

# **Communicating Conflict and Ambiguity in Requirements Engineering**

Thesis submitted to Lancaster University in fulfilment of the requirements for  
the degree of Doctor of Philosophy in Computer Science

by

**Maria Cabral Diogo Pinto Albuquerque**

Licenciatura in Applied Mathematics and Computing,

Universidade Técnica de Lisboa, Portugal

M.Sc. in Computing,

Universidade de Lisboa, Portugal

Supervisor: Prof. Awais Rashid

December, 2013

# Acknowledgements

I would like to thank Prof. Awais Rashid for his supervision, which was outstanding, with his excellent qualities as a researcher, a person, and a supervisor. In particular I would like to thank him for his positive attitude, always very well balanced with a demand for quality. I am also thankful for his constant care and support.

I would like to thank Prof. Mehmet Akşit and Prof. Peter Sawyer for the corrections proposed which improved the quality of the thesis.

I would like to thank my husband Carlos, for his love, for standing by me throughout this PhD thesis project, and accepting all the efforts needed so that I could complete this work. I am also thankful for his help with the MATLAB routines to obtain the graphical drawings of the Jig3P prototype tool.

I would like to thank my “adoptive” parents Maria Deolinda and Victor, the grandparents of my daughters, for their love, generosity and support. Without their help this project would not have been possible while maintaining our family in a healthy equilibrium.

I would like to thank my friends Raquel Mateus Palma, Paula Mourão Moreira, and Vera Burnay Batalha (and the many friends from *Comunidade Verbum Dei de Lisboa*) for their support in maintaining my personal equilibrium in such a variety of life dimensions. I am thankful to Joana Vasconcelos for her friendship and good laughing moments. I am thankful to Margarida César for her friendship, support and for sharing good music to listen while working.

I would like to thank all the AOSD team at Lancaster University, with a first reference to Phil Greenwood for his excellent work in proof-reading and revising the first version

of this thesis. I would like to thank Ruzanna Chitchyan, Safoora Shakil Khan, Pauline Anthony, Steffen Zschaler, Alberto Sardinha and Joost Noppen, whose PhD thesis inspired me to develop research in the imperfection information area. I am thankful for their great welcome each time I visited Lancaster, and the sincere interest in my work and fruitful discussions.

I would like to thank Prof. Daniel Berry, from University of Waterloo for the valuable discussions and advice at the Doctoral Symposiums of RE'08 and REFSQ'11.

I would like to thank from Lancaster University, Prof. Alan Dix and Dr. John Mariani for some interesting discussions, Yvonne Rigby and Sarah Brown for their friendship.

I would like to thank ISCTE-IUL, Lisbon University Institute, especially my department and colleagues for the support. In particular I would like to thank Nuno David and my ISCTE-IUL colleagues who are also Lancastrians, Rui Lopes and José André Moura for their friendship and support.

I would like to thank my 17 ISCTE-IUL colleagues who generously participated in the experiments for my work, and for the great suggestions they provided.

I would like to thank my father António and my new “adoptive mother” Maria do Rosário, my sisters and brothers (some in law): Leonor, Isabel, Rita, Pedro, António, and Jorge, and the new generation: Miguel, Marta, Gonçalo, and Leonor, for their love and support.

Last but not the least; I would like to dedicate this thesis:

- In memory of my mother Maria Tereza, for the love and life she gave me, and for showing me where the real Love and Life are.
- To my daughters Sofia and Joana, wishing that they have the possibility to do with their life what they really love to; and that in any case, they are happy.

## **Declaration**

This dissertation is the result of my own work, under the supervision of Prof. Awais Rashid, and has not been submitted for the award of a higher degree at any other university. Therefore, I acknowledge full responsibility for the work herein presented.

Signature:

# Communicating Conflict and Ambiguity in Requirements Engineering

Thesis submitted to Lancaster University in fulfilment of the requirements for the degree of  
Doctor of Philosophy in Computer Science by

**Maria Cabral Diogo Pinto Albuquerque**

Licenciatura in Applied Mathematics and Computing, Universidade Técnica de Lisboa, Portugal  
M.Sc. in Computing, Universidade de Lisboa, Portugal

Supervisor: Prof. Awais Rashid

December 2013

## **Abstract**

Effective requirements engineering in the presence of imperfection remains a major research problem. There is a lack of metaphors to aid communication about such imperfections during consultation with stakeholders.

The aim of this thesis research is to improve the identification, communication, and handling of ambiguity and conflict in non-functional requirements, inadvertently introduced during the RE process.

The thesis proposes a new approach based in the jigsaw puzzle metaphor, which is a novel contribution in the area of visual metaphors, and as a communication mechanism to make conflict and ambiguity explicit during stakeholder consultation meetings. This metaphor is based on jigsaw puzzles, where each puzzle piece represents a requirement. When the requirement text contains ambiguities and/or conflicts with other requirements, the respective puzzle pieces almost fit together but

not perfectly. The approach presents heuristics to identify the most pertinent conflicts and ambiguities to handle and thus to make explicit through the badly-fitting matches. The gamming nature of the jigsaw puzzle metaphor, the fact it presents an easy to understand and learn language, as well as the analogy with misshapen graphical visualization (the badly-fitting matches) to represent that there is a problem, and its adequacy to a creative task as RE is; altogether are key characteristics that contribute to the adequacy and success of the jigsaw puzzle metaphor when used in stakeholder consultation meetings.

In fact the jigsaw puzzle metaphor used together with the proposed method for conducting the consultation meetings with the stakeholders proved successful in:

- Increasing effectiveness when compared with text presentation.
- Fostering team work and communication, and improving commitment of stakeholders in co-authoring of requirements and co-responsibility in ambiguity and conflict handling.
- Promoting a relaxed environment to improve team cooperation and creativity.

A key contribution of this thesis is its focus on separating the processing of the information about the imperfection from the issue of communicating that imperfection. Such a separation, though critical, has not been proposed to date.

# Table of contents

Acknowledgements .....	1
Declaration .....	3
Abstract.....	4
Table of contents .....	6
List of figures .....	12
List of tables .....	14
1 Introduction.....	15
1.1 Introduction .....	15
1.2 Problem .....	16
1.2.1 Ambiguity and conflict in software requirements .....	16
1.2.2 Characteristics of requirements engineering .....	19
1.2.3 Definitions of ambiguity and conflict in requirements.....	20
1.2.4 Causes of ambiguity and conflict in requirements .....	26
1.3 Scope of the thesis.....	27
1.4 Limitations of state-of-the-art.....	28
1.4.1 Imperfect information support through modelling languages .....	28
1.4.2 Ambiguity and conflict identification and handling .....	30
1.4.3 Visual communication metaphors in requirements engineering.....	32
1.4.4 Open challenges in handling of ambiguity and conflict in RE .....	33
1.5 Aim and objectives .....	34
1.6 Proposed approach.....	35

1.7 Novel elements of the thesis work .....	36
1.8 Outline of the thesis .....	38
2 Requirements engineering and imperfect information support .....	40
2.1 Introduction .....	40
2.2 Requirements elicitation and creativity .....	41
2.3 Imperfect information support through modelling languages .....	43
2.3.1 Fuzzy logic to specify and analyse imprecise, and vague requirements, and the relationship between requirements .....	43
2.3.2 Fuzzy logic to manage the software development process .....	46
2.3.3 Fuzzy functional and non-functional requirements, and design artifacts including design alternatives.....	53
2.4 Ambiguity identification and handling .....	58
2.4.1 Understanding ambiguity in requirements.....	58
2.4.2 Pattern-driven inspection techniques to identify ambiguities .....	61
2.4.3 Software linguistic tools to support ambiguity identification .....	62
2.4.4 Nocuous and innocuous ambiguity .....	64
2.5 Conflict identification and handling .....	65
2.5.1 Non-Functional Requirements Framework (NFRF) .....	66
2.5.2 Viewpoint-based approaches.....	68
2.5.3 Detecting conflicts in aspect oriented textual requirements.....	74
2.6 Analysis .....	76
2.6.1 Artefacts supported.....	76
2.6.2 How imperfection is supported, what for and with what user interaction....	77
2.7 Research agenda.....	80



3 Information visualization in software engineering.....	83
3.1 Introduction .....	83
3.2 Text, sketches and its presentations .....	84
3.2.1 Table-based and chart-based visualizations .....	85
3.2.2 Hypertext and hypermedia.....	86
3.2.3 Textual presentation and annotations .....	87
3.2.4 Sketches and scenarios.....	88
3.3 Diagrams .....	89
3.3.1 Graphs and trees.....	93
3.4 Visual metaphors .....	94
3.5 Visualizing characteristics of information in software engineering .....	97
3.6 Research agenda.....	98
4 Proposed solution.....	101
4.1 Introduction .....	101
4.2 Approach .....	102
4.3 Identification of the most problematic ambiguities and conflicts.....	105
4.3.1 Ambiguity identification .....	105
4.3.2 Conflict identification.....	111
4.3.3 Selection of ambiguities and conflicts to be discussed in stakeholder consultations .....	116
4.4 Jigsaw puzzle metaphor.....	121
4.4.1 Description .....	121
4.4.2 Rationale .....	124

4.5 Jigsaw puzzle metaphor in action.....	126
4.5.1 Description .....	126
4.5.2 Rationale .....	128
4.6 Synthesis of the approach.....	130
4.7 Resolution of ambiguity and conflict .....	135
4.7.1 Resolution of ambiguity in natural language SRS .....	136
4.7.2 Resolution of conflict in requirements .....	138
4.8 Conclusion .....	141
5 Evaluation.....	143
5.1 Introduction .....	143
5.2 Goals .....	143
5.3 Research methodology .....	144
5.3.1 General design of the experiments .....	145
5.4 Experiments description.....	146
5.4.1 Experiment 1 .....	147
5.4.2 Experiment 2 .....	155
5.4.3 Experiment 3 .....	170
5.4.4 Synthesis of discussions about possible developments .....	180
5.5 Analysis and discussion .....	187
5.5.1 Increase effectiveness when compared with text presentation.....	188
5.5.2 Improve co-authoring of requirements and co-responsibility in ambiguity and conflict handling and Promote team cooperation and creativity.....	194
6 Conclusion and future work .....	197

6.1 Summary and contributions.....	197
6.2 Aims and objectives revisited .....	198
6.3 Future work.....	203
6.3.1 Realisation of the approach .....	203
6.3.2 Jigsaw puzzle supported in a digital media .....	205
6.3.3 How much information and how much perfection?.....	207
6.3.4 Other uses for the jigsaw puzzle metaphor .....	208
6.4 Final remarks .....	209
Appendix A Crisis Management Systems example.....	211
A.1 Text used to introduce the jigsaw puzzle set.....	211
A.2 Complete text of the non-functional requirements .....	211
A.3 Text used in experiment 2.....	215
A.4 Text used in experiment 3.....	216
Appendix B Health Watcher example .....	218
B.1 Text used to introduce the jigsaw puzzle set.....	218
B.2 Text used in experiment 2.....	218
B.3 Text used in experiment 3.....	219
Appendix C CAS example .....	222
C.1 Text used to introduce the jigsaw puzzle set.....	222
C.2 Text used in experiment 2 .....	223
Appendix D Jig3P tool prototype.....	226
Appendix E Research protocol form .....	243
Appendix F Research protocol - consent form .....	247

Appendix G Survey about preferences for pieces background and more data from experiment 2.....	249
G.1 Set up.....	249
G.2 Analysis of results .....	251
G.3 Threats to validity .....	252
Appendix H Lists of ambiguities and conflicts to discuss in each experiment.....	253
H.1 Experiment 1 .....	253
H.1.1 Imperfections with no visual cue but detected by participants .....	253
H.2 Experiment 2 .....	254
H.2.1 Conflicts with visual cue .....	254
H.2.2 Session 1 – Imperfections with no visual cues but detected by participants .....	256
H.2.3 Session 2 – Imperfections with no visual cue but detected by participants .....	257
H.2.4 Session 3 – Imperfections with no visual cues but detected by participants .....	259
H.3 Experiment 3 .....	260
H.3.1 Conflicts with visual cues .....	260
H.3.2 Session 1 – Imperfections with no cues but detected by participants .....	261
H.3.3 Session 2 – Imperfections with no cues but detected by participants .....	263
References.....	265

# List of figures

Figure 1-1 The text of the Real-time requirement as published [Kienzle09]	21
Figure 2-1 A fuzzy requirement and its alternative interpretations [Noppen07]	55
Figure 2-2 A design tree showing a design decision with design alternatives [Noppen07]	56
Figure 3-1 A screenshot of a synergy map [Eppler04]	91
Figure 3-2 A screenshot of a Parameter Ruler session [Eppler04]	92
Figure 3-3 House Metaphor showing a misshaped, and a well-shaped glyph [Bocuzzo07]	96
Figure 4-1 Diagram with the three high level activities of the approach	103
Figure 4-2 The text for Reliability as it is published [Kienzle09]	103
Figure 4-2 (cont.) The text for Availability, Accuracy, and Real-time as it is published [Kienzle09]	104
Figure 4-3 The jigsaw puzzle for the requirements Availability, Reliability, Real-time, and Accuracy of the CMS case study [Kienzle09]	122
Figure 4-4 Illustration of the realisation for the high level activities of the approach	131
Figure 4-5 Illustration of the synthesis of the approach, showing its iterative nature	132
Figure 5-1 The cardboard jigsaw puzzle for part of CMS, after being used in the first experiment	150
Figure 5-2 The cardboard jigsaw puzzle for part of CMS, used in experiment 2	157
Figure 5-3 The K-line jigsaw puzzle for part of Health-Watcher, used in experiment 2	158
Figure 5-4 The K-line jigsaw puzzle for part of CAS example, used in experiment 2	159
Figure 5-5 The jigsaw puzzle for CMS example, used in experiment 3	173
Figure 5-6 The jigsaw puzzle for the Health-Watcher example, used in experiment 3	173

