete <b>content</b> (which I added) from a person's profile page so that I remove information that I no longer want to share.</p> <p>As an Administrator, I'm able to delete <b>content</b> (which a user added)</p>	<p style="text-align: center;"> <input type="radio"/>   <input type="radio"/>   <input type="radio"/>   <input type="radio"/>   <input type="radio"/> </p> <p style="text-align: center;">             impossible   unlikely   likely   certain   don't know           </p>

	<p>from a person's profile page.</p> <p>As an Administrator, I'm able to manage the media gallery so that I add or remove <b>content</b>.</p> <p>As an Administrator, I'm able to edit existing media elements of a particular gallery so that I update the <b>content</b>.</p> <p>As an Administrator, I'm able to edit an existing event so that I update the <b>contents</b>.</p>	
<b>Password Reset</b>	<b>Video</b>	
As a User, I can request a <b>password reset</b> so that I am still able to login whenever I forgot my password.	As a User, I'm able to add an <b>video</b> (YouTube Vimeo link) to a person's profile page .	<input type="radio"/> impossible know <input type="radio"/> unlikely <input type="radio"/> likely <input type="radio"/> certain <input type="radio"/> don't know
<b>Event Plot</b>	<b>Plot</b>	
<p>As a Visitor, I'm able to open the interactive map from the person's profile page so that I know the location of the person's <b>event plot</b>.</p> <p>As a Visitor, I'm able to see a map of the <b>event plot</b> so that I know where the person has been positioned.</p>	As a User, I'm able to view an interactive map of the Event Region so that I view intact mass event locations listed by <b>plot #</b> .	<input type="radio"/> impossible <input type="radio"/> unlikely <input type="radio"/> likely <input type="radio"/> certain <input type="radio"/> don't know
<b>Content</b>	<b>Profile</b>	
<p>As a Visitor, I'm able to view the profile of a particular person so that I identify that person and add additional <b>content</b> to their profile.</p> <p>As a Visitor, I'm able to see who (display name) added the <b>content</b> and when so that I know more about the contributor of the <b>content</b>.</p> <p>As a User, I'm able to add <b>content</b> to the selected profile.</p>	<p>As a Visitor, I'm able to search for people in the ABC database so that I review the <b>profiles</b> of the positioned people.</p> <p>As an Administrator, I'm able to manage people so that I add edit or delete <b>profiles</b>.</p> <p>As a Visitor, I'm able to view the <b>profile</b> of a particular person so that I identify that person and add additional content to their <b>profile</b>.</p>	<input type="radio"/> impossible <input type="radio"/> unlikely <input type="radio"/> likely <input type="radio"/> certain <input type="radio"/> don't know

<p>As a User, I'm able to edit the <b>content</b> that I added from a person's profile page to update the information .</p> <p>As a User, I'm able to delete <b>content</b> (which I added) from a person's profile page so that I remove information that I no longer want to share.</p> <p>As an Administrator, I'm able to delete <b>content</b> (which a user added) from a person's profile page.</p> <p>As an Administrator, I'm able to manage the media gallery so that I add or remove <b>content</b>.</p> <p>As an Administrator, I'm able to edit existing media elements of a particular gallery so that I update the <b>content</b>.</p> <p>As an Administrator, I'm able to edit an existing event so that I update the <b>contents</b>.</p>	<p>As a Visitor, I'm able to navigate to the next and previous <b>profile</b> so that I easily scan through the search results.</p> <p>As a Visitor, I'm able to close the selected <b>profile</b> so that I return to the search results or homepage.</p> <p>As a User, I'm able to add content to the selected <b>profile</b>.</p>	
<b>News Section</b>	<b>Description</b>	
<p>As an Administrator, I'm able to manage the list of news items so that I keep the <b>news section</b> up to date.</p>	<p>As a User, I'm able to add a <b>description</b> to a person's profile page.</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>
<b>Description</b>	<b>Text</b>	
<p>As a User, I'm able to add a <b>description</b> to a person's profile page .</p>	<p>As a User, I'm able to add <b>text</b> to a person's profile page.</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>
<b>Information</b>	<b>Link</b>	
<p>As a Visitor, I'm able to review key <b>information</b> of all found people so that I have an indication of the person's details.</p> <p>As a Visitor, I'm able to view the general <b>information</b> of a person so that I learn</p>	<p>As a User, I'm able to add an video (YouTube Vimeo <b>link</b>) to a person's profile page.</p> <p>As a User, I'm able to add an audio fragment (Soundcloud <b>link</b>) to a person's profile page.</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>

<p>more about an individual's event on Event Region.</p> <p>As a User, I'm able to edit the content that I added from a person's profile page to update the <b>information</b> .</p> <p>As a User I'm able to delete content (which I added) from a person's profile page so that I remove <b>information</b> that I no longer want to share.</p> <p>As a Visitor. I'm able to view textual <b>information</b> on the selected media element .</p> <p>As an Administrator, I'm able to edit the <b>information</b> of an existing album so that I update it.</p>		
<b>Contact Form</b>	<b>Audio Fragment</b>	
<p>As a Visitor, I am able to use the <b>contact form</b> so that I contact the administrator.</p>	<p>As a User I'm able to add an <b>audio fragment</b> (Soundcloud link) to a person's profile page.</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>
<b>Plot</b>	<b>Plot Location</b>	
<p>As a User, I'm able to view an interactive map of the Event Region so that I view intact mass event locations listed by <b>plot #</b>.</p>	<p>As a User, I'm able to click a particular <b>plot location</b> from the map and thereby perform a search of people associated with that plot number.</p> <p>As a User, I'm able to open the interactive map from a person's profile page so that I see that particular <b>plot location</b>.</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>
<b>Information</b>	<b>Profile</b>	
<p>As a Visitor, I'm able to review key <b>information</b> of all found people so that I have an indication of the person's details.</p> <p>As a Visitor, I'm able to view</p>	<p>As a Visitor, I'm able to search for people in the ABC database so that I review the <b>profiles</b> of the positioned people.</p> <p>As an Administrator, I'm</p>	<p><input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/></p> <p>impossible   unlikely   likely   certain   don't know</p>

<p>the general <b>information</b> of a person so that I learn more about an individual's event on Event Region.</p> <p>As a User, I'm able to edit the content that I added from a person's profile page to update the <b>information</b>.</p> <p>As a User I'm able to delete content (which I added) from a person's profile page so that I remove <b>information</b> that I no longer want to share.</p> <p>As a Visitor, I'm able to view textual <b>information</b> on the selected media element.</p> <p>As an Administrator, I'm able to edit the <b>information</b> of an existing album so I can update it.</p>	<p>able to manage people so that I add edit or delete <b>profiles</b>.</p> <p>As a Visitor, I'm able to view the <b>profile</b> of a particular person so that I identify that person and add additional content to their <b>profile</b>.</p> <p>As a Visitor, I'm able to navigate to the next and previous <b>profile</b> so that I easily scan through the search results.</p> <p>As a Visitor, I'm able to close the selected <b>profile</b> so that I return to the search results or homepage.</p> <p>As a User, I'm able to add content to the selected <b>profile</b>.</p>	
<b>Detail</b>	<b>Plot</b>	
<p>As a Visitor, I'm able to review key information of all found people so that I have an indication of the person's <b>details</b>.</p> <p>As an Administrator, I can edit the <b>details</b> of a User .</p>	<p>As a User, I'm able to view an interactive map of the Event Region so that I view intact mass event locations listed by <b>plot #</b>.</p>	<p> <input type="radio"/> impossible   <input type="radio"/> unlikely   <input type="radio"/> likely   <input type="radio"/> certain   <input type="radio"/> don't know </p>

# Appendix E: Correlation study survey

1. Pick 2 concept pairs and indicate for each pair why you think the requirements in which they appear are ambiguous:

**Concept pair:** .....

**Ambiguous because:**

.....  
.....

**Concept pair:** .....

**Ambiguous because:**

.....  
.....

2. Pick 2 concept pairs of which you are not sure whether they are ambiguous and indicate why:

**Concept pairs:** .....

**Not sure because:**.....

.....

**Concept pairs:** .....

**Not sure because:**.....

.....

3. Indicate any comments you may have regarding this study:

.....  
.....  
.....

# Appendix F: Evaluation Description

Experimenter: Ivor van der Schalk

Affiliation: Utrecht University

**Introduction:** You are invited to participate in a research study that investigates the performance of a novel visualization for finding ambiguities and missing user stories.

**Instructions:** In this study you will be given a set of user stories that originate from a Software Architecture course. In this set of user stories you need to find ambiguities and missing user stories by using the assigned method (the visualization *or* pen & paper).

- Write down any ambiguities found on the '*List of Ambiguities*';
- Write down any missing user stories found on the '*List of Missing User Stories*'.

**Context:** the given set of user stories is about the FETCH system. The goal of this system is: to support festivals and event houses in organizing the ticketing and payments during events. This system contains the following features:

- Online ticket sales for events;
- Ticket sales on site, including payment;
- Support events on multiple locations at the same time;
- Works in the open field as well as in buildings;
- Checking of tickets at the entrance(s) of the event location(s) with a (branded) RFID chip on a card or wristband;
- Cashless payments at different shops, machines and bars at the event location(s) with the same RFID chip;
- Overview and connection to bookkeeping software of event organizers;
- Customization of branding to promote the event.

**Explanation:** To help explain on how to find ambiguities and missing user stories take the example user stories below:

**User story A:** As a User, I want to edit a photo in the gallery.

**User story B:** As a User, I want to delete a picture from the media gallery.

- **Finding missing user stories:** A user story is missing when its absence blocks a prerequisite of another user story.
- **Finding ambiguities:** Ambiguities in user stories can be found by looking at the concepts that are found in the user stories. **If the concepts use different terminology but (possibly) share the same meaning they are ambiguous.** Such concepts can be found by looking for synonyms and hyponyms. For example, in *user story A* and *user story B* the concepts “photo” and “picture” are synonyms. A hyponym shares a is-a relationship with another concept. For example, in *user story A* and *user story B* the concepts “gallery” and “media gallery” are hyponyms (a media gallery **is** a type of gallery).



## Appendix G: List of Ambiguities Tool

Use the table below to note down every concept pair that is ambiguous.

#	Concept A	Concept B
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		

## Appendix H: List of Ambiguities Pen & Paper

Use the table below to note down every concept pair that is ambiguous.

#	Concept A	ID	Concept B	ID
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

# Appendix I: List of Missing User Stories

Use the table below to note down every missing user story.

#	User story
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	

## Appendix J: List of User Stories

Event Organizer	
E1	I want to publish basic information on the event I am organizing online, so that I can reach out to more people.
E2	I want to define several locations for the event I am organizing, so that I can appeal to customers in different regions.
E3	I want to organize an online ticket sale for my events, to make it easier for people to buy the tickets.
E4	I want to sell tickets at the event location, to allow people without internet access to buy tickets.
E5	I want to define the ticket price for each ticket category.
E6	I want to define the time schedules included for each ticket category.
E7	I want to define the event location included for each ticket category.
E8	I want to be able to check ticket validity at event site entrance using RFID chip technology, to prevent unauthorized people from attending my event.
E9	I want to sell different items at the event venues.
E10	I want to customize the branding material for each of my events, so that the branded RFID material is produced.
E11	I want to set the credit type so that all shown balances and prices for one or multiple events are in the given credit type.
E12	I want to update event details so that I can edit mistakes or add new or changed information.
E13	I want to view a previously created event so that I have an overview of the created event.
E14	I want to put a constraint on which roles are allowed to enter an event.
E15	I want to put constraints on the number of people for an event.
E16	I want to set an event duration.
E17	I want to set which roles are allowed to enter at a given entry point.
E18	I want to receive a message when a type of event I'm currently running is changed so that I am aware of a change in event type.
E19	I want to see how many people have checked into the event.
E20	I want to see how many people bought an event ticket.
E21	I want to connect the FETCH system to my bookkeeping software so that the systems' data is synchronized.
E22	I want to disconnect the FETCH system from my bookkeeping software.
E23	I want to receive an invoice so that I know the amount due for using FETCH.
E24	I want to print an invoice so that I can distribute it within my own organization.
E25	I want to export an invoice so that I can distribute it within my own organization.

<b>Event Attendant</b>	
C1	I want to access information on the event I am attending online, so that I am able to find my way to the event venue.
C2	I want to be able to transfer my purchased tickets to another person, as I may not be able to attend the event.
C3	I want to be able to upgrade my ticket category online.
C4	I want to be able to upgrade my ticket category at the event site using RFID chip technology.
C5	I want to be able to put more money on my RFID chip at the event locations.
C6	I want to be able to check the net balance on my RFID chip at the event locations.
C7	I want to view information on different ticket categories online, so that I can decide which ticket category is best for me.
C8	I want to purchase items using my chip credit so that I can use the cashless payment system.
C9	I want to know how much credit was deducted in an item purchase so that I know the new balance on my chip.
C10	I want to purchase multiple entry tickets so that I can gain access to events.
C11	I want to see entry ticket details.
C12	I want to choose a payment method so that I can purchase an entry ticket.
C13	I want to receive a purchased entry ticket.
C14	I want to search for events.
C15	I want to use an entry ticket to gain entrance to an event so that I can enter that event.
C16	I want to log into FETCH using my account so that I can manage my chip and purchase entry tickets.
C17	I want to logout of FETCH.
C18	I want to receive a message when I do not have enough balance for a purchase.
C19	I want to attach my account to an existing chip so that I can reuse a chip for multiple events.
C20	I want to detach my account from a chip so that I can stop using the service or change over chips.
C21	I want to set a password for my account so that I can use my chip ID.
C22	I want to cancel my account so that I can stop using FETCH.
C23	I want to receive a message when my account is created so that I know that my account exists.
C24	I want to provide my personal details for my account so that entry tickets are personalized.
C25	I want to change my personal details so that I can change updated details or fix mistakes.
<b>Distributor</b>	
D1	I want to give out an account to the event attendant so that the event attendant is able to use the system.
D2	I want to optionally give out a chip to the event attendant so that the event attendant can use the system using a provided chip.