Figure D-1 Picture of the output of Jig3P, for the requirements Availability, Reliability, Real-time, and Accuracy of CMS	227
Figure G-1 The jigsaw puzzle for CMS example, with picture A (“clouds and water”)	249
Figure G-2 The jigsaw puzzle for CMS example, with picture B (“desert”)	250
Figure G-3 The jigsaw puzzle for CMS example, with picture C (“red, green, blue gradient”)	250
Figure G-4 The jigsaw puzzle for CMS example, with picture D (“orange, yellow abstract”)	251

# List of tables

Table 1-1 Relation between ambiguity and conflict with its causes	27
Table 2-1 Artefacts supported by the approaches studied	76
Table 2-2 SE activities, types of support, technique used and user-interaction provided by the approaches studied	77
Table 2-2 (cont.) SE activities, types of support, technique used and user-interaction provided by the approaches studied	78
Table 5-1 Conflicts and ambiguities with a visual cue in the jigsaw puzzle for the CMS system in experiment 1	149
Table 5-2 Systems, type of presentation, and number of participants in each session of experiment 2	156
Table 5-3 Number of imperfection with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 1	161
Table 5-4 Number of imperfection with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 2	162
Table 5-5 Number of imperfection with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 3	162
Table 5-6 Systems, type of presentation and order of usage in each session of experiment 3	171
Table 5-7 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 3, session 1	174
Table 5-8 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 3, session 2	174
Table G-1 Table showing the number of points each picture obtained for participants preference	252
Table H-1 Conflicts with a visual cue in the jigsaw puzzle for the CMS system in experiment 2	254
Table H-2 Conflicts with a cue in the jigsaw puzzle for the CMS system in experiment 3	261

# **1 Introduction**

## **1.1 Introduction**

Effective requirements engineering in the presence of imperfection remains a major research problem. The vast majority of requirements documentation is described using natural language [Mich04, Mich04a]. Natural language is inherently imperfect [Meyer85]. Thus, requirements are pieces of information where imperfection, such as incompleteness, ambiguity, and conflict are inherently present [Davis89].

The focus of this thesis is on identifying conflict and ambiguity, inadvertently introduced during the requirements engineering process. Another key focus is to provide a metaphor to make the identified conflicts and ambiguities explicit during meetings with stakeholders, in order to support handling of these imperfections.

This chapter introduces the problem of the presence of ambiguity and conflict in requirements engineering, defines the scope of the thesis and briefly describes the state of the art in relevant research areas. The chapter then presents the aims and objectives of the thesis. It introduces the proposed approach to make the presence of conflict and ambiguity in requirements explicit, in order to facilitate communication amongst requirements engineers and stakeholders, and find a resolution (when resolution is possible and desirable). The chapter discusses the novel aspects of the thesis and finishes with an outline of the thesis contents.



## 1.2 Problem

### 1.2.1 Ambiguity and conflict in software requirements

The Guide to the Software Engineering Body of Knowledge (SWEBOK) states that “a software requirement is a property which must be exhibited by software developed or adapted to solve a particular problem” [IEEE-CS04]. Requirements are often classified into functional requirements (also known as capabilities) and non-functional requirements (also known as constraints or quality requirements). Also following the SWEBOK, “functional requirements describe the functions that the software is to execute”, while “non-functional requirements are the ones that act to constrain the solution” [IEEE-CS04].

This thesis adopts the definitions for the characteristics of a good software requirements specification (SRS) from the “IEEE Recommended Practice for Software Requirements Specifications” [IEEE-SA98]. According to IEEE Std. 830-1998 [IEEE-SA98] an SRS should be: 1) correct; 2) unambiguous; 3) complete; 4) consistent (no requirements in conflict); 5) ranked for importance and/or stability; 6) verifiable; 7) modifiable; and 8) traceable.

The focus of this thesis is on addressing ambiguity and inconsistency (conflict) in non-functional requirements. Berry et al [Berry03], citing a tutorial by Donald Gause, note the following two key sources of requirements failure:

- “failure to effectively manage conflict, and
- too much unrecognized disambiguation.”

Unrecognized disambiguation is defined “as that process by which a reader, totally oblivious to other meanings of some text that he has read, understands the first

meaning that comes to mind and takes it as the only meaning of text.” And that “unconscientiously assumed meaning may be entirely wrong” [Berry03].

Lamsweerde [Lamsweerde00] in his research perspective of RE in the year 2000, when explaining why the process of engineering requirements is so complex points out among others that:

- “There are multiple concerns to be addressed beside functional ones – e.g., safety, security, usability ... These non-functional concerns are often conflicting.”

Non-functional requirements (NFR) very often present ambiguity and conflict, requiring clarification and trade-offs, which should be discussed with stakeholders [Sommerville97, Lamsweerde00, Marhold09]. Non-functional requirements can be decisive in the choice of architecture, and thus have strong implications in terms of cost and achievement of business goals [Bass03]. Experience also shows that user non acceptance of a system originates more often from inadequate non-functional requirements, than from problems with functional requirements. Among the causes of user non-satisfaction related with NFR are ambiguity, and conflict [Marhold09]. While functional requirements may also present ambiguity and conflict, handling these types of imperfections in non-functional requirements is a major challenge in itself.

Hence the thesis focuses on addressing these issues for non-functional requirements only<sup>1</sup>; handling of ambiguity and conflict in functional requirements is left for future work. This research addresses conflicting requirements expressions, and not conflicts due to different interests of stakeholders and/or software engineers.

The ultimate reason to address imperfection, and in particular ambiguity and conflict, in requirements is that if ambiguity and conflict are not made explicit and

---

<sup>1</sup> From this point on the thesis will use the term requirements to mean non-functional requirements, in particular.

appropriately handled, it may be presumed that the requirements are unambiguous and consistent.

If and when requirements are presumed unambiguous and consistent (and in general possessing all the good characteristics they should), development will proceed and the system produced will contain errors or unwanted features. Big disasters, even costing human lives, due to poor quality of requirements specifications, are well known in systems engineering history (e.g., Therac-25 [Leveson95], Mars Climate Orbiter [Stephenson99]). It is also known that the sooner these errors are corrected the less costly they are [Fagan76]. In fact, it is crucial for the quality of software systems, that both information and its quality (or lack of it), are represented in a way that enables and promotes communication between the different actors involved in software development. Examples of such actors include system owners, end users, developers, etc.

When ambiguous and/or conflicting requirements are identified, one may defer the decisions concerning those requirements, in order to consider the alternatives raised by ambiguity or conflict (which can be advantageous [Balzer91, Goguen94, and Marcelloni01]). Alternatively, one may undertake additional work (elicitation and analysis) to try to resolve these imperfections. The focus of this thesis is on the latter, that is, resolution of conflict and ambiguity in requirements where possible. This is done using an innovative communication mechanism for stakeholders meetings where the inherent ambiguity and conflict of human interpretation of elicited requirements is dissected seeking clarification and resolution, where possible.

## 1.2.2 Characteristics of requirements engineering

Requirements engineering (RE) starts with the expression by stakeholders of business goals, ideas and needs using natural language, which is inherently ambiguous and inconsistent. These goals, ideas, and needs are interpreted by software engineers, to develop a system aiming to serve those business goals and needs. This communication process between stakeholders and requirement engineers is central, and thus critical to RE.

Requirements engineering possesses a unique context within systems development, with some specific characteristics and needs. Some of these are as follows:

1. It is an analytical task, and also a creative task, i.e., it is a task where often participants are implicitly building something (a system, a part of it) entirely new, out of some pieces [Goguen92, Maiden04, Maiden05, Maiden07, and Robertson02].
2. It is performed by a heterogeneous community of software engineers and stakeholders with different interests, as well as a common interest, which is to elicit requirements for the system to be built. In fact, there is a co-responsibility of co-authoring the requirements documentation, coordinated by a requirements engineer. The balanced and moderated inclusion of the different points of view and interests is crucial for the success of the system being produced [Sommerville97, Sim08].
3. The different software engineers involved and other stakeholders have usually different backgrounds, which are not necessarily computing, engineering or mathematical backgrounds [Sommerville97, Lamsweerde00]. It is known that a lack of specialist notation improves communication in such multi-disciplinary contexts [Millard98].

4. There may be misunderstandings, and conflicts, which pitch people against each other. These misunderstandings and conflicts tend to transform working meetings into unpleasant and boring ones [Sommerville97].

5. In consultation meetings with stakeholders, typically, requirements engineers want to focus on a small number of requirements (say 4 to 6) [Sommerville97].

To ensure that the concomitant analytical and creative nature of RE is enabled (point 1 above), while enabling team work by a heterogeneous community of stakeholders (point 2) and avoiding misunderstandings and conflicts (point 4), it is important to promote cooperation, and improve communication. As such it is essential to provide a communication mechanism free of specialist notation (point 3). Games are known to induce team cooperation and creativity (point 1). Games help to “break the initial ice” and induce a fun and relaxed environment [Maiden07, Maiden08].

These characteristics and needs form part of the specific context of RE, and form the bedrock of the approach proposed in this thesis.

The definitions and causes of ambiguity and conflict are presented next before moving on to describing the proposed approach.

### **1.2.3 Definitions of ambiguity and conflict in requirements**

This sub-section describes relevant definitions commonly used in software engineering literature. In order to better understand these definitions, an example of requirements documentation is introduced. Figure 1-1 shows the Real-time requirement text for a Crisis Management System (CMS) [Kienzle09].

A. Real-time

1. The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.
2. The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
3. The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.

**Figure 1-1 The text of the Real-time requirement as published [Kienzle09].**

The complete requirements documentation of the Crisis Management System Case Study is available in [Kienzle09]. Appendix A reproduces the complete set of non-functional requirements for ease of access for the reader.

The characteristic of a software requirements specification (SRS) of being unambiguous, according to IEEE Std. 830-1998, is defined as follows:

**Unambiguous**

«An SRS is **unambiguous** if, and only if, every requirement stated therein has only one interpretation» [IEEE-SA98].

Meyer proposes a list of the seven sins of the specifier [Meyer85]. One of them is ambiguity defined as “the presence in the text of an element that makes it possible to interpret a feature of the problem in at least two different ways” [Meyer85].

Berry et al point out that the IEEE Std. 830-1998 definition is problematic: “there is no unambiguous specification simply because for any specification, there is always someone who understands it differently from someone else” [Berry03]. They conclude that there is no single comprehensive definition of ambiguity in software engineering literature. Instead there are some definitions highlighting different aspects, complementing each other [Berry03]. Berry et al present some of these definitions:

- Davis test for ambiguity: “Imagine a sentence that is extracted from an SRS, given to ten people who are asked for an interpretation. If there is more than one interpretation, then the sentence is probably ambiguous” [Davis93]. Of course the opposite (ten persons giving the same interpretation) is not a “complete” guarantee that the sentence is unambiguous. Instead, this can be seen as a practical test to assess if the SRS is unambiguous for most practical purposes.
- Schneider, Martin, and Tsai definition: “An important term, phrase, or sentence essential to an understanding of system behaviour has either been left undefined or defined in a way that can cause confusion and misunderstanding. Note that these are not merely language ambiguities such as an uncertain pronoun reference, but ambiguities about the actual system and its behaviour” [Schneider92].

The definition by Schneider et al [Schneider92] points out two pertinent aspects of ambiguity. One aspect has to do with the causes of ambiguity, namely human error or missing (or insufficient) information. These causes of ambiguity are discussed in the next section. Another aspect is that ambiguity may not only be due to language

ambiguity (e.g. uncertain pronoun reference) but due to different interpretations of the domain knowledge. It is clear that the interpretation of a description is very much a function of the reader's background [Berry03].

Example 1 below illustrates the presence of ambiguity in the Real-time requirement of the Crisis Management System SRS.

*Example 1 – Ambiguity: “are control centre and system the same or different entities?”*

The second phrase of the “Real-time” requirement states that: “the delay in communication of information between control centre and rescue personnel...shall not exceed 500 milliseconds”; while the third phrase prescribes that “the system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.”

It is not clear if the expressions “control centre”, and “system” refer to the same entity. There are at least two possible interpretations: one that “control centre” and “system” are the same entity; and the other that they are different entities.

## **Consistency**

According to IEEE Std. 830-1998, consistency is about agreement inside the SRS itself, and is defined as follows:

«**Consistency** refers to internal consistency. If an SRS does not agree with some higher-level document, such as a system requirements specification, then it is not correct<sup>2</sup>.

---

<sup>2</sup> ‘Correct’ is another characteristic of a good SRS; a definition of ‘correct’ can be found in [IEEE-SA98].



An SRS is **internally consistent** if, and only if, no subset of individual requirements described in it **conflict**<sup>3</sup>» [IEEE-SA98].

The Merriam Webster English Dictionary [Merriam-Webster12] defines conflict (the second entry) as

2.
  - a. “competitive or opposing action of incompatibles : antagonistic state or action (as of divergent ideas, interests, or persons)
  - b. mental struggle resulting from incompatible or opposing needs, drives, wishes, or external or internal demands”.

Meyer defines contradiction (another of the seven sins of the specifier) as “the presence in the text of two or more elements that define a feature of the system in an incompatible way” [Meyer85]. Instead of the word ‘contradiction’ Meyer should have used the word ‘inconsistency’. In fact using the expression ‘incompatible way’, the concept he defines is more general than contradiction (contradiction is one possible form of incompatibility).

Using the IEEE Std. 830-1998 and Meyer definitions above as a basis, this thesis defines that:

An SRS is **internally consistent** if, and only if, no subset of individual requirements described in it define a feature of the system in an incompatible way.