<b>Merchandiser</b>	
M1	I want to create new types of events so that these types of events can be created by event organisers.
M2	I want to change types of events.
M3	I want to delete a type of event so that the type of event can stop being used.
<b>Financial Employee</b>	
F1	I want to input an invoice for an event organizer.
F2	I want to set an invoice due date.
F3	I want to change an invoice due date so that I can fix mistakes made or apply a change in contract.
F4	I want to update the invoice status to paid when.
F5	I want to receive a message when the invoice status is unpaid at the due date so that I am aware that an invoice is unpaid.
F6	I want to receive a monthly report of financial transactions.
F7	I want to handle fiat financial transactions.

## Appendix K: List of Criteria

<b>User</b>	
<b>U1</b>	The tool is meant to be used by requirement engineers. The tool should support multiple users as collaboration plays an important role in agile software development. The use of a video wall fosters collaboration between users and allows users to interact with the displayed model. Further improvement on this aspect is possible as it currently requires for requirement engineers to be present at the same physical location and time to collaborate.
<b>U2</b>	It is important the tool can be used without heavy training as it would otherwise be an obstacle for a requirement engineer to start using the tool. Therefore the tool presented in this work employs help icons to explain the tool's features.
<b>U3</b>	The tool presented in this work does not support real-time performance.
<b>U4</b>	The tool is integrated into an existing software development environment as it is meant to be used in an agile software development environment as it uses user stories as input.
<b>U5</b>	The tool uses practitioner guidelines as it supports in-viewpoint checking as it allows to focus on one viewpoint at the time and supports inter-viewpoint checking as it allows to compare multiple viewpoints with each other. Furthermore, for finding ambiguities and missing requirements the visualization of the tool is inspired by the concept of concept classification [43].
<b>Data</b>	
<b>D1</b>	A set of requirements can contain many requirements. It is therefore important that the tool supports large-scale input. The tool is able to process a large number of requirements, however it should be noted that the more requirements there are the longer it takes for the tool to construct the model. This is due to the fact that it takes some time to compute the ambiguity score between concepts.
<b>D2</b>	The tool does not support heterogeneous input types as it only accepts data from user stories that is extracted by using the tool of [1].
<b>D3</b>	The tool does not support automatic pre-processing.
<b>Model</b>	
<b>M1</b>	A set of requirements typically represents multiple viewpoints. As such the tool presented in this work explicitly visualizes each viewpoint in the model. Each viewpoint is represented as a large circle in which the associated concepts and requirements can be found. A limitation is that due to the nature of a Venn diagram only a limited number of viewpoints can be displayed
<b>M2</b>	Since it is not feasible for a requirement engineer to manually construct a model from a (large) set of user stories within a reasonable time it is important that the tool supports automatic model construction. For this reason the tool employs the tool of [1] to automatically extract concepts and relationships from a set of user stories. Once these concepts and relationships are loaded into the tool

	it automatically construct a model based on the given input. Further improvement on this aspect is possible by integrating the functionalities of the tool proposed in this work and the tool of [1] into one tool.
M3	The tool does not support model extension or customization features. In the future allowing users to customize the colors and shapes of elements could be useful to allow the user to change the displayed information to its personal needs.
M4	The tool somewhat supports model traceability as it allows to create a model from a set of requirements and allows to view such a model back again by using the provided link to the visualization. By uploading a set of requirements multiple times over a period of time one can create multiple models that help to visualize how the set of requirements has changed over time. This task is currently done mostly manual. In the future the tool could incorporate new features that further support model traceability.
<b>Visualization</b>	
V1	The tool allows to inspect each viewpoint separately which allows the requirement engineer to gain insight in the view a stakeholder holds about a system and to conduct in-viewpoint checking.
V2	Supporting Inter-view navigation is important as it allows to compare different views with each other and thus support inter-viewpoint checking. The tool supports inter-view navigation as it is possible to add and remove viewpoints from the model. A limitation of the visualization is that it only allows to display a limited number of viewpoints due to the nature of a Venn diagram.
V3	Due to the fact that a set of requirements can contain many requirements it is important that the tool supports browse features. As such, the tool allows to zoom into elements of interest and to move the display.
V4	The tool does not support searching. In the future the tool could incorporate a search functionality that allows to easily find requirements and concepts back.
V5	The tool does not support query-drilling.
V6	Since a set of requirements contains much information and only portions of such information can be of interest it is important that the tool supports filtering. As such, the tool allows to filter viewpoints, association relationships, concept states and concepts based on their ambiguity score.
V7	To help explain the requirement engineer what the displayed elements are about it is important that the tool uses annotations to describe the displayed elements. For this purpose role nodes contain a label that describes the role it represents. In addition to that concept nodes contain a label that describes the concept it represents. Furthermore, association icons contain a one character label to help the requirement engineer recognize what the association icon is about.
<b>Knowledge</b>	
K1	To help identify ambiguities and missing requirements it is important that the visualization offers the requirement engineer new insights. For this purpose all concept that are ambiguous according to the RAI method are colored red. Furthermore, the tool helps to reveal the concept state of a concept by positioning a concept in a certain area which ultimately helps to identify ambiguities and missing requirements.



<b>K2</b>	The tool supports detailed explanation as it allows to inspect the user stories associated to a concept by clicking on its concept node.
<b>K3</b>	The tool somewhat supports hypothesis based reasoning as it is possible to drag concept nodes from one area in the model to the other. This allows a requirement engineer to see what the model looks like when a concept is removed or added from a viewpoint.
<b>K4</b>	The tool somewhat supports scenario-based reasoning as it allows to upload different set of requirements. By uploading different sets of requirements that are about the same system one can create different scenarios (e.g. one could upload a set of requirements that is based on a low budget and a set of requirements that is based on a high budget). By comparing the resulting models a requirement engineer is able to analyse the different scenarios.
<b>K5</b>	The tool motivates the user to look at the concepts that are colored red and thus to solve ambiguities. Furthermore, the tool motivates the user to wonder whether the displayed concepts appear in the correct area and thus if their associated concept state is correct.