Example 2 illustrates some conflicts in the Real-time requirement of the Crisis Management System SRS.

---

<sup>3</sup> In the remainder of the thesis, we will use the word ‘conflict’ to mean lack of internal consistency.

Example 2 – Conflict: “Real-time time description requiring times possibly incompatible.”

The extracts of the “Real-time” requirement under consideration are:

1. “The control centre shall receive and update ... information ... at intervals not exceeding 30 seconds.
2. The delay in communication of information between control centre and rescue personnel ... shall not exceed 500 milliseconds.
3. The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.”

One pertinent question this text raises is: “Does communication of information (in the second phrase) require its retrieval (in the third phrase)?” This is an ambiguity issue.

In the case that communication of information does require its retrieval, it is relevant to interrogate: “Are the values for maximum delay in these requirements compatible?”

Another key issue is to know if the maximum delays of 500 milliseconds allowed for “the delay in communication of information” (second phrase) and the ability to “retrieve any stored information” (third phrase) are already accounted for and compatible with the first phrase that demands “receive and update ... information ... at intervals not exceeding 30 seconds”.

It is interesting to note that the conflicts can be related with ambiguity, as it is the case in example 2.


## 1.2.4 Causes of ambiguity and conflict in requirements

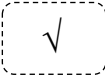
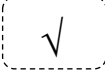
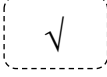
Ambiguity and conflict may be present in requirements due to unavailability of sufficient information. In RE sufficient information may not be available at the time the requirements are written, and may or may not become available later [Marcelloni01, Berry03, Noppen07a, Noppen08].

Requirements may also contain ambiguity, and conflict caused by human error, such as an omission or oversight on the part of the person(s) who wrote them. Requirements writers may also introduce information that is already ambiguous or in conflict. In fact, the process of interpretation by the requirements writers may produce errors, simply because their interpretation may be wrong [Sommerville97a, Berry03, Noppen07a, Noppen08].

Another reason for the presence of ambiguity and conflict in requirements is associated with systems whose requirements are more likely (compared to other systems) to enter in conflict with other requirements. These systems have a higher tendency for conflict, because they are very stringent with requirements (usually NFRs) which may be difficult to comply with at the same time with high levels of exigency [Sommerville97, Chung00]. One class of this type of systems is critical systems which need to have high levels of security, maintaining high speed of response as well as high level of accuracy. These are systems whose high levels of exigency are difficult to comply with for two or more requirements at the same time, like security and accuracy, or security and mobility.

This thesis focuses on conflict and ambiguity in requirements due to human error. The thesis is also particularly concerned with conflicts in systems with higher tendency for conflicts between requirements. The following table (Table 1-1) relates

ambiguity and conflict with their causes. The situations representing the thesis focus are marked in the table with .

Cause \ Types	Unavailability of sufficient info.	Human error	System with tendency for conflict
Ambiguity	√	 √	
Conflict	√	 √	 √

**Table 1-1 Relation between ambiguity and conflict with its causes.**

The thesis does not consider conflict or ambiguity from the temporal point of view, nor does it handle the fact that some information that is not sufficient now may or may not be sufficient in the future.

### **1.3 Scope of the thesis**

The focus of this thesis is on making conflict explicit in non-functional requirements, so that communication amongst requirements engineers and stakeholders is facilitated, seeking a resolution of conflict and associated ambiguity. Specifically, the thesis focuses on communication during stakeholder consultation meetings in this regard. The proposed approach offers an easy to use visual metaphor, free of

specialist modelling notation to facilitate dialogue across a heterogeneous community of stakeholders and engineers.

The thesis scope is circumscribed to conflict and ambiguity due to human error, i.e., that might be inadvertently introduced by the requirements engineer during requirements capture or authoring. The thesis scope includes conflict among requirements of systems which are particularly stringent with pairs of requirements difficult to comply with at the same time (i.e. that have higher tendency for conflict).

The next section briefly summarises the gaps in the state-of-the-art in this regard.

## **1.4 Limitations of state-of-the-art**

### **1.4.1 Imperfect information support through modelling languages**

Some methods have been developed, to provide decision support for specific software activities that are hampered by imperfect information. These methods extend the expressive capabilities of the software development process. This is achieved by adding models, which describe important properties of imperfect information, for instance by means of probability theory and/or fuzzy logic [Yen93, Lee03, Liu96, Marcelloni01, Akşit01, Marcelloni04, Marcelloni04a, Tekinerdoğan03, and Noppen07].

The focus of these approaches is to model the imperfect information in the software development process covering aspects such as the evaluation of the quality of the knowledge sources [Tekinerdoğan03], the vagueness in requirements and the degree to which requirements conflict (or not) [Yen93, Lee03, Liu96]. More recent works [Marcelloni01, Akşit01, Marcelloni04, Marcelloni04a, Tekinerdoğan03, and Noppen07] extend this effort to support imperfect information to the decisions taken during the development process aiming to cover the entire life cycle.

In one hand it is arguable that, these approaches are designed to provide support for software engineers mainly in activities where the input of stakeholders with no SE background is not intended. But in the other hand it is true that even for software engineers it is not easy to interact with formalisms using fuzzy logics. Also if and when stakeholders' input is needed or critical, these formalisms can't be used directly by them.

Noppen [Noppen07] proposes tools with graph visualizations to support the software engineer dealing with fuzzy logics and probability concepts. It is a progress but it is still user interaction with mathematical models, which is not accessible for all.

This thesis proposes a separation of the processing of information about imperfection in requirements from the issue of communicating that imperfection. This separation is crucial to permit a user-centred design of the communication of imperfection in requirements.

Although this separation is proposed (between processing and communicating information on imperfection in RE) the communication mechanism can and should be used as a front end of the processing of information on imperfections. In particular, the jigsaw puzzle metaphor proposed in this thesis, to make conflict explicit in stakeholder meetings, can be used in complement with these approaches that use probability theory and/or fuzzy logic to model imperfection in RE. In fact the fuzzy

description relations between requirements can be translated into visual cues in the jigsaw puzzle pieces, such as the interlocking shape between two pieces being bigger or smaller according to the fuzzy description. This type of visual cue would help visualize concepts like the “size” of a conflict.

### **1.4.2 Ambiguity and conflict identification and handling**

Berry et al [Berry03] organise the techniques to reduce the level of ambiguity in requirements during requirement validation as follows:

1. Formalization of informal requirements: for example [Yen93, Lee03, Liu96, Marcelloni01, Akşit01, Marcelloni04, Marcelloni04a, Tekinerdoğan03, and Noppen07].
2. Pattern-driven techniques, such as:
  - a. reading techniques for requirements inspections e.g. [Kamsties01],
  - b. natural language processing tools e.g. [Kiyavitskaya08, Gleich10].
3. Compare the interpretations of a document by different stakeholders: if they differ there is an ambiguity in the original document.
4. Communicate an interpretation back to the requirements author, after which he can easily point out misinterpretations.

According to Berry et al [Berry03] these last two classes of techniques (3 and 4) are probably the most effective strategies for finding ambiguities in SRS. They are also

the most resource-demanding. Consequently there is a shortage of techniques in these two classes.

This thesis proposal addresses this gap by promoting communication among requirement engineers and stakeholders to enable identification and discussion about imperfection. In particular, this thesis shows that the use of the jigsaw puzzle metaphor is an effective strategy for conflict and ambiguity identification.

Chantree et al [Chantree06] and Yang et al [Yang11] developed approaches that refine the ambiguity cases presented, through a separation of what is considered nocuous<sup>4</sup> from innocuous ambiguity.

Conflict identification has been recognized as a RE problem for many years. Viewpoints and Inconsistency Management (VIM) [Finkelstein92, Nuseibeh94, and Easterbrook96] and the Non-Functional Requirements Framework (NFR) [Mylopoulos92, Chung00] provide specification formalisms that explicitly represent and manage relationships between the artefacts (ViewPoints, Softgoals respectively) that represent the knowledge about the system to be built. However, one needs to be knowledgeable in the formalism (and these are quite complex) to be able to manage the conflict relationships.

In Preview [Sommerville97a] the requirements belonging to viewpoints whose foci intersect are classified into three categories: overlapping, conflicting, independent. Using Preview it is possible to build a list of the most problematic requirements. The conflict detection tools [Weston09, Sardinha09, and Sardinha10] developed under the umbrella of aspect-oriented textual requirements approaches that use semantics-based decomposition also present the output as a list of possible conflict situations. These tools, however, do not address the communication goals (in particular with non-software engineer stakeholders), which are the focus of this thesis.

---

<sup>4</sup> Nocuous ambiguities are the ones that lead to misunderstanding, while innocuous ambiguities are the ones that have only one obvious interpretation [Chantree06].



The approach proposed in this thesis aims to use the outputs of ambiguity and conflict detection tools as input to a communication mechanism accessible to all kinds of stakeholders, thus promoting their participation and aiming at co-responsibility in requirements.

### **1.4.3 Visual communication metaphors in requirements engineering**

Gotel et al [Gotel08] envisioned the potential synergy between Information Visualization and SE Visualization to provide good metaphors for software development tools, enabling effective communication in software development. In fact, visual metaphors<sup>5</sup> have, for a long time, been used to represent information in Software Engineering. In the last fifteen years the predominant visualizations in the area of requirements engineering are either associated with UML diagrams [OMG\_UML] or i\* goal models [Yu97, Gotel08]. Field studies [Bresciani08] confirm that the visualization's efficacy in diagrams is dependent on the user's previous experience and visual literacy, which can present a barrier for communication and work among multidisciplinary user profiles, and thus between requirements engineers and stakeholders.

Some more sophisticated visual metaphors have been recently developed to visualize software [Knight00, Charters02, Panas03, Balzer04, Wettel07a, Wettel07b], but with different goals compared to this thesis. In fact, the more complex visual

---

<sup>5</sup> A visual metaphor is an analogy that underlies a graphical representation of an abstract entity or concept with the goal of transferring properties from the domain of the graphical representation to that of the abstract entity or concept [Diehl07].

metaphors (more complex than two- or three-dimensional geometric ones) have been applied predominantly to artefacts and software that already exist (to show metrics, for program comprehension, for reverse engineering, etc.) [Diehl07]. The work of Boccuzzo et al [Boccuzzo07] describes an interesting usage of the concept of well-shaped graphical visualization to represent that the corresponding artefact is well designed. Their work shows that the metaphor “language” can be used to express information (quality aspects) about the artefacts that are being represented.

This thesis foresees that a well-assembled metaphor, making an analogy with a task where participants build something out of some pieces, can provide interesting tools for RE communication processes. In order to accommodate usage by different professional profiles that cooperate in RE, the visual metaphor should build on a well-known concept to be easily used by the broadest set of professional profiles. Such a metaphor can effectively increase communication and cooperation amongst engineers and stakeholders during requirements engineering.

#### **1.4.4 Open challenges in handling of ambiguity and conflict in RE**

The discussion in sections 1.4.1 to 1.4.3 highlights the following open research challenges:

- There is a need for approaches that handle the problem of imperfection in requirements through a separation of the processing of the information about the imperfection from the issue of communicating that imperfection, i.e. instead of “coding” imperfection in a formalism. Technical formalisms are not easily understandable by the heterogeneous community of stakeholders,

making communication about imperfection less efficient. Such a separation will enable communication mechanisms accessible to a variety of stakeholders, thus promoting their participation in the RE activities where their involvement may be crucial for the quality of the SRS.

- There is a need for communication mechanisms to adequately raise awareness of conflict and ambiguity in requirements, improving communication and cooperation amongst engineers and stakeholders during consultation meetings.

## **1.5 Aim and objectives**

The aim of the research, described in this thesis, is to improve the identification, communication, and handling of ambiguity and conflict in requirements, inadvertently introduced during the RE process.

The approach presented proposes a visual metaphor to make the identified conflicts and ambiguities explicit, during meetings with stakeholders organised for handling these imperfections. The following objectives are pursued:

1. Develop guidelines/heuristics to identify the ambiguities and conflicts in requirements that are most pertinent to be discussed during stakeholder consultation meetings.
2. Develop an effective communication mechanism that makes the ambiguity and conflicts in key NFRs explicit. In order to be effective, the communication

mechanism must consider the task at hand in the consultation meetings as an analytical/creative task, performed by a group of heterogeneous stakeholders.

3. Develop and investigate a method for conducting the consultation meetings with the stakeholders, using the communication mechanism to promote cooperation and co-responsibility towards the requirements document. The method should make the analysis and search for conflict and ambiguity resolution an enriching experience, through a fun and relaxed environment.

## **1.6 Proposed approach**

This thesis proposes an approach to communicate and handle ambiguity and conflict in requirements engineering, through a jigsaw puzzle metaphor to be used in stakeholder consultation meetings, aiming at:

- Increasing effectiveness when compared with text presentation.
- Fostering team work and communication, and improving commitment of stakeholders in co-authoring of requirements and co-responsibility in ambiguity and conflict handling.
- Promoting a relaxed environment to improve team cooperation and creativity.

The key focus of the thesis is on identifying conflict and ambiguity, inadvertently introduced during the requirements engineering process. The proposed approach provides heuristics and tools to identify the most pertinent conflicts and ambiguities,

to be discussed in a consultation meeting with stakeholders, and thus relevant to be made visually explicit.

Another key focus is the jigsaw puzzle metaphor aimed as an effective communication mechanism to make the identified conflicts and ambiguities explicit, and promote discussion about their resolution or trade-offs. Being a commonly known, and easy to learn game, the jigsaw puzzle metaphor provides an easy to comprehend metaphor for the heterogeneous community of stakeholders.