# Detection of Ambiguity and Incompleteness in RE via Information Visualization and NLP

Ivor van der Schalk, Garm Lucassen, Fabiano Dalpiaz  
Department of Information and Computing Sciences  
Utrecht University, The Netherlands

**Abstract**—The identification of requirements defects such as ambiguity, unclarity and incompleteness is by no means a trivial task. The literature in requirements engineering (RE) has shown that combining humans’ cognitive and analytical capabilities with automated reasoning is an effective combination to achieve such result. In this paper, we introduce a novel software tool that blends natural language processing (NLP) and information visualization techniques with the aim of identifying potential ambiguities and missing requirements. We use a Venn-diagram visualization to organize the concepts that appear in user story requirements according to the extent to which they are shared by multiple viewpoints. Our approach relies on NLP in two ways: (i) we employ our prior tool Visual Narrator to extract concepts and relationships from a collection of user stories, and (ii) we highlight potential ambiguities by combining state-of-the-art semantic similarity and relatedness algorithm. We illustrate feasibility and applicability of our prototype platform with the aid of case study requirements from the software industry.

**Keywords**—Ambiguity; Incompleteness; NLP; Information Visualization; User Stories

## I. INTRODUCTION

Defects such as ambiguity, inconsistency, unclarity and incompleteness are common phenomena in requirements engineering [1], [2], [3], and they can potentially lead to misunderstandings between stakeholders, overlooked requirements, and unsound implementations that do not meet the real needs.

Identifying requirements defects is not trivial. On the one hand, automated approaches based on natural language processing (NLP) require trade-offs between precision and recall [4], [2], [5], also because NLP technology is still mostly at the syntactic level [6]. On the other hand, manual approaches that rely solely on human intelligence are obviously not scalable. Combining these two approaches is therefore essential.

We make a step toward the synergistic use of NLP and human analysis in the context of our research on user stories and agile requirements. In particular, our input consists of the concepts and relationships that are automatically extracted by our Visual Narrator tool [7] from a collection of user stories.

In this paper, we introduce a novel software tool that blends natural language processing (NLP) and information visualization (IV) techniques with the aim of identifying potential ambiguities and missing requirements in a set of user stories. The paper makes four concrete contributions:

- We construct a framework for identifying potential ambiguity and incompleteness based on RE literature about inconsistent viewpoints and conceptual modeling (Sec. II).

- To aid stakeholders analyze multiple viewpoints—and identify potentially missing requirements—we propose a Venn-diagram visualization of the concepts appearing in requirements that highlights the shared concepts and the specific ones (Sec. III).
- To identify possibly ambiguous requirements, we combine state-of-the-art NLP algorithms that assess the semantic similarity between couples of concepts in their usage context, and we use color gradients to visualize the computed ambiguity scores (Sec. IV).
- To demonstrate feasibility, we develop a prototype based on Web 2.0 technologies and apply it to real-world data sets from the industry. (Sec. V)

This paper opens research avenues for the use of NLP and IV techniques that can support requirements engineers and other stakeholders in identifying defects in requirements by harnessing the synergy between human cognitive capabilities and artificial intelligence techniques (see Sec. VI).

## II. A CONCEPTUAL FRAMEWORK FOR POTENTIAL AMBIGUITY AND INCOMPLETENESS

Based on previous RE literature on viewpoints and inconsistency, we construct a conceptual framework for identifying potential ambiguity and incompleteness in requirements.

### A. Viewpoints and inconsistency

The different types of users of a software system are typically interested in distinct aspects of the system; for example, a website’s administrators care about content creation and structuring, while readers are mostly interested in accessing existing content. According to Mullery [8] a *viewpoint* is a description of one stakeholder’s perception of a system, and it consists of concepts and inter-relationships between them.

The existence of viewpoints does inevitably lead to inconsistencies and conflicts in stakeholders’ requirements. Recognizing these and reconciling their differences is an essential task in requirements analysis [9]. To resolve these inconsistencies, it is necessary to (i) check the consistency of the specification within the viewpoint (in-viewpoint checks), and (ii) check the consistency of the specification with those maintained by other viewpoints (inter-viewpoint checks) [10].

### B. Conceptual modeling in RE

Requirements analysis can be aided by the creation of *conceptual models* [11] that depict a holistic view of the system

instead of relying on lengthy textual requirements documents. Among other uses, conceptual models can be used to explore the concepts that belong to different viewpoints.

We can identify two families of conceptual modeling techniques for RE: (i) requirements modeling languages (e.g., goal modeling [12] and problem frames [13]) for representing the requirements in a diagram, and (ii) conceptual overviews of the entities and relationships that can be (automatically) extracted from natural language requirements [14], [15], [7]).

We build on the latter family of approaches. In particular, our starting point consists of the outputs of the Visual Narrator tool [7] that extracts the main concepts and relationships from user story requirements (“As a *role*, I want *means*, so that *ends*”). While we made this choice due to the high precision and recall of this tool, the approach can be generalized to those NL notations for requirements from which concepts and relationships can be extracted with sufficient accuracy.

### C. From inconsistency to ambiguity and incompleteness

Inconsistencies in requirements arise when the same concepts are used with different meanings, e.g., in different viewpoints. Shaw and Gaines [16] introduce four degrees of inconsistency categorized according to two aspects: (i) *distinction* referring to concepts’ semantics, and (ii) *terminology* referring to the terms in which a concept is described. Consider for example the word “bank”, which can mean either a financial institution or a ground alongside a body of water. The four inconsistency states are:

- 1) *Consensus*: same distinction, same terminology, therefore no inconsistency. For example, the term “bank” is used twice to refer to financial institution.
- 2) *Correspondence*: same distinction, different terminology. For instance, both “bank” and “treasury” are used to refer to a financial institution.
- 3) *Conflict*: different distinction, same terminology. For instance, “bank” is used twice, once to refer to a financial institution and once to denote ground.
- 4) *Contrast*: different distinction, different terminology. For instance, “treasury” is used to refer to a financial institution and “bank” is use to refer to a ground.

A requirement is ambiguous when it has more than one valid interpretation [17]. The link with inconsistent viewpoints is thus clear: when a requirement set contains at least one concept in a correspondence or conflict state, there is ambiguity.

Fig. 1 provides an overview of our framework (that extends [16]) that defines potential ambiguity and incompleteness (i.e., missing concepts that may be relevant for a viewpoint) starting from inconsistency states.

We illustrate the framework with the help of the following real-world user stories from the WebCompany data set [7] (concepts are emphasized in the text):

- U1. As a *Visitor*, I am able to view the *media gallery* so that I can see interesting *photos* about the *event region*.
- U2. As an *Administrator*, I am able to edit existing *media elements* of a particular *gallery*, so that I can update the *content*.

- U3. As a *User*, I am able to add *content* to the selected *profile*.
- U4. As a *Visitor*, I am able to use the *contact form*, so that I can contact the *administrator*.

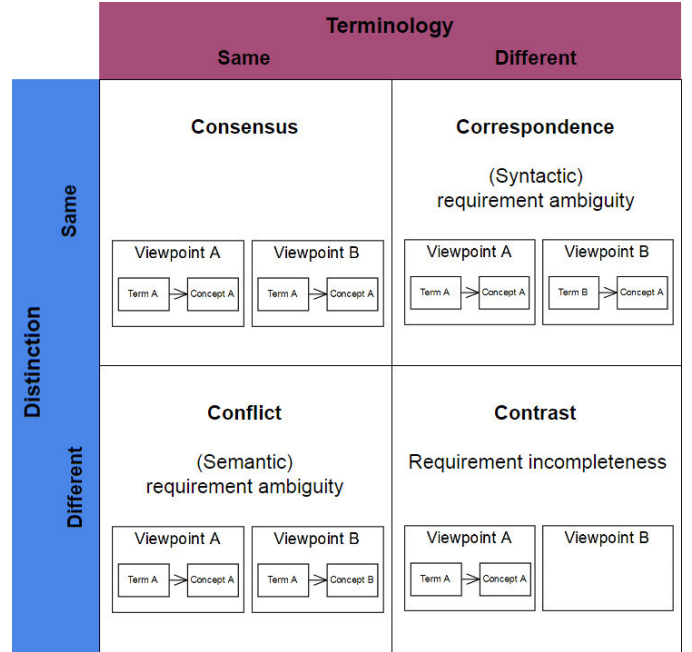


Fig. 1. Conceptual framework for identifying potential ambiguity and incompleteness from inconsistency states (extending [16]).

The consensus state does not lead to any ambiguity. For example, *Visitor* is described in the same terminology both in U1 and U4 to refer to a visitor of the website.

Syntactic ambiguity *may* occur in the correspondence state: two different terms are used to refer to the same concept. The term *media gallery* in U1 and the term *gallery* in U2 are likely to refer to the same concept, and this may make those user stories ambiguous.

Semantic ambiguity *may* occur in the conflict state: the same term is used to refer to different abstractions. In U2 the term *content* refers specifically to a media element while in U3 the term *content* refers to either a text, description, image, video or audio fragment.

Incompleteness *may* occur in the contrast state, i.e., when one viewpoint refers to concepts that do not appear in another viewpoint. U4 includes *contact form* that the visitor uses to contact the administrator. However, there is no user story that specifies how the administrator can respond to this action.

For the purposes of this paper, we focus on syntactic ambiguity that arises from the correspondence state. We leave the exploration of semantic ambiguity to future work.

### III. PINPOINTING AMBIGUITY AND INCOMPLETENESS VIA INFORMATION VISUALIZATION

Building on the framework proposed in Sec. II (see Fig. 1), we present a novel use of IV techniques that is intended for analysts to explore multiple viewpoints and to identify ambiguity and incompleteness.

Our starting point are the conceptual models automatically generated by the Visual Narrator tool, which provide a holistic overview of the concepts in a user story collection. These models, however, quickly become too large and complex for a human to grasp [7].

To improve the situation, we visualize viewpoints by means of a Venn diagram, which is a suitable means for displaying overlapping elements [18]. Fig. 2 provides an example where the concepts of the stakeholders *Administrator*, *User* and *Visitor* are shown alongside their overlap. We refer to this visualization as *Venn Concept Viewpoints*.

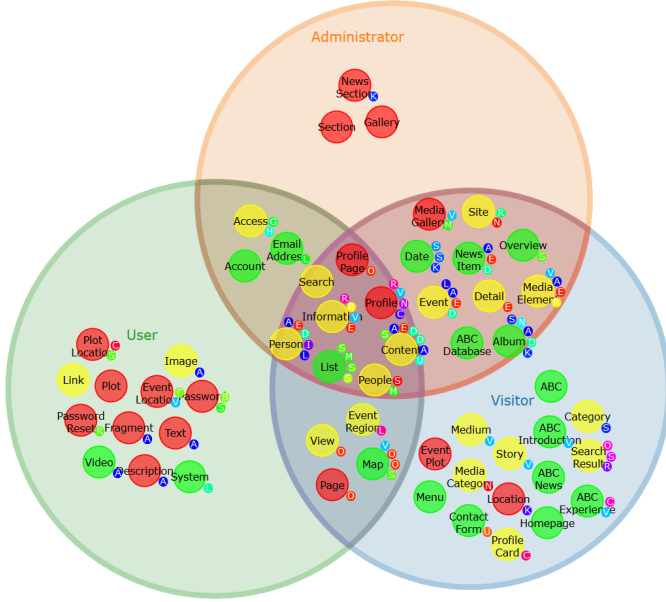


Fig. 2. Venn Concept Viewpoints visualization displaying three viewpoints.

Venn Concept Viewpoints enable a requirements engineer to examine how the concepts of multiple unique stakeholders interrelate. Consider the Venn diagram in Fig. 3 made up of three unique stakeholder viewpoints, whose intersection produces 7 areas (A–G). Areas A, C, G include concepts that appear in a single viewpoint. Area E contains the concepts that are shared by all three viewpoints. Areas B, D, F include the concepts that appear in exactly two viewpoints.

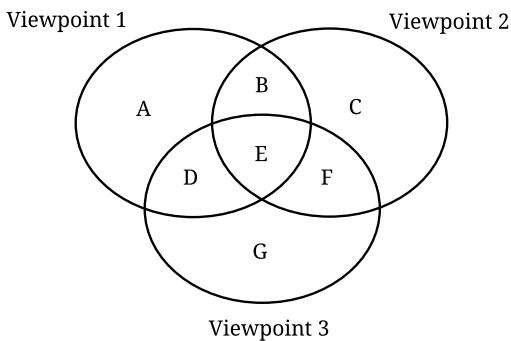


Fig. 3. The 7 areas (A–G) of a Venn Concept Viewpoints visualization.

Note that concepts in areas shared by different viewpoints are in either a consensus or conflict state, as shown in Fig. 4.

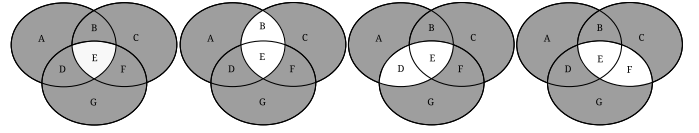


Fig. 4. The white areas denote either consensus or conflict state.

Similarly, concepts that appear in areas *not* shared by different viewpoints can be in either correspondence or contrast state as shown in Fig. 5. When comparing all three viewpoints, the concepts in all areas except E are in a correspondence or contrast state (ABCDFG). Comparing viewpoints 1 2, 1 3 and 2 3, the concepts in correspondence or contrast state are those in areas ACD, ACF, BCF, BCD, CDF, and ABC, respectively.