As the method for conducting the consultation meetings with the stakeholders uses a metaphor with such a playful nature, it inherently promotes creativity and encourages team work and communication.

## **1.7 Novel elements of the thesis work**

The overall contribution of this thesis is its focus on separating the processing of the information about the imperfection from the issue of communicating that imperfection. Such a separation, though critical, has not been proposed to date.

The jigsaw puzzle metaphor is a novel contribution in the area of visual metaphors, and in particular as a communication mechanism used to make conflict and ambiguity explicit during stakeholder consultation meetings. The novelty of the jigsaw puzzle metaphor resides in:

- being an easy to understand and learn language instead of a technical formalism
- the analogy with misshapen graphical visualization to represent that there is a problem;
- its adequacy to a creative task as RE is; and
- its gaming nature.

The metaphor is even easily understood by a person who has never played with a jigsaw puzzle before.

The novelty of the jigsaw puzzle metaphor, as a software visualization technique, resides in its use for a creative task. When working with the jigsaw puzzle metaphor participants are implicitly building something (a system, a part of it). In a jigsaw puzzle game users build something created from different pieces that have to be correctly assembled to yield a final “perfect” object.

Another advantage and novelty of the jigsaw puzzle metaphor as a communication mechanism is its gaming nature. Games are often used in the beginning of meetings, and in particular of elicitation meetings [Maiden07, Maiden08] to induce a fun and relaxed environment, and promote creativity. In this approach there is no need to introduce a game in the meeting; the meeting itself is a game, a jigsaw puzzle, which is at the same time the communication mechanism.

## 1.8 Outline of the thesis

Chapter 2 is dedicated to an overview of the main requirements engineering approaches that address the issue of imperfection in requirements documentation, and in particular ambiguity and conflict identification and handling. It discusses also research on creativity as an important ingredient for requirements elicitation.

Chapter 3 describes information visualization techniques and considers its usefulness, as a communication mechanism, to make explicit and enable the handling of ambiguity and conflict in requirements engineering, considering both the RE characteristics, as well as, its users: stakeholders and engineers.

Chapter 4 describes an approach to the handling of ambiguity and conflict in requirements engineering. The approach is presented and then the three challenges addressed are exposed with more detail. These challenges comprise the identification of the most pertinent ambiguities and conflicts, the development of a communication mechanism - the jigsaw puzzle metaphor- for the requirements and its ambiguities and conflicts, and the development of a method of work with the jigsaw puzzle metaphor that is effective in consultation meetings with stakeholders. Chapter 4 includes a synthesis of the approach, and an overview of ambiguity and conflict resolution approaches, and ends with a conclusion.

Chapter 5 presents the evaluation used in this research. After explaining the goals and research methodology chosen, the performed experiments are described. The chapter presents the threats to validity, the set-up and analysis of the results for the experiments. It includes a synthesis of the conclusions from the discussions about possible developments. Chapter 5 closes with an analysis of the results and a discussion of the results in the overall context of the thesis.

Finally, Chapter 6 presents the conclusions and describes the future work.



## **2 Requirements engineering and imperfect information support**

### **2.1 Introduction**

The chapter opens with a brief description of what is elicitation of requirements and describes some creative techniques used in this context.

This chapter then describes some approaches to support imperfect information in requirements engineering and the software development process, focusing on the extension of the modelling language, with formalisms to specify the imperfections. These approaches cannot alone solve the problem addressed in this thesis, as these languages are too technical to be used in meetings with stakeholders.

The chapter also surveys tools where the focus lies in detecting and managing imperfections, namely ambiguity and conflict. This thesis aims to use the output from these ambiguity and conflict tools as input to communication mechanisms accessible to all kinds of stakeholders.

The chapter ends with an analysis of the works surveyed and a research agenda.

## 2.2 Requirements elicitation and creativity

As Nuseibeh and Easterbrook [Nuseibeh00] explain, “elicitation is perhaps the activity most often regarded as the first step in the requirements engineering process”. But information collected during requirements elicitation has to be interpreted, analysed, communicated, and validated. These activities are interconnected as requirements engineering is an iterative process.

Already in 2000, Nuseibeh and Easterbrook [Nuseibeh00] reported a vast list of elicitation techniques.

Although brainstorming, one of the most popular creativity techniques used for requirements identification, dates back to 1935, the use of creativity techniques in requirements engineering is still under investigated [Mich04b].

This thesis shares the view of Maiden et al [Maiden05, Maiden07] that “requirements engineering is a creative process in which stakeholders work together to create ideas for new software systems that are eventually expressed as requirements.”

Maiden and his research group developed a requirements engineering process, RESCUE [Maiden07], which incorporates creativity workshops to foster creative thinking to discover requirements. Maiden et al [Maiden05] use the cognitive psychology definition of creativity as: “the ability to produce work that is both novel (i.e. original, unexpected) and appropriate (i.e. useful, adaptive concerning task and constraints)”.

These creativity workshops use several techniques to encourage creativity, including brainstorming and analogical reasoning. Brainstorming is a technique where “the requirements engineer asks a group of stakeholders to generate as many ideas as possible, with emphasis on generation rather than on evaluation; it is a good

technique for eliciting high-level domain entities and questioning assumptions which might otherwise have constrained approaches considered” [Maiden96]. Analogical reasoning is defined “as an inference process in which a similarity between a source and a target is inferred from the presence of known similarities, thereby providing new information about the target when that information is known about the source” [Russel96].

From the experiments reported by Maiden and colleagues [Maiden04, Maiden05, and Maiden07], they draw some conclusions that are useful to note and discuss:

1. Workshop participants found it difficult to exploit the analogies (between air traffic management and textile and music domains) [Maiden04]; in fact studies from cognitive science reveal that analogical reasoning with unfamiliar domain classes is difficult without prior learning. Maiden et al conclude that it is important to explain the analogies to participants with simple examples.

2. Brainstorming generated more creative ideas than analogical reasoning, and it was more cost-effective and easier to use. However the ideas obtained with analogical reasoning were described in more detail than the brainstormed ideas [Maiden05]. In [Maiden05] they stress again that more explanation of analogical mappings might be useful. [Maiden07] presents some strategies to make analogical reasoning more effective.

The difficulties reported by Maiden et al in promoting analogical reasoning as a technique to foster creativity appear to be normal as the analogies used are difficult and costly to explain and learn. Instead this thesis proposes visual metaphors as figures of speech that construct visual analogies between two things or ideas, as a solution that helps overcome the problems reported by Maiden et al [Maiden04, Maiden05, and Maiden07]. This recognises the relevance of defining and developing the requirements engineering process as a creative process, in which engineers and

stakeholders work together to build something new. This is a conviction that underpins this proposal for a visual metaphor that uses an analogy based on building a final solution out of several pieces: the jigsaw puzzle metaphor.

## **2.3 Imperfect information support through modelling languages**

This section describes some of the approaches developed to provide decision support for specific software activities that are hampered by imperfect information. These methods extend the expressive capabilities of the development process. This is achieved by adding models, which describe important properties of imperfect information, for instance by means of probability theory and/or fuzzy logic.

### **2.3.1 Fuzzy logic to specify and analyse imprecise, and vague requirements, and the relationship between requirements**

Yen and Lee [Yen93], and Liu and Yen [Liu96] propose frameworks based on fuzzy logic to model imperfect functional requirements. Yen and Lee [Yen93] define soft functional requirements, as an extension of the notion of soft post-conditions in task-

based specification methodology (TBSM) [Yen93a], to explicitly capture the impreciseness of functional requirements. In TBSM a task is defined as a system functional unit [Yen93a] and the functionality of a task is specified by properties of its state-transition  $\langle b, a \rangle$ , where 'b' is the state before the task and 'a', is the state after invoking the task. The functional requirement of a task can thus be specified using a pair  $\langle \text{precondition}, \text{postcondition} \rangle$ . In the conventional approach to software specification, the precondition and the postcondition describe properties that should be held by states 'b' and 'a', whereas a soft functional requirement describes state properties that can be satisfied to a degree. More specifically, the soft functional requirement is represented using the canonical form in Zadeh's test-score semantics [Zadeh86]. Yen and Lee claim their framework is also applicable to non-functional requirements, although they focus on functional requirements.

Yen and Lee [Yen93] discuss three issues in adopting fuzzy logic as the formal foundation for specifying soft functional requirements:

1. Difficulty in expressing the meaning of terms directly using membership functions. In these cases it is necessary to define the meaning of a term indirectly using other variables that are more easily measurable.
2. It is often desirable to represent requirements with varying degrees of criticality.
3. The meaning of fuzzy predicates/terms is context-dependent. To address this problem Yen and Lee formally define context-dependent membership functions based on Zimmermann's [Zimmermann91] proposal of membership functions whose parameters are determined by the context and thus characterize a context-dependent fuzzy set.

The consideration of these expressiveness and context-dependency issues is highly important when building tools to perform RE tasks and handle imperfection in requirements. It is critical that the meaning of what is being communicated is not lost.

Otherwise the tool to support handling of imperfection may become non-effective; as it might be that the handling mechanism introduces imperfection. Due to the importance of these issues in RE, this thesis approach focus on separating the processing of the information about the imperfection from the issue of communicating that imperfection. This separation enables the communication mechanism to handle the expressiveness and context-dependency issues. Such a separation, though critical, has not been proposed to date.

Lee et al [Lee03] extend [Yen93] and present an integrated approach for acquiring heterogeneous requirements in the elicitation phase, modelling vague requirements as fuzzy object models (soft requirement concept) in the modelling phase, and analysing requirements in the analysis phase. This approach enables the analysis of the trade-off among vague requirements by identifying the relationship between requirements, which could be conflicting, irrelevant, cooperative, counterbalance, or independent. In order to obtain a feasible overall requirement it establishes a requirement hierarchy based on the notion of criticality and degree of cooperation. The aggregation of requirements in a requirement hierarchy is done through: (1) a breadth first search algorithm to traverse the requirements hierarchy and obtain an ordered list, and (2) the combination of the individual requirements using the fuzzy and or fuzzy or operator recursively in the list.

Liu and Yen [Liu96] define imprecise requirements using the concept of elastic criteria. An imprecise requirement imposes an elastic constraint. The universe being constrained by a requirement is called its domain. The constraint imposed by an imprecise requirement is represented as a satisfaction function, that maps the elements of the requirement's domain to a number in the interval  $[0,1]$  representing the degree to which the requirement is satisfied. Based on this definition, the framework provides heuristics to derive relationships between imprecise requirements from relationships already identified and classified. The relationships

can be classified as cooperative, conflicting, mutually exclusive, or irrelevant. These “base” relationships are classified in the first place by experienced requirements analysts.

The approaches described in this Section 2.3 were the first works to make the author of this thesis aware of the importance of making explicit the presence of imperfection in RE and SE, as well as the need to subsequently handle that imperfection. This study permitted to understand the immense value and potential of the usage of modelling languages extended with fuzzy logics and/or probability theory.

In particular [Yen93] called our attention to the importance of expressiveness and context-dependency issues, when expressing and handling imperfection in RE. This was influential in this thesis’ approach focus on separating the processing of the information about the imperfection from the issue of communicating that imperfection, as well as in the choice of a tool (the EA-Analyzer) to select the most pertinent conflicts capable of minimizing the loss of information.

### **2.3.2 Fuzzy logic to manage the software development process**

In the development of software systems, a knowledge source to be useful for solving a problem must have abstraction quality (objective value plus relevance). A simplistic approach to evaluate domain knowledge is to use two-valued logic, whereby a knowledge source either has the abstraction quality or not. But in practice the process of domain analysis is complex and often related to subjective evaluations, vagueness, and uncertainty. Tekinerdoğan and Akşit [Tekinerdoğan03] define the three requirements necessary for a more practical and precise evaluation of the knowledge sources:

1. Expressing the degree of quality. The evaluation of the objective value and the relevance value of knowledge domains are dependent on the background and expertise of the domain engineer. Thus, it may be hard to decide if a given knowledge source completely possesses the quality factors (objective, relevance, and abstract) or not. Instead the knowledge source quality factors should/could be mapped to a number in  $[0, 1]$ .
2. Need for linguistic evaluation of knowledge sources. In fact, the assignment of numbers as suggested by point '1.' is counter intuitive for the domain engineer whom is used to linguistic evaluations, such as fairly, possibly, etc.
3. Providing means to cope with evolution of knowledge and problems. Both evolution of knowledge and evolution of problems, may impact the value of the quality factors of the knowledge sources. This implies that the evaluation of the quality of knowledge should be adaptable to the changing context. Thus, two-valued logic is not adequate as it leads to the absolute acceptance or elimination of the knowledge sources.

To cope with these requirements, Tekinerdoğan and Akşit [Tekinerdoğan03] propose a model and an approach for evaluating domain knowledge using fuzzy logic techniques. They propose a fuzzy knowledge source evaluator (FKSE). The FKSE takes as input the relevance and the objective fuzzy values of a knowledge source and computes the abstraction quality. The inference engine adopts 25 fuzzy heuristic rules. They performed a case-study, which enabled to conclude that applying fuzzy logic techniques in evaluating domain knowledge is of practical use and can support the solution domain analysis process.

Marcelloni and Akşit [Marcelloni01] conclude that current software development methods do not provide adequate means to model inconsistencies, forcing software engineers to resolve inconsistencies whenever they are detected. They argue that



certain kinds of inconsistencies, however, are desirable and should be maintained as long as possible. For instance, in object-oriented (OO) methods a candidate class is generally identified by applying a specific rule. While applying the rule, the software engineer has to follow a consistency constraint: 'an entity is either a candidate class or not a candidate class, but not partially both'. But the software engineer may perceive that an entity partially fulfils the specific rule for a candidate class and thus would classify the entity as a partial class. However this classification as a partial class is considered as an inconsistent class definition by the current OO methods. This is an example of multiple conflicting solutions for the same problem. In those cases each solution should be preserved to allow further refinements along the development problem. In fact, in such cases, too early a resolution of inconsistencies may result in loss of information and excessive restriction of the design space.