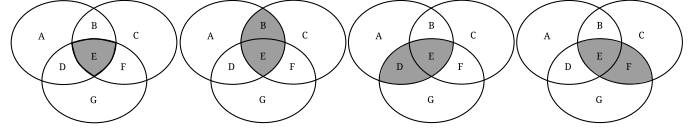


Fig. 5. The white areas denote either correspondence and contrast state.

In his work, Shneiderman [19] describes seven dimensions for effective IV. We describe those dimensions that are relevant for the Venn Concept Viewpoints visualization.

#### A. Overview

The overview dimension describes how to view the entire collection of elements. The overview in the Venn Concept Viewpoints includes three elements: (1) the role node, (2) the concept node and (3) the association relationship icon.

Table I describes each element. A viewpoint is represented by the combination of a role node, the concept nodes that appear in a role node and the association relationship icons that relate to these concept nodes.

TABLE I. ELEMENTS IN THE VENN CONCEPT VIEWPOINTS OVERVIEW.

Element	Description
Role	A role is represented by a <b>role node</b> . This is a large circle with a unique color. This circle contains a label with the name of the role that it represents.
Concept	A concept is represented by a <b>concept node</b> , a small circle that appears within a role node. Its color is determined by the highest ambiguity score (see Sec. IV) that it shares with another concept. This circle contains a label with the name of the represented concept.
Association relationship	An association relationship is represented by an <b>association relationship icon</b> . This is a small icon that appears next to the concept that it relates to. The icon of each association relationship has its own unique color. The icon also contains a label of the first character of the association relationship that it represents.

#### B. Filter

The filter dimension describes how unwanted items can be removed from the displayed set. We propose three filter types:

- 1) *Concept state filter* removes the concepts in a consensus/conflict state or those in a correspondence/contrast state from the display. This helps the requirements engineer focus on specific shared

concepts or unique viewpoint elements, respectively. Illustrated in Fig. 6.

- 2) *Viewpoint filter* removes some viewpoints from the display, so that the analyst can focus on the remaining viewpoints. For example, Fig. 6 shows only two viewpoints.
- 3) *Ambiguity filter* shows the elements within a given ambiguity score range. This can be useful to better examine the elements with high ambiguity score or to double check those with low-medium score. This is illustrated in Fig. 7. A possible ambiguity filter range is illustrated in Fig. 8: the actual color ranges can be customized for a given domain.

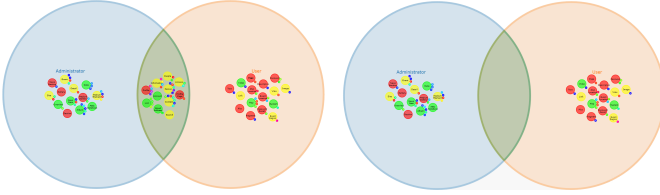


Fig. 6. Illustration of the concept state filter: on the right-hand side, concepts in a consensus/conflict state are removed from the display.



Fig. 7. Illustration of the ambiguity filter: on the right-hand side, only concepts with an ambiguity score above 0.4 are shown.

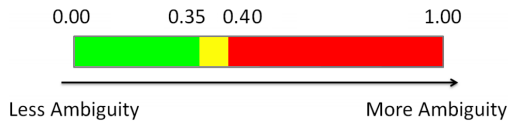


Fig. 8. A possible ambiguity filter range: the closer the score of a concept pair is to 1, the more ambiguous they are (see Sec. IV). Here, scores under 0.35 are green (unlikely ambiguity), scores between 0.35 and 0.40 are yellow (possible ambiguity), and scores above 0.40 are red (probable ambiguity).

### C. Details-on-demand

The details-on-demand dimension describes the features that can be used to get the details of a selected group, subgroup or item. We support the inspection of concepts and of association relationships, see Table II for an overview.

Fig. 9 illustrates how the association relationship and concept ambiguity details are displayed. On the left side, the association relationship inspection reveals that the icon represents the *view* association relationship, and that this association relationship belongs to the *Visitor* viewpoint. On the right side, the ambiguity score of the concept *media gallery* is inspected, revealing that high ambiguity is due to the concept *gallery*. Fig. 10 exemplifies detailing a user story concept, showing all the user stories in which the concept *gallery* appears.

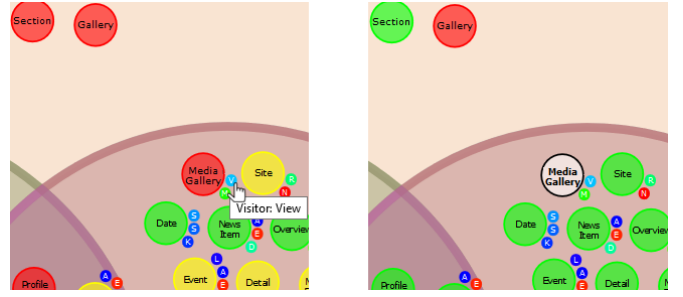


Fig. 9. Illustrations of details-on-demand for the association relationship *view* (left) and ambiguity details of the concept *media gallery* (right).

TABLE II. FILTER OPTIONS IN VENN CONCEPT VIEWPOINTS.

Detail-on-demand feature	Description
Association relationship details	The details of an association relationship can be inspected by clicking on the association relationship icon. This opens a small pop-up window that display the full name of the association relationship and the role to which the relationship belongs to.
Concept ambiguity details	The ambiguity that a concept shares with other concepts can be inspected by clicking on a concept. This applies boldface emphasis to the concept and sets a white background. The color of all the other concepts is changed based on the ambiguity score that they share with the selected concept.
Concept user story details	The user stories in which a concept appear can be inspected by double clicking on a concept. Doing so opens a pop-up window which lists all those user stories. The detailed concept is given a black background, and other concepts in those stories are given a blue background to help the analyst recognize the concepts within each user story.

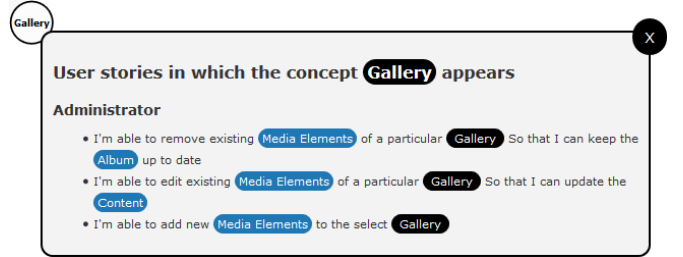


Fig. 10. An example of inspecting the user stories of a concept.

## IV. AUTOMATED IDENTIFICATION OF SYNTACTIC AMBIGUITY VIA NLP

A key component of our IV approach presented in Sec. III is its capability to determine potentially ambiguous couples of concepts that appear in the requirements. This ability is used to determine the concept node color in the overview, in the ambiguity filter, and in the ambiguity details-on-demand.

To automatically detect correspondence state concepts we rely on NLP tools that calculate the *semantic distance* between two concepts: a numerical representation of how close or distant two terms are in their meaning [20]. Current state-of-the-art NLP tools, such as Word2Vec, employ word statistics based on in which contexts a word is used to establish their semantic similarity on a 0-1 scale [21]. The higher the similarity score, the more likely it is that the two concepts share the same meaning. The Semantic Folding Theory (SFT) is an alternative novel method that uses a neuroscience based mechanism of distributional semantics [22].

All these functions rely on the notion of an *ambiguity score*, a number between 0 and 1 that represents how likely it is that two concepts are ambiguous. In particular, our approach relies on the ambiguity detection (AD) algorithm shown in Fig. 11. As its input, the AD algorithm takes a list of concepts extracted from a user story collection, e.g., using the Visual Narrator tool [7]. Then, the following six steps are applied to associate the ambiguity score with all pairs of concepts:

- 1) Create unique pairs of all the extracted concepts;
- 2) Calculate semantic similarity score for each concept pair by invoking a semantic similarity algorithm. For example, our prototype implementation calls the Cortical.io REST API<sup>1</sup> that employs SFT;
- 3) For each concept, build its context: the set of all user stories that the concept occurs in;
- 4) For each concept pair, build its context that includes all and only the user stories where only one of the two concepts occurs;
- 5) Calculate the relatedness score for each concept context pair (again, Cortical.io can be used);
- 6) Compute the ambiguity score that combines concept similarity and context relatedness (see Equation 1).

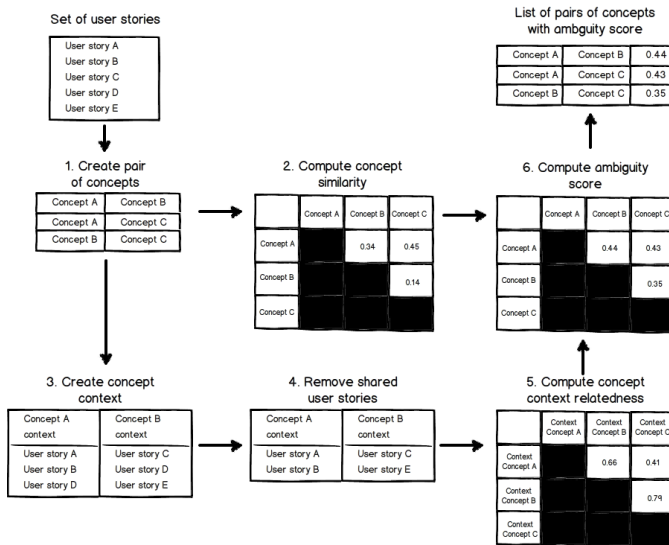


Fig. 11. The Ambiguity Detection algorithm for calculating the ambiguity score for pairs of concepts.

Our heuristic excludes shared user stories from the concept pair context because two concepts that frequently appear in the same user stories are not necessarily similar. For example, the concepts *email address* and *password* share the same context as they both appear in user story 5 (U5), however they do not share the same meaning.

U5. As a *User*, I can login using my *email address* and *password*, so that I get access to the user-only *features* of the *website*.

Equation 1 determines the ambiguity score  $aScore$  for two concepts  $c_i$  and  $c_j$  (and their contexts  $ctx_i$  and  $ctx_j$ ) as the weighted average of the semantic concept similarity ( $simScore(c_i, c_j)$ ) and the relatedness of those concepts'

contexts  $relnScore(ctx_i, ctx_j)$ . We currently assign a weight of 2 to  $simScore$  and a weight of 1 to  $relnScore$ .

$$aScore = \frac{simScore(c_i, c_j) \times 2 + relnScore(ctx_i, ctx_j)}{3} \quad (1)$$

We tested the reliability of our  $aScore$  via a study that tested the alignment between the AD algorithm and human-judged similarity. In this study, we employed the *WebCompany* data set that consists of 98 user story requirements. Eight participants with a background in information science participated in this study. Each participant had to fill in a questionnaire that contained 12 concepts pairs with their contexts. Four of these concept pairs have a low ambiguity score, 4 of these concepts pairs have a medium ambiguity score and 4 of these concept pairs have a high ambiguity score according to the algorithm. For each concept pair, the participant had to indicate how likely they perceived the concept pair to be ambiguous, using the scale "Impossible", "Unlikely", "Likely", "Certain" or "Don't know". In total, 24 concept pairs were processed by the 8 participants. A Pearson correlation on the data shows there is a *strong* and *significant positive correlation* between the scores computed by the algorithm and by the participants,  $r = .806$ ,  $p < .001$ .

## V. FEASIBILITY

To show the feasibility of our approach, we developed a prototype tool that implements the Venn Concept Viewpoints visualization of Sec. III and the AD algorithm of Sec. IV. The tool is a Web 2.0 environment that is built on top of the well-known Bootstrap framework, relies on D3.js<sup>2</sup> for the visualization component, and calls the REST API of cortical.io to compute the semantic similarity between concepts and relatedness between contexts. The tool can be accessed online<sup>3</sup>.

We applied the tool to two sets of real-world user stories: besides *WebCompany*, we chose the *CMS-Company* data set [7]. After importing the data sets into the tool, the visualization shows immediately what viewpoints are the most prominent and contain the most concepts. For example, in the *CMS-Company* data set, the viewpoint of the *Channel Manager* contains only a few concepts, and many of those are shared with the viewpoint of the *Marketer* (Fig. 12): this shows that these two roles have a strong connection.

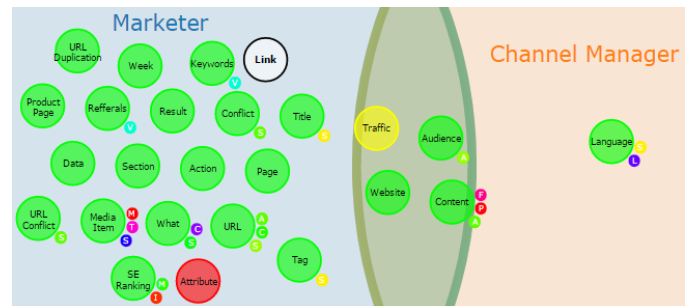


Fig. 12. An excerpt from the *CMS-Company* data set.