The need for tolerating inconsistencies during software development has been pioneered by [Balzer91], who proposes a formalism that allows development environments to tolerate and manage their inconsistencies. This is done through the softening of constraints, without introducing special cases, but by treating violations as temporary exceptions that will eventually be corrected. Until it is corrected the inconsistent data is automatically marked by guards, that enable to identify it to the procedures that can help resolve the inconsistency (which normally consists of notifying and involving human agents) as well as to protect such inconsistent data from other procedures sensitive to the inconsistency.

Marcelloni and Akşit [Marcelloni99, Marcelloni00] report two major problems how rules (used to identify and classify an entity) are defined and applied in current OO methods: two-valued logic cannot effectively express the approximate and inexact nature of a typical software development process (allowing loss of information); and, the influence of contextual factors on rules, is generally not modelled explicitly.

Contextual factors may influence validity of the result of a methodological rule in two ways: the validity of a rule may depend on contextual factors such as application domain, changes in user's interest and technological advances; and the input of a rule can be largely context dependent. This last factor can be illustrated with the following example. In a rule to decide if an entity is a class based on its descriptive properties, the elimination of a class is based on whether the software engineers' perception finds that tentative class more descriptive than an equivalent tentative class [Marcelloni99, Marcelloni00].

The loss of information and excessive restriction of the design space that can be caused by too early a resolution of inconsistencies is due to the so called quantization error [Marcelloni99]. The quantization error is the difference between the perception of the software engineer and the two 'quantization levels' imposed by the methodological rules. Examples of such rules are the ones prescribed by methods to classify entities as relevant or not relevant. Instead, inconsistencies should be considered as useful components of the development process and techniques should be adopted to tolerate these inconsistencies as long as needed.

Marcelloni and Akşit [Marcelloni01] propose three requirements for improving current software methods, namely:

1. Reduce the quantization error and its negative effects, through the preservation of the desired inconsistencies, and its resolution only when necessary. Consider that a demand for resolving an inconsistency may be context or language dependent (contextual bias problem). Thus, the objective of preserving inconsistencies has not to be achieved to the detriment of the intuitiveness of the methods. In particular the adopted rules, alternative rules and measures must be expressed in an intuitive way.

2. Provide a measure for alternative solutions, as they are not necessarily all equally valid.
3. Manage increasing complexity of design due to deferring consistency enforcement; finding techniques to manage the trade-off between increased complexity without necessarily giving up the design flexibility.

Marcelloni and Akşit [Marcelloni01] focus on modelling and handling desirable inconsistencies, proposing a fuzzy logic-based software development technique for handling inconsistencies during the software development process.

Marcelloni and Akşit [Marcelloni04, Marcelloni04a] characterize a software development method in terms of two major components: artefacts types, such as classes, operations, relations; and methodological rules. Each artefact type is characterized by a set of properties whose values determine the membership (value) of an artefact to that type. The relation between property values and the membership values is defined by heuristics, typically expressed in natural language. Artefacts may have some causal order among each other. The heuristics implicitly express how an artefact is causally related to other artefacts. The causal order among artefacts identifies the software process. The properties of artefacts are modelled as linguistic variables, and the methodological heuristics are defined as fuzzy rules. Artefact types are represented as fuzzy categories. The application of each rule infers, from the property values, a value of membership of an artefact to an artefact type. Thus an artefact can be an instance of several (possibly conflicting) artefact types with different extents. This fact can be considered a measure of the alternative, i.e., the artefact can be considered of a certain type 'A' with a certain measure 'a', or of another type 'B' with a measure 'b'.

The use of fuzzy logic presents several advantages. It enables the effective modelling of inconsistencies without altering the intuitive expressiveness of current methods. It offers, also, a unique opportunity to both model methodological rules and handle

inconsistencies within the same framework. Marcelloni and Akşit [Marcelloni01] approach respect the three requirements they propose and that are listed above, and thus improve current software methods, namely:

1. It reduces the quantization error, through the proposal of fuzzy logical methodological rules, which increase the number of possible values for properties of artefact types and consequently the number of quantization levels [Marcelloni99]. Also in the fuzzy-logic based method, the accumulation of the quantization error during the software development process is much less than the accumulation of error in the classical logic-based method [Marcelloni00]. The use of fuzzy logics instead of classical-based logics enables to capture as much as possible of the software engineer perception.
2. The fuzzy-logic based method not only allows inconsistencies but also associates a measure for each inconsistent alternative. In the end these measures are used to resolve inconsistencies.
3. To manage complexity the software engineer can reduce the design space, through the establishment of a threshold on membership values of an artefact to an artefact type. Other manners to reduce complexity include strategies such as to guide the software engineer to work on the alternative with highest value, and if by any chance this measure decreases other alternatives can be brought to attention.

Akşit and Marcelloni [Akşit01] implemented and tested a small fuzzy-logic based method using an experimental CASE environment. They showed that the proposed approach increases the adaptability and reusability of design models, when compared with the design models developed using standard methods where inconsistencies are resolved during the application of each rule. In the fuzzy logic-based method inconsistencies are left, and a priori none of the artefacts is eliminated.

The fuzzy-logic based method can be viewed as a learning process as after the application of each rule a new aspect of the problem in hands is learned. This new aspect can modify the previously gathered property values. Using fuzzy logic theory it is possible to reason and compose the results of the rules. This leads to adaptable and reusable design models.

In [Marcelloni04a] the authors formally introduce the quantization-error and contextual bias problems which affect software development methods based on two-valued logic. These problems were explained in an intuitive manner in their previous work [Marcelloni99, Marcelloni00, Akşit01, and Marcelloni01].

Marcelloni and Akşit [Marcelloni04a] draw as conclusion, from their previous work, that the use of classical sets and classical logic lack expressive power to model the software development process appropriately. In fact software development is a human intensive activity where perception and intuition have an important and extensive role. In [Marcelloni04a] they show how fuzzy logic can be a valid expressive tool to manage the software development process. This work is based on the claim by Zadeh [Zadeh99] that fuzzy logic provides a unique foundation for a computational theory of perceptions, i.e., modelling how humans make perception-based rational decisions in an environment of imprecision, uncertainty, and partial truth. This computational theory is the computational theory of perceptions (CTP) proposed by Zadeh [Zadeh99].

Although the focus of this thesis is not to consider conflict and ambiguity from the temporal point of view, it was interesting to study these works. They model and manage the abstraction quality of knowledge sources and inconsistencies in the broader context of imperfection management in the software development process. The first work [Tekinerdoğan03] proposes the use of fuzzy logics to evaluate the quality of domain knowledge. The remaining works concern decision that frequently

has to happen when information is not sufficient now, and may or may not be sufficient in the future. This lack of sufficient information may produce inconsistencies. The use of fuzzy logics is proposed to model and manage inconsistencies, while keeping the intuitive expressiveness of current methods.

These works together with Noppen's work (that will be discussed in next Section) were the most influential in showing the potentiality of fuzzy logics and probability theory in modelling imperfection across the software development process.

### **2.3.3 Fuzzy functional and non-functional requirements, and design artifacts including design alternatives**

Noppen [Noppen07], in his thesis, extends Shaw [Shaw95, Shaw96] notion of credential through mathematical models to express the credibility of information. These models work upon functional requirements, non-functional (or quality) requirements, design artefacts and software process management artefacts<sup>6</sup>. They can be used to systematically analyse decision alternatives and maintain the credibility information. They also provide information on some relations between artifacts and decisions.

Noppen defines perfect information to be the information that contains all the attributes and values with sufficient precision and certainty for the purpose for which it is used. Thus, imperfect information is the information that is not perfect. Uncertain information is the information that is imperfect, but will become certain at some point

---

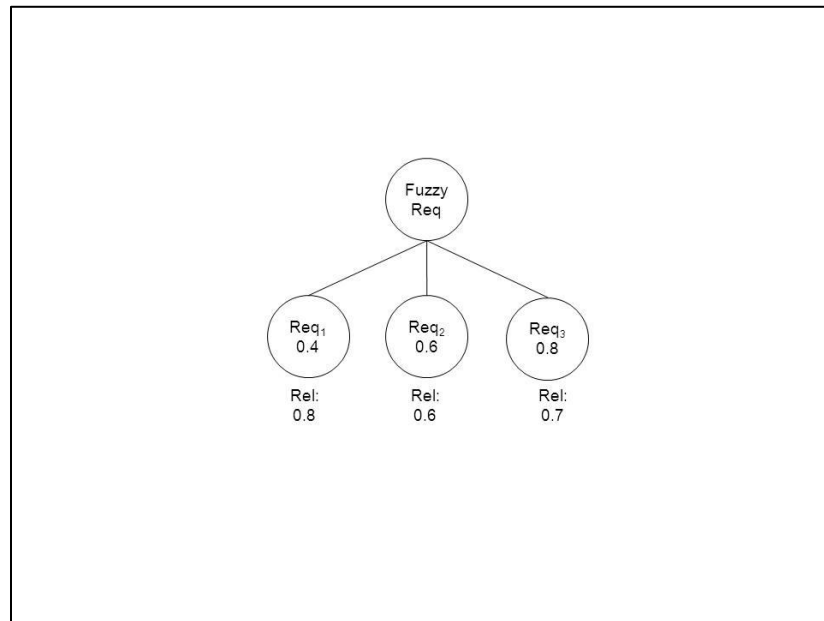
<sup>6</sup> In the description of Noppen's work it will be used 'artifact' as Noppen used

in the future. And, imprecise Information is the information that is imperfect, and that will remain imperfect to a certain degree.

Noppen thesis addresses the problem of imperfect information in functional requirements through the concept of fuzzy functional requirements. This concept enables the inclusion of alternative requirement interpretations by means of fuzzy sets. The fuzzy functional requirements concept is combined with the Artifact Trace Model to support the analysis of the resulting design based on different trade-offs, such as cost minimization or relevance maximization. The Artifact Trace Model is a directed graph, in which the nodes are the intermediate design artefacts that result from the software development process. The ATM models traditional perfect requirements as singular nodes. In the fuzzy requirement concept an imperfect requirement is replaced by a number of possible interpretations, each of which is tagged with a value between zero and one corresponding to a particular stakeholder interest. By treating these interpretations as normal “perfect” requirements, software engineers are enabled to continue the development process as usual. In the case of Figure 2-1 which depicts a fuzzy requirement in a tree form and its alternative interpretations (from [Noppen07]), the child nodes also have a value for the stakeholder interest relevancy. When the Artifact Trace Model is used with “crisp” perfect requirements, its goal is to determine the best subset of all requirements that needed to be implemented. In the presence of fuzzy requirements, the goal moves to determine the best subset of all interpretations, which at least has one interpretation for each fuzzy requirement.

During software design a number of design decisions are taken in a sequential manner. Typically, for each design issue, several design alternatives (candidate solutions) are considered. These design alternatives are evaluated based on quality expectations, in order to establish an ordering among them. The design alternative that offers the best quality is then selected to fulfil the design issue in the design of

the system [Noppen07]. The software design activity of evaluating and selecting design alternatives, based on their expected quality attributes, is hampered by the presence of imperfect information. Both, the quality requirements (the non-functional requirements) and the quality estimations that are used in the evaluations can be considered to be imperfect.

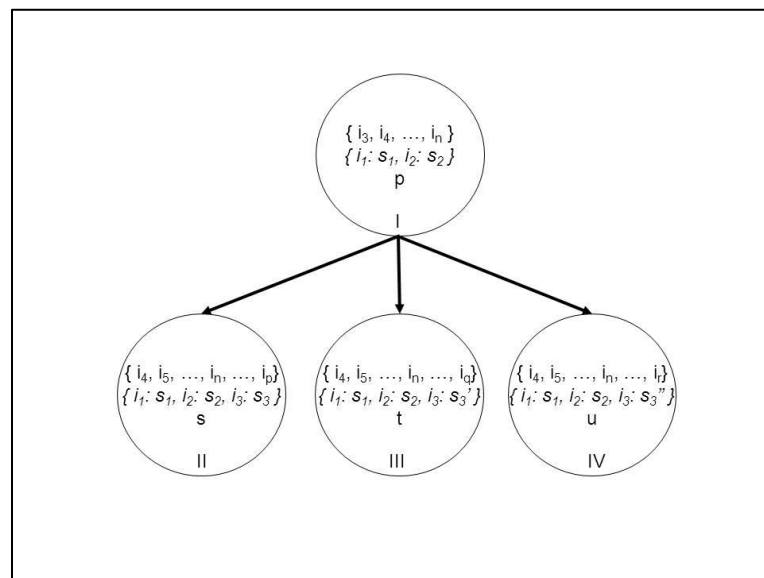


**Figure 2-1 A fuzzy requirement and its alternative interpretations [Noppen07].**

Noppen [Noppen07] proposes an approach for specifying numerical expressions in quality requirements (NFRs) and quality estimations that are subject to imperfection, by means of fuzzy sets and probability distributions. For these specifications, the definition of quality requirement is refined: a quality requirement is an interval of acceptable quality attributes. The model addresses two types of imperfection for quality requirements (NFRs): impreciseness and uncertainty. This approach is completed with the definition of comparison operators needed to evaluate imperfect requirements and imperfect estimations.



The specification of imperfect quality requirements and imperfect quality estimations is combined with the Design Tree Model (DTM) to form a decision support model for design decisions with support for imperfect information. The Design Tree Model is a tracing model that creates a design tree (a tree structure) and uses it to describe the design issues that have been resolved, the order in which they have been addressed and the design alternatives that have been considered. Figure 2-2 shows an example design decision with design alternatives.



**Figure 2-2 A design tree showing a design decision with design alternatives [Noppen07].**

Noppen [Noppen07] developed two separate tools (in fact three but this thesis focuses on the ones concerned with requirements). One of them implements the approach based on the Artifact Trace Model to assist the software engineer during the refinement steps and performs the optimisation steps for the trade-off between stakeholder interests and implementation effort. The interface of this tool depicts the direct graphs used in the Artifact Trace Model and provides textual areas to insert requirements and stakeholder interests. The other tool, the Decision Tracer Tool, traces the design decisions and contemplated alternatives during the software

development process according to the design tree approach. The user interface of the Designer Tracer Tool contains textual areas for the quality attributes and design issues and an area where the design tree is visualized.

The approaches described in this Section 2.3 were the first works to make the author of this thesis aware of the importance of making explicit the presence of imperfection in RE and SE, as well as the need to subsequently handle that imperfection. The study of these approaches enabled to gain a deeper understanding of imperfection in SE. This study permitted to understand the immense value and potential of the usage of modelling languages extended with fuzzy logics and/or probability theory; but it also prompted the need and importance of providing mechanisms to communicate the information about imperfection in a manner easier to understand and to handle, than the one provided by the modelling languages (which are too technical). These communication mechanisms may be integrated to complement these modelling language approaches in the construction of a broader framework to support imperfection in SE.

Noppen work was the one that made more evident the need for a more easy to understand and usable communication mechanism than the modelling language itself even if visualized through graphs.

## 2.4 Ambiguity identification and handling

This section starts with a sub-section describing works by Berry et al [Berry03] and Kamsties [Kamsties05] whose focus is mainly on understanding ambiguity in requirements. The remaining section describes some of the existent techniques to identify and handle ambiguity in requirements.

### 2.4.1 Understanding ambiguity in requirements

Berry, Kamsties, and Krieger [Berry03] produced a handbook aimed at describing the ambiguity phenomenon from several points of views ranging from the dictionary definition, to the linguistic definitions, and the software engineering definition<sup>7</sup>.

Berry et al [Berry03] present the techniques to reduce the level of ambiguity in natural language requirements organised according to the RE activity to which they are applicable.

For requirements elicitation the techniques<sup>8</sup> to reduce the level of ambiguity are as follows:

1. Establish a context, make it explicit and agreed to by all the stakeholders.
2. The requirements engineer should paraphrase what is understood from the stakeholders according to what he understood, to enable stakeholders to spot

---

<sup>7</sup> Part of this work has already been referred to by this thesis in chapter 1 when discussing the definition of ambiguity in RE.

<sup>8</sup> [Berry03] provides references for these techniques; these references are omitted here.

their own ambiguities. There are communication techniques that support these strategies.

For requirements documentation the techniques<sup>9</sup> to reduce the level of ambiguity are as follows:

1. Increase the precision of natural language.
2. Provide more context information.
3. Establish conventions on how ambiguous phrases shall be interpreted.

For requirements validation the techniques<sup>10</sup> to detect ambiguity are as follows:

1. Formalization of informal requirements: examples are described in previous section 2.3;
2. Pattern-driven techniques, such as:
  - a. reading techniques for requirements inspections: an example is described in section 2.4.2, and
  - b. natural language processing tools, such as the ones described in section 2.4.3.
3. Compare the interpretations of a document by different stakeholders: if they differ there is an ambiguity in the original document.
4. Communicate an interpretation back to the requirements author, after which he can easily point out misinterpretations.

According to Berry et al [Berry03] these last two techniques (3 and 4) are probably the most effective strategies for finding ambiguities in SRS. They are also the most

---

<sup>9</sup> [Berry03] provides references for these techniques; these references are omitted here.

<sup>10</sup> [Berry03] provides references for these techniques; these references are omitted here.

resource-demanding. The approach proposed in this thesis is based on the same philosophy that promoting communication among requirements engineers and stakeholders is a very effective strategy to enable identification and correct handling of ambiguity.

The handbook by Berry et al [Berry03] describes solutions for avoiding ambiguities, covering a variety of common linguistic, lexical, structural, scope, referential, and language-error ambiguities. This work also reviews other writing guides for their advice on avoiding ambiguity.

Berry et al work was fruitfully used in this thesis. Apart from sharing the same conviction that promoting communication among requirements engineers and stakeholders is a very effective strategy to enable identification and correct handling of ambiguity, and imperfection in general; Berry et al work provide the ambiguity definition (and discussion on the concept) that serves the basis of the ambiguity definition, and the heuristics to detect ambiguity proposed in this thesis.

Kamsties [Kamsties05] reports the results of empirical studies concerned with understanding ambiguity in requirements. The results show that:

1. Ambiguities are reported less often, but are resolved unconsciously more often than other types of imperfections. This unconscious resolution of ambiguities leads to implicit assumptions that are likely to be wrong in complex systems.
2. A considerable number of ambiguities tend to be misinterpreted (20% to 37%).

Supported by these empirical study results, Kamsties concludes that:

1. Ambiguity problems are not solved by formalization during further development activities, and
2. It is difficult to detect ambiguities, even if the reader is aware of all the facets of ambiguity.

This thesis is based in the same conviction as the first conclusion of Kamsties: that imperfection in RE is not solved by formalization during further development activities. This is why this thesis approach proposes to separate the processing of the information about the imperfection, which benefits from formalization, from the issue of communicating that imperfection. This thesis also takes the view that it is difficult for a reader to detect ambiguities, and this is why it proposes tools to automatize the identification of ambiguities (as well as conflicts).

#### **2.4.2 Pattern-driven inspection techniques to identify ambiguities**

Kamsties, Berry, and Paech [Kamsties01] proposed a metamodel of requirements specification sentences as patterns to allow identification of ambiguities in natural language specifications. This metamodel needs to be adapted to the domain of the requirements specification. Kamsties et al [Kamsties01] propose pattern-driven inspection techniques, namely checklists and scenario-based reading, whose

effectiveness in detecting ambiguities in natural language requirements specifications has been empirically validated.

This thesis approach proposes the use of tools to identify ambiguities, but it calls attention to the fact that the ultimate choice of ambiguities, to be handled through stakeholders meetings, pertains to the requirement engineers. If it is true that, in general, it is better to have tools to detect ambiguities, it is also true that the output of these tools, consisting of lists of detected ambiguities (in the best of options ranked according probability), will have to be handled by a human user. Thus, checklists can be useful in guiding the user to distinguish the most relevant ambiguities to treat. Checklists that present and organise the knowledge about ambiguity phenomenon, such as the ones presented in [Kamsties01] constitute valuable input for the developers of ambiguity detection tools.

Concerning the scenarios such as the ones proposed by Kamsties et al [Kamsties01], they can be used, in stakeholder meetings, in complement to the jigsaw puzzle metaphor, for instance to discuss deeper the connection between ambiguity with a possible conflict between two requirements.

### **2.4.3 Software linguistic tools to support ambiguity identification**

Instead of, or in addition to, manual reading software linguistic tools can be used to support ambiguity identification in natural language requirement specifications. Kiyavitskaya et al [Kiyavitskaya08] propose a two-step tool-assisted approach to

identify ambiguities in natural language requirements specifications. In the first step, a tool applies a set of ambiguity measures to a requirements specification in order to identify potentially ambiguous sentences, and computes the level of ambiguity of each sentence. In the second step, another tool shows what specifically is potentially ambiguous in each potentially ambiguous sentence. The final decision of ambiguity remains with the human users of the tools. The authors performed some experiments with prototypes of the two tools. These experiments showed that any such tool should respect the following requirements:

- 100% recall,
- not too much imprecision, i.e., the user should not be inundated by false positives to the point of preferring not to use the tool, and
- high summarization, i.e., the size of the output that the user must wade through is a small fraction of the size of the input to the tool.

Gleich et al [Gleich10] present a tool for ambiguity detection, in natural language requirements specification, which relies on a grep-like technique to detect ambiguities. This makes the tool highly reliable, applicable to different languages and independent from error-prone natural language parsing. For every sentence that contains a detected ambiguity, the tool provides an explanation why the detection result represents a potential problem. The tool provides reliable ambiguity detection, in the sense that it detects four times as many genuine ambiguities as an average human analyst.

The reliability of Gleich et al [Gleich10] tool to identify ambiguities, together with the production of an explanation of why a certain part of an SRS has been marked as an ambiguity, made it a very well suited choice to integrate the approach proposed in this thesis. This will be further explained in Chapter 4.



#### 2.4.4 Nocuous and innocuous ambiguity

Some approaches for ambiguity identification follow the philosophy of classifying ambiguity in natural languages into nocuous or innocuous [Chantree06, Yang11]. Using this classification these approaches can reduce the number of ambiguity cases which need to be considered.

Chantree et al [Chantree06] and Yang et al [Yang11] defend that ambiguity is not just a property of a text, but a conjoint property of the text and of the interpretations held by the readers of that text. In fact, any ambiguity presented in a requirement can be innocuous in a certain context and nocuous in another. It may be innocuous if the readers share the domain knowledge leading them all to choose the same interpretation for that possible ambiguity. But it may be nocuous if there is not such a sharing of domain knowledge. In fact, not all potential cases of ambiguity are potentially harmful. Thus, these works [Chantree06, Yang11] aim to classify ambiguity in requirements as innocuous and nocuous to inform the analyst of the potential dangerous ambiguity cases.

The approach proposed in Yang et al [Yang11] builds on previous work, including that of Chantree et al [Chantree06]. Although these approaches treat different specific types of ambiguity<sup>11</sup>, the technique used follows the same general steps:

1. The starting point is a dataset of ambiguous phrases from a requirements corpus, and associated human judgements about their interpretation.
2. Then a classifier is trained using heuristics, based on linguistic features of the text and the distribution of the judgements, to automatically replicate these judgements.

---

<sup>11</sup> [Chantree06] works with coordination ambiguity, and [Yang11] with anaphoric ambiguity.

3. The heuristics eliminate ambiguities which people interpret easily, leaving the nocuous ones to be analysed and rewritten by hand.

Chantree et al [Chantree06] report that many of the proposed heuristics achieve high precision, and recall is greatly increased when they are used in combination.

Yang et al [Yang11] also report that their approach achieves high recall with a consistent improvement on baseline precision subject to appropriate ambiguity thresholds, allowing highlighting realistic and problematic ambiguities. An interesting feature of this work is the possibility to vary the sensitivity of the analysis depending on the readership (i.e., group of readers).

These tools [Chantree06, Yang11] were chosen to integrate the approach proposed in this thesis. They are based in the interesting idea of classifying ambiguity in natural languages into nocuous or innocuous, enabling to reduce the number of ambiguity cases which need to be considered. As stakeholder meetings are time consuming, it is very useful to select the most pertinent cases, which would be the most nocuous. This will be further explained in Chapter 4.

## **2.5 Conflict identification and handling**

There are a number of requirements engineering approaches that focus on conflict identification. This section describes and discusses some examples.

## 2.5.1 Non-Functional Requirements Framework (NFRF)

The Non-Functional Requirements Framework (NFRF) [Mylopoulos92, Chung00] gathers and records several types of information during requirements engineering. The information gathered is visualized using two types of diagrams: the softgoals interdependency graphs (SIGs) and the catalogues.

The central concept of the NFRF is the softgoal, which represents a goal that has no clear-cut definition and/or criteria as to whether it is satisfied or not. The softgoals are used to represent non-functional requirements. Softgoals are related through relationships which represent the interdependency of one softgoal to another [Chung00]. The software interdependency graphs (SIGs) are graphs with two functions: to record the developer's consideration of softgoals, and to show the interdependencies among softgoals. SIGs represent the softgoals visually as clouds with associated labels containing values to represent the degree to which a softgoal is achieved. SIGs also show the interdependency links, represented as lines, often with arrowheads. These interdependency links (called explicit interdependencies) show the refinements of "parent" softgoals downwards into other more specific "offspring" softgoals, and the contribution (impact) of "offspring" softgoals upwards upon the meeting of "parent" softgoals. NFRF uses an evaluation procedure (labelling algorithm) to determine whether softgoals are achieved, taking into account labels, contributions, and also decisions by the developer.

The catalogue "diagrams" organize previously accumulated design knowledge. There are three kinds of catalogues. The NFR type catalogues organise the knowledge about particular types of NFRs (e.g. catalogue about security, another about performance). The methods catalogues store knowledge about development techniques (methods) which are intended to help meet requirements (e.g. "response

time for accounts” can be operationalized with “use indexing” or “use uncompressed format”). An interdependency generated as the result of a method application is an explicit interdependency. The correlation rule catalogues organise knowledge about implicit interdependencies (correlations) among softgoals. These implicit interdependencies are detected by comparing a portion of a SIG with a catalogue of relationships among softgoals. These interdependencies can either be positive or negative contributions and are shown as dash lines in figures.

NFRF also enables to relate functional requirements with non-functional requirements and the decisions made for the target system.

The Non-functional requirements framework represents an impressive and much valuable approach to gather, organise and thus making explicit a vast amount of information during the development process. This covers not only information on the requirements for the system but also on the development process itself (e.g. the decisions the developer made).

In particular, conflict detection in NFRF is achieved through the detection of negative contributions of both explicit and implicit interdependencies among softgoals. Refinements of more general softgoals into more specific ones, as well as, operationalizations can help detect and clarify ambiguities. Refining or operationalizing a softgoal makes it more explicit what exactly the developer is thinking about that softgoal.

The i\* [Yu97] framework is an adaptation of the NFRF framework. It was developed for modelling and reasoning about organizational environments and their information systems. It consists of two main modelling components. The Strategic Dependency model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations

of systems and environments. The i\* framework builds on a knowledge representation approach to information system development [Mylopoulos90].

The NFR framework [Mylopoulos92, Chung00] offered valuable inspiration for this thesis approach. NFR catalogues are one of the examples of catalogues that can be used in the approach. These catalogues may be used as the domain of qualities to consider in applying the heuristics proposed by the approach. In particular, the Quality dependency conflict heuristic, presented in Section 4.3.2, builds on the NFR type catalogues, which support the discovery of conflicts, when there are qualities that require other qualities.

## **2.5.2 Viewpoint-based approaches**

Sommerville and Sawyer [Sommerville97a] define a viewpoint-based approach to RE as one that recognises that all the information about a system's requirements cannot be discovered by considering the system from a single perspective. It is thus needed to collect and organise requirements from a number of perspectives, which are represented by 'viewpoints'. A 'viewpoint' is an encapsulation of partial information about a system's requirements. To have the final specification of the system information from different viewpoints must be integrated.

This section surveys two viewpoint-based approaches to RE: ViewPoints and Inconsistency Management (VIM), and Process and Requirements Engineering Viewpoints (Preview).

### **2.5.2.1 ViewPoints and Inconsistency Management (VIM)**

Finkelstein et al [Finkelstein94, Easterbrook96] propose a viewpoint-based approach, based on the ViewPoints framework [Finkelstein92, Nuseibeh94, Nuseibeh94a], and aimed at inconsistency management. Inconsistencies between ViewPoints are managed by explicitly representing relationships between them, and recording both resolved and unresolved inconsistencies.

This approach works with the concept of inconsistency. The authors [Easterbrook96] discuss the difference between inconsistency and conflict. They explain that an inconsistency occurs if a rule has been broken. Such rules are defined by method designers, to specify the correct use of a notation (and method), and the relationship between different notations (and methods). They define conflict as the interference in the goals of one party caused by the actions of another party. But this does not imply that any consistency rules have been broken, thus it does not imply that there is an inconsistency. They also show that despite the fact that their approach is based on management of inconsistency, it helps with the identification and resolution of conflicts and mistakes. But, it is not guaranteed that all conflicts (and mistakes) will manifest as inconsistencies. In such cases of conflicts not detected (if they did not manifest as inconsistencies), the cause is a weakness in the definition of consistency rules.

Nuseibeh and Finkelstein [Nuseibeh92] developed the Viewer, which is a prototype computer-based environment and associated tools with distinct modes of use: method design and method use.

One interesting aspect of this approach is the possibility to use multiple representations and development methods, i.e., the requirements do not need to be all described using the same representation (notation) neither the software

development method needs to be the same. The approach enables a specific method to be implemented through the definition of a set of ViewPoint templates, which together describe the set of notations provided by the method, and the rules by which they are used independently and together [Easterbrook96]. This aspect is very relevant (as seen in Chapter 1) for the support of a heterogeneous community of users (engineers, other stakeholders) who should be enabled to input requirements using different representations and methods.

One other relevant aspect is the issue of inconsistency handling, i.e., how to act in the presence of inconsistencies. The approach propose the following list of examples of possible appropriate actions to take when an inconsistency is detected: ignore, delay, circumvent, ameliorate, and resolve. Another hypothesis concerning inconsistency handling is tolerating inconsistency. This possibility has already been introduced in [Easterbrook96] and is further exploited at [Nuseibeh01]. Nuseibeh et al [Nuseibeh01] argue that it is not always possible to avoid inconsistency, and in many cases insisting on maintaining consistency can be counterproductive. Thus, tools that tolerate and carefully manage inconsistency provide more flexibility and thus better requirements engineering support.

Furthermore, Nuseibeh et al, argue that the biggest problems arise not with the presence of inconsistencies but when there are undetected inconsistencies [Nuseibeh01]. This thesis takes the same view that conflict detection support has to enable consideration of a range of possible actions to manage it, including toleration.

Easterbrook and Nuseibeh [Easterbrook96] also conclude that the analysis of inconsistency helps reveal the concept models used and assumptions made by the development participants. Likewise, this thesis foresees that imperfection management should explore the fact that imperfections in requirements are unavoidable and thus should “start” by making them explicit, because otherwise they

will remain implicit and cause potentially serious problems. In fact many a times the way requirements are expressed hide the conceptual models and assumptions of those who wrote (or represented) them.

The discussion about the imperfection of requirements in VIM [Easterbrook96] is done at the level of the formalisms used to describe the requirements (and that's why what is detected is inconsistency between formalisms and not conflicts between requirements) and not at the level of the semantics of the requirements. This fact presents further problems that will be discussed next, when presenting aspect-oriented requirements approaches with semantics-based composition.

#### **2.5.2.2 Process and Requirements Engineering Viewpoints (Preview)**

Sommerville and Sawyer [Sommerville97a] propose Process and Requirements Engineering Viewpoints (Preview) a flexible, 'lightweight' model of viewpoints. They recognize that "for technical, human and environmental reasons, system requirements specifications will always be imperfect". They consider that the quality of specifications can be improved in two ways:

1. "By improving the requirements engineering process so that errors are not introduced into the specification
2. By improving the organisation and presentation of the specification itself so that it is more amenable to validation".

Preview is presented as an approach to system requirements engineering which addresses both of these improvement dimensions.



Similar to VIM, Preview enables the requirements associated with a viewpoint to be expressed in any notation (from natural language to formal notations). As pointed out before this aspect is very relevant (as seen in Chapter 1) for the support of a heterogeneous community of users (engineers, other stakeholders) who should be enabled to input requirements using different representations and methods.

Another key characteristic of Preview is that the analysis is driven by a set of concerns<sup>12</sup> which reflect the critical non-functional characteristics of the system. This view of the decisive role of the NFRs in the development of the analysis is shared by this thesis.

Preview prescribes to start with an initial outline of requirements and ask sources to describe its deficiencies and omissions. This thesis takes inspiration from this and complements Preview by proposing a communication mechanism that makes conflicts explicit, exactly to promote the cooperation of stakeholders in identification/confirmation of imperfections in requirements.

During the discovery of the requirements for each viewpoint, it is sometimes helpful to consider the decomposition of a viewpoint into sub-viewpoints. One of the reasons to consider this decomposition is the presence of conflict among the requirements of a viewpoint, especially if the sources of the requirements have imperfectly matched foci. This decomposition, per se, does not solve the conflict, but through the division of the problem and the association of the requirements with their source, it helps the requirements negotiation process.

The requirements belonging to viewpoints whose foci intersect are classified, using a tabular method, in three categories:

---

<sup>12</sup> Viewpoint concerns correspond to high-level strategic objectives for the system. They are used to ensure that the requirements for the system are consistent with the business goals of the procuring organisation [Somerville97a].

- Overlapping: There is some overlapping between requirements which should be discussed aiming at simplifying the requirements. A '1000' is used to indicate overlapping requirements.
- Conflicting: There is a conflict between the two requirements which should be resolved. A '1' is used to indicate conflicting requirements.
- Independent: The requirements are independent. A '0' is used to indicate two independent requirements.

The numeric values inserted in the table enable to use simple arithmetic on the values of rows and columns to obtain the number of overlaps and conflicts. These numbers reveal the most problematic requirements, which is very useful information to inform the negotiation process.

Preview does not prescribe how conflicts are resolved neither how overlapping requirements are rationalised.

The Preview approach [Sommerville97a] offered valuable inspiration for this thesis approach. In particular, the Synonymous/antonymous quality conflict heuristic and the Actions operationalizing quality conflict heuristic are influenced by Preview approach to find conflicts. This will be further explained in Section 4.3.2.

### 2.5.3 Detecting conflicts in aspect oriented textual requirements

A key goal of Aspect-Oriented Requirements Engineering (AORE) is to identify possible crosscutting concerns<sup>13</sup>, and to develop composition specifications, which can be used to reason about potential conflicts in the requirements [Weston09].

Initially AORE approaches used syntactic-based composition. Chitchyan et al [Chitchyan07] showed that syntactic-based composition leads to several problems including pointcut<sup>14</sup> fragility, a lack of expressiveness and a loss of information concerning the intent of the stakeholders and the requirements engineer. In order to combat these problems, they propose an expressive composition approach based on the grammatical syntax and semantics of the natural language. This is done through the Requirements Description Language (RDL). Using RDL, compositions can be specified using natural-language operators, allowing engineers a semantically rich vocabulary instead of pure syntax. This means that conflict detection can be performed on the basis of the semantics of the requirements and their compositions, rather than merely their syntactic references. The implications of this approach are that conflict detection is much more robust to change. It also improves the probability to detect more subtle semantic conflicts and allows deriving the meaning of a conflict much more readily. In order to achieve this, the composition mechanism must have a formal underpinning, such that the natural language semantics in the compositions can be unambiguously understood. Weston et al [Weston09] present such a formal framework for the natural language-based compositions, which uses predicate logic including explicit temporal elements.

---

<sup>13</sup> A concern is an interest, which pertains to the system's development, its operation or any other matters that are critical or otherwise important to one or more stakeholders [Berg05].

<sup>14</sup> Pointcut is an expression which can pick out one or more requirements or other elements at which the composition applies [Weston09].

Automation support has been proposed both by Weston et al [Weston09] and by Sardinha et al [Sardinha09]. Both proposals work from requirements in natural-language that have been “translated” to RDL. The “translation” to RDL corresponds to the identification and structuring of requirements.

Weston et al [Weston09] propose automation through a set of of-the-shelf tools that start by the formalisation of compositions and continue with the detection of the conflicts through the use of the logical conjunction of the formalisation of compositions, in particular searching for temporal overlap between compositions.

Sardinha et al [Sardinha09] propose a tool, called EA-Analyzer, to support for conflict detection through a novel application of a Bayesian learning method that has been effective at classifying text. The output of this text classification is a list with the words and its associated probability of pertaining to the Conflict class.

Sardinha et al [Sardinha10] developed further their approach of conflict management to include not only the detection of conflicts (as in [Sardinha09]) but also the maximization of stakeholder’s satisfaction without violating the availability of resources. The third step of their approach is the negotiation with stakeholders which is exactly the goal of the jigsaw puzzle metaphor this thesis proposes.

This thesis indicates EA-Analyzer [Sardinha09] to integrate the proposed approach, specifically for identifying and ranking conflicts. This choice is justified as conflict detection in EA-Analyzer is performed on the basis of the requirements semantics, minimizing loss of expressiveness and of information, while improving probability to find subtle conflicts.

The next section presents the analysis of the works described in this chapter.

## 2.6 Analysis

### 2.6.1 Artefacts supported

The following table (Table 2-1) shows the coverage of imperfect information support in terms of the artefacts supported by the approaches surveyed.

	Requirements	Relations btw requirements	Soft. development process (sources, artefacts, method_ logical, rules, decisions)
Yen93 and Lee03	x	x	
Liu96	x	x	
Tekinerdoğan03			x
Akşit01, Marcelloni01, Marcelloni04 and Marcelloni04a			x
Noppen07	x		x
Kiyavitskaya08 and Gleich10	x		
Chantree06 and Yang11	x		
NFRF	x	x	x
VIM	x	x	
Preview	x	x	
Weston09 and Sardinha09	x	x	

**Table 2-1 Artefacts supported by the approaches studied.**

The approaches studied are divided essentially in two groups: the ones that focus more on the RE phase and thus support this phase artefacts: requirements and relationships between them; and the ones that work with the software development process supporting issues such as quality of knowledge sources, the decisions concerning, for instance, the classification of artefacts together with the associated inconsistencies.

## 2.6.2 How imperfection is supported, what for and with what user interaction

The following table (Table 2-2) shows, for each work surveyed the SE activity supported, and type of support provided concerning imperfect information. It also shows the technique used, and the type of user interaction provided.

	<b>SE activities and type of support</b>	<b>Technique</b>	<b>User interaction</b>
<b>Yen93 and Lee03</b>	RE: model vague requirements; exploration of trade-offs among vague requirements; identify and classify relations btw requirements	Fuzzy logic	Not implemented
<b>Liu96</b>	RE: model imprecise requirements; derivation of the classification of relationships between imprecise requirements, but from relationships already identified and classified	Fuzzy logic	Not implemented
<b>Tekinerdoğan03</b>	Development process: evaluation of domain knowledge sources	Fuzzy logic	Not implemented
<b>Akşit01, Marcelloni01, Marcelloni04 and Marcelloni04a</b>	Development process: model and handle inconsistencies (not obeying consistency constraints) in development decisions.	Fuzzy logic	Not implemented
<b>Noppen07</b>	RE: model multiple interpretations of ambiguous requirements Development process: support the tracing between the FR and the components that implement it; support quality estimations for NFR; support analysis of designs based on different trade-offs	Fuzzy logic and probability theory	Directed graph structure, tree structure and textual descriptions
<b>Kiyavitskaya08 and Gleich10</b>	RE: identification of potentially ambiguities, computes the level of ambiguity of each case, and shows what is potentially ambiguous Gleich10: adds 'why' explanation, for every ambiguity detected	Natural language processing	Tool showing the identified ambiguities and other computed information Gleich10: the tool is web-based

**Table 2-2 SE activities, types of support, technique used and user-interaction provided by the approaches studied.**

	<b>SE activities and type of support</b>	<b>Technique</b>	<b>User interaction</b>
<b>Chantree06 and Yang11</b>	RE: ambiguity classification into nocuous and innocuous; identify the potential dangerous ambiguity cases	Corpus-based statistics information and machine learning	Chantree06: not implemented Yang11: classifier, which notifies authors that text may lead to misunderstandings
<b>NFRF</b>	RE: detection of conflict – through detection of negative contributions of both explicit and implicit interdependencies among softgoals; help detect ambiguity through refining or operationalizing of softgoals; record the developer consideration of soft goals and his decisions	RE goal-oriented approach	Diagram-based visualization; not implemented: it's a methodological proposal
<b>VIM</b>	RE: detection of conflict through detection of inconsistency in the rules defined for a notation; inconsistency management	RE viewpoint - oriented approach	Frame-based visualization with text , tables, and diagrams
<b>Preview</b>	RE: identification of conflicts, through the notion of viewpoint focus – requirements belonging to viewpoints whose foci intersect; selection of the most problematic requirements – the ones with higher numbers of conflicts and overlaps	RE viewpoint - oriented approach	Not implemented: it's a methodological proposal
<b>Weston09</b>	RE: identification of conflict using the logical conjunction of the formalisation of semantics-based compositions, in particular temporal overlap between compositions	RE aspect-oriented approach	Not implemented: proposal for implementation steps
<b>Sardinha09</b>	RE: identification of conflict – 1 <sup>st</sup> structuring requirements using semantics-based composition; 2 <sup>nd</sup> applying a Bayesian learning method to classify text, retrieving the probability of a word belonging to the conflict class	RE aspect-oriented approach and Bayesian learning method	Table-based interface showing the list of words and its associated probabilities for being classified as conflict or harmony.

**Table 2-2 (cont.) SE activities, types of support, technique used and user-interaction provided by the approaches studied.**

In a first group of works [Yen93, Lee03, Liu96, NFRF, VIM, Tekinerdoğan03, Akşit01, Marcelloni01, Marcelloni04, Marcelloni04a, and Noppen07]) the focus is to set up mathematical foundations (using fuzzy logics, probability theory, and other RE

models) to integrate imperfect information in the artefacts (e.g. requirements, architectural components). It is upon these mathematical foundations that additional formalisms (e.g. graphs, trees) are built to support imperfection handling and decision (in the face of imperfection). These approaches are designed to provide support for software engineers in activities where the communication and interaction with tools is technical. In fact, in these approaches, the information about the imperfection is “coded” in the formalisms, which requires that software engineers are knowledgeable in the formalisms. This can raise questions about ease of use, even for software engineers.

This thesis proposes a separation between the mechanisms for identifying and collecting the information about the imperfection in requirements, from the communication mechanism. Such a separation enables to choose a communication mechanism much more adequate to the RE tasks, than the formalisms proposed by the first group of works, described in this Chapter.

In the second group of works for ambiguity and conflict detection [Kiyavitskaya08, Gleich10, Chantree06, Yang11, Weston09, Sardinha09], the focus is to identify and sometimes also manage ambiguity and/or conflict, and to provide this information as lists of pieces of text with probable ambiguity or conflict cases. Usually the output includes more characteristics such as a value of probability for that imperfection, and in some cases even explanations for the notification of possible imperfection presence.

The fact that these works present the identified ambiguities and conflicts as text lists, make them the “perfect” tools/approaches (in the current state-of-the-art) to integrate into an approach to identify and handle imperfection in RE, together with a communication mechanism that can be separately developed to be the most appropriate both for software engineers and stakeholders.



## 2.7 Research agenda

The vision of this thesis has at its foundations the importance to make explicit, for the users of RE tools, the factors that influence decision in SE and that are usually implicit. This is the case for imperfection and in particular for ambiguity and conflict. The explicit support of implicit aspects that influence software development would be greatly beneficial to the communication among teams of developers and between software developers and other stakeholders. The decision process and needed trade-offs would be much more informed, through more information being explicit and conveyed in an appropriate way.

As far as this chapter showed, there have been two ways to identify and store information about the ambiguity and conflict that exist in requirements documentation:

- One way is to convey the information about ambiguity and conflict embedded into a formalism.
- The other way is through tools dedicated to ambiguity and conflict detection, which produce lists of imperfection cases.

These two ways to identify and store information about the ambiguity and conflict are not appropriate (or in the second case the best possible) to communicate ambiguity and conflict, both amongst engineers, as well as, with stakeholders when there is a need to consult them to clarify these imperfections. This lead this thesis approach to pursue the open research challenges (described in Section 1.4.4) concerning the handing of ambiguity and conflict in RE, which succinctly are:

- separation of the processing of the information about the imperfection from the issue of communicating that imperfection, and

- need for communication mechanisms to adequately raise awareness of conflict and ambiguity in requirements, improving communication and cooperation amongst engineers and stakeholders during consultation meetings.

A grand research challenge (broader than the scope of this thesis) is to build a broader framework to handle imperfection covering the software development lifecycle.

It would be interesting to explore, for instance, how the tools to detect ambiguity directly in text [Chantree06, Yang11, Kiyavitskaya08, Gleich10], the conflict detection tools that use AORE with semantics-based composition [Weston09, and Sardinha09], a tool built to support Preview approach [Sommerville97a], and the works of Marcelloni, Akşit, and Noppen [Akşit01, Marcelloni01, Marcelloni04, Marcelloni04a, and Noppen07] on how to analyse decision alternatives, could be integrated in a more general framework that integrates imperfect information support to enable better decision support.

Another interesting approach to develop a broader framework to handle imperfection may well be in using the fuzzy logics/probability theory formalisms in “background”, for all the steps that identify, store, and handle information about imperfection. The advantage is that the “background” language is the same throughout the lifecycle. Noppen [Noppen07] work presents an approach that goes in this direction.

A critical issue, whatever the approach to achieve a broader framework, is how to present the output from the tools (that identify, store and handle the information about imperfection) in a communication mechanism suitable for the needs of imperfection management in RE and its users (as described in Section 1.2.2). Sometimes before the challenge of “how to present” there is yet to “make” the imperfection support tools

provide the information (about imperfection) that can be usable and useful for the task in hands and its users. The author of this thesis can report an experience to connect the EA-Analyzer [Sardinha09] with the communication mechanism proposed by the thesis. This was not a straightforward task, because what EA-Analyzer gives is not exactly what the tool to construct the communication mechanism needs. Thus, this had to be left for future work.

This chapter has surveyed the state of the art respecting the first objective of the approach: identify the most pertinent imperfections to be discussed during stakeholder consultation meetings.

From the issues described in this research agenda, this thesis addresses the issue of how to present the output from the tools to detect ambiguity and conflict, in a communication mechanism suitable for ambiguity and conflict analysis in RE (as described in Section 1.2.2). How this issue is addressed through the approach will be presented in Chapter 4.

The next chapter will explore the existing information visualization approaches in software engineering, to see how to best address the second objective: generate an effective communication mechanism that makes the ambiguity and conflict in key NFRs explicit, to be used in consultation with stakeholders.

# **3 Information visualization in software engineering**

## **3.1 Introduction**

This chapter explores information visualization techniques and their potential for visualizing imperfect information in software engineering and in particular in requirements engineering.

Visualizations are built from points, lines, areas and volumes. These primitives have various properties (like size, colour, shape, position). The graphical objects built from graphical primitives can have dynamics as the properties can change over time [Diehl07]. All these graphical object properties can be used to encode information. One interesting question that arises is: how can imperfect information be encoded and visualized in a manner that is useful for and supports software engineering, in particular decision support?

Section 3.2 will describe briefly the research areas that have explored visualizations for and related with text and sketches in software engineering with focus on analysis and design. This overview will describe some works that explored table and chart-based visualizations, hypertext-based visualizations and scenarios.

Section 3.3 is concerned with diagrammatic visual languages that have been built to support work in software engineering. The Sub-section 3.3.1 describes the usage of

visualizations for graphs and trees as a means to visualize some aspects of software engineering in particular decision processes.

Section 3.4 presents the work done on building visual metaphors to support software development.

Section 3.5 describes two works that follow a more general approach: one concerning imperfect information in general, the other only uncertain information. These approaches begin to study what sources of imperfection exist in information, and from that knowledge search what could be a “good” visualization for imperfect information. In this journey they also review some of the existing work pertinent for their approach.

Finally Section 3.6 points out a research agenda.

## **3.2 Text, sketches and its presentations**

Text consists of words. Words are sequences of characters from an alphabet and usually have meaning. Textual information can be augmented by visual cues (e.g. underlining, colour) to show additional information, highlight the important parts, or make the structure more explicit [Diehl07].

When facing the task of visualizing text descriptions or some of their characteristics (such as information about the person that created text) there are several approaches that can serve different purposes as described below.

### 3.2.1 Table-based and chart-based visualizations

One of the simpler types of visualization and quite common in software engineering is a line-based display as a table. Visual clues like colours are then used to help conveying information. An early example of such visualization can be seen in the SeeSoft tool [Eick92]. This tool was created to visualize statistics associated with lines in text files. The line-based visualization of SeeSoft maps each line of source code into a thin row, with files comprising the system arranged in columns across the screen. The colour of each row represents a value of the attribute that is being visualized, such as age or developer that authored it.

A good example of several sorts of table and chart-based visualizations can be seen in the work of Eick [Eick00]. This work demonstrates a series of possible visualization techniques, using the Advizor tool, which is an interactive environment for building tightly, linked visual query and analysis applications. The tool enables the presentation of data in a wide range of views from matrix displays, to 2D and 3D bar charts, pie charts and zoomable text displays. All views are interactive and linked together, so that selection in one view causes updates in all other views.

Focusing on requirements engineering tool support, the most common types of visualization lie in displays with several panes where the information is presented in textual line-based visualizations, including sometimes charts. Some examples are the commercial tool DOORS [DOORS] and EA-Miner [Sampaio07].

The work developed in visualization of multidimensional databases with focus on the techniques to select, extract and produce information from large multidimensional databases is also relevant for the effective support of software development. An interesting example of the work in visual analytics is Polaris [Stolte02]. Polaris is an interface for the exploration and analysis of large multidimensional databases. This

tool extends the Pivot Table interface to display relational query results using a rich, expressive set of graphical displays.

Table and chart-based visualizations are always good to present numeric information, like statistics or relative importance of characteristics. These techniques are not useful as the only means of visualization in the early phases of software development, which are mainly creative and exploratory. Visual analytics techniques like the ones that have been mainly developed for multidimensional databases can provide improvements in software development support.

### **3.2.2 Hypertext and hypermedia**

Text is usually presented in a linear fashion. This presentation is suitable if the reader wants to follow the author's idea and reasoning. Hypertext is a technique that attempts to provide alternative ways of browsing text (that may include charts, figures and other media). Hypertext structures text into a mesh rather than a line. Hyperlinks allow the user to access different blocks of text from the current one which should enable the user to follow his/her own path [Dix93].

One interesting example of hypertext use to support the early phase of requirements engineering is the work of Kaindl [Kaindl93]. He presents a tool that provides a mediating representation between the completely informal ideas of the user in the very beginning of the software development process and the more formal representation of domain models and requirements. The tool uses hypertext for this purpose, providing also links among requirements statements and the representation of objects in a domain model.

Hypertext and hypermedia have already some tradition to support the software development process, specifically in the area of design rationale. The pioneering example of this usage is the gIBIS system [Conklin88], which used hypertext techniques upon the famous IBIS (Issue-Based Information Systems) [Kunz70], an argumentative approach to design rationale. gIBIS (graphical IBIS) is a hypertext tool designed to facilitate the capture of early design deliberations.

### **3.2.3 Textual presentation and annotations**

Storey et al [Storey08] study how task annotations, embedded within source code play a role in the work practices of software developers. In particular, the study reports that annotations can be used to support a variety of activities fundamental to articulation work within software development. It also describes how task management is negotiated between the more formal issue tracking systems and the informal annotations that programmers write within their source code.

Ko et al [Ko08] present a new kind of debugging tool, the Whyline, that enables developers to select a question about program output from a set of “why did” and “why didn’t” questions derived from the program’s code and execution. Evaluations of the tool on one task showed that novice programmers with the Whyline were twice as fast as expert programmers without it. This tool does not allow to write a question but to select among several questions; it is any way an interaction where communication in natural language presented in text mode is an important part of interaction.

In fact, these works reflect that the more informal modes of conveying information through an informal way like text (which can be handwritten) or sketches should be



provided in development support tools as a complement to interaction and visualization. These more informal modes of expression enable to make explicit reasoning and concerns that do not fit into formal notations. Without these informal and flexible modes of expression reasoning remains implicit and is not communicated to other team members and stakeholders, which is particularly critical in some phases of software development like requirements engineering.

### **3.2.4 Sketches and scenarios**

Sketching is known to be a useful means to explore design ideas and alternatives [Preece94]. Sketches help in communication and facilitate visual brainstorming both by the individual and in teams. As sketching is informal and can be used by all types of professional profiles it is certainly useful for requirements engineering, when creative and explorative processes are more important.

Rich pictures [Checkland81] are a particular technique of sketchy drawings, developed by Checkland, as part of his Soft Systems Methodology for gathering information about a complex situation. They are used in problem solving and creative thinking methods in several areas of human activity.

Sketched-based interaction support has reached a relative maturity as the book by Buxton [Buxton07] shows. Mangano [Mangano08] also illustrates the research advances in this area.

A scenario is a personalized, fictional story with characters, events, products and environments [Preece94]. Scenarios help the designer to explore ideas and the

ramifications of design decisions. Typically, scenarios use text and sketches but can also use other information support modes or media. There is an extensive work on narrative and scenario-based presentations for rationale-based SE [Burge08]. It is interestingly to note that Use Cases from the Unified Modeling Language, UML [OMG\_UML] use a scenario based technique.

In fact, sketches and scenario techniques share with textual language its informal characteristic that enables to make explicit and communicate reasoning and concerns. They are thus a complementary mode of interaction.

### **3.3 Diagrams**

A diagram is a graphical representation where the geometric relations between its parts represent relations between the objects represented by those parts. These geometric relations include neighbourhood, linkage, containment, and