<sup>1</sup><http://api.cortical.io/>

<sup>2</sup><https://d3js.org/>

<sup>3</sup><http://eve-digital.nl/interactive-narrator/>

Using the filter, we investigated each viewpoint separately. Most viewpoints included some red-highlighted concepts indicating ambiguity, indicating intra-viewpoint inconsistencies: most viewpoints in both data sets are not self-consistent. An example can be found in the *User* viewpoint of *WebCompany*: the concepts *text* and *description* share a high ambiguity score. When inspecting the requirements that refer to these concepts, both contain a very similar fragment “adding a text” and “adding a description”. This example of syntactic ambiguity reveals a duplicate requirement.

Although the displayed association relationships have a not-too-shallow learning curve, they proved to be useful for finding missing requirements. For example, in the *WebCompany* data set the association relationship *add* is represented by a blue circle containing the letter ‘A’. The viewpoint that represents the *User* role contains many concepts having that icon (Fig. 2). We could see that *User* has many ‘add’ relationships, but there are no requirements referring to ‘edit’ or ‘delete’ actions: this is a likely occurrence of incompleteness.

Another useful method for finding missing requirements was to explore for concepts in a contrast state. An example is the missing *contact form* concept in the *Administrator* viewpoint in Sec. II. Another example from the *WebCompany* data set is the *account* concept, which appears in the viewpoints of the *Administrator* and the *User* but not in *Visitor*; the latter role needs to create an account to become a user.

## VI. FUTURE DIRECTIONS

This paper opens the doors for future work that combines information visualization techniques with automated reasoning algorithms such as NLP. While here we explored user stories and we focused on incompleteness and ambiguity, other types of requirements can be considered as well as other defect types.

We will empirically validate the usefulness of our developed tool for requirements engineers to identify ambiguities and missing requirements, both by visualizing existing data sets to product managers and developer and by conducting experiments that test the effectiveness of our technique compared to manual analysis of text. A specific case we want to study is that of multiple user story owners that collaborate at long distance (e.g., in global RE [23]).

The algorithm for the detection of ambiguity can certainly be improved and has to be tested and tuned based on empirical data: the challenge is that of avoiding over-fitting. We intend to study whether the use of domain ontologies can lead to a deeper understanding of the requirements and their relationships. Finally, we are interested in the use of IV techniques to facilitate the transition from RE to architectural design.

## REFERENCES

- [1] D. M. Berry, E. Kamsties, and M. M. Krieger, “From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity,” School of Computer Science, University of Waterloo, Canada, Tech. Rep., 2001.
- [2] M. Bano, “Addressing the challenges of requirements ambiguity: A review of empirical literature,” in *Proc. of the International Workshop on Empirical Requirements Engineering (EmpiRE)*, 2015, pp. 21–24.
- [3] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini, “Using NLP to detect requirements defects: An industrial experience in the railway domain,” in *Proc. of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer, 2017, pp. 344–360.
- [4] D. Berry, R. Gacitua, P. Sawyer, and S. Tjong, “The Case for Dumb Requirements Engineering Tools,” in *Proc. of the International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, ser. LNCS. Springer, 2012, vol. 7195, pp. 211–217.
- [5] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, “Improving Agile Requirements: The Quality User Story Framework and Tool,” *Requirements Engineering*, vol. 21, no. 3, pp. 383–403, 2016.
- [6] E. Cambria and B. White, “Jumping nlp curves: a review of natural language processing research [review article],” *IEEE Computational Intelligence Magazine*, vol. 9, no. 2, pp. 48–57, 2014.
- [7] M. Robeer, G. Lucassen, J. Van der Werf, F. Dalpiaz, and S. Brinkkemper, “Automated Extraction of Conceptual Models from User Stories via NLP,” in *Proceedings of the International Requirements Engineering Conference (RE)*. IEEE, 2016.
- [8] G. P. Mullery, “CORE - a Method for Controlled Requirement Specification,” in *Proc. of the International Conference on Software Engineering (ICSE)*, pp. 126–135, 1979. [Online]. Available: <http://dl.acm.org/citation.cfm?id=800091.802932>
- [9] I. Sommerville and P. Sawyer, “Viewpoints: principles, problems and a practical approach to requirements engineering,” *Annals of Software Engineering*, vol. 3, no. 1, pp. 101–130, 1997. [Online]. Available: <http://dx.doi.org/10.1023/A:1018946223345>
- [10] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke, “Viewpoints: A Framework for Integrating Multiple Perspectives in System Development,” *International Journal of Software Engineering and Knowledge Engineering*, vol. 2, no. 1, pp. 31–57, 1992.
- [11] J. Mylopoulos, “Conceptual modelling and Telos,” in *Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development*, P. Loucopoulos and R. Zicari, Eds. New York: Wiley, 1992, pp. 49–68.
- [12] A. Van Lamsweerde, “Goal-oriented requirements engineering: A guided tour,” in *Proc. of the IEEE International Symposium on Requirements Engineering (SRE)*. IEEE, 2001, pp. 249–262.
- [13] M. Jackson, *Problem frames: analyzing and structuring software development problems*. Addison-Wesley Longman, 2000.
- [14] N. Omar, J. Hanna, and P. McKeivitt, “Heuristics-Based Entity-Relationship Modelling through Natural Language Processing,” in *Proc. of the Irish Conference on Artificial Intelligence & Cognitive Science (AICS)*, 2004, pp. 302–313.
- [15] S. Du and D. P. Metzler, “An Automated Multi-component Approach to Extracting Entity Relationships from Database Requirement Specification Documents,” in *Proc. of the International Conference on Applications of Natural Language to Information Systems (NLDB)*, ser. LNCS, vol. 3999. Springer, 2006, pp. 1–11.
- [16] M. Shaw and B. Gaines, “Knowledge support systems for constructively channeling conflict in group dynamics,” in *Proc. of the AAAI-94 Workshop on Models of Conflict Management and Cooperative Problem Solving*, 1994.
- [17] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.
- [18] L. Micallef, “Visualizing set relations and cardinalities using venn and euler diagrams,” Ph.D. dissertation, University of Kent, 2013.
- [19] B. Shneiderman, “The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations,” in *Proc. of the IEEE Symposium on Visual Languages (VL)*, 1996, pp. 336–343.
- [20] L. J. Rips, E. J. Shoben, and E. E. Smith, “Semantic distance and the verification of semantic relations,” *Journal of Verbal Learning and Verbal Behavior*, vol. 12, no. 1, pp. 1–20, 1973.
- [21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. of the International Conference on Neural Information Processing System (NIPS)*, 2013, pp. 3111–3119.
- [22] F. De Sousa Webber, “Semantic folding theory and its application in semantic fingerprinting,” *arXiv preprint arXiv:1511.08855*, 2015.
- [23] D. E. Damian and D. Zowghi, “Re challenges in multi-site software development organisations,” *Requirements Engineering*, vol. 8, no. 3, pp. 149–160, 2003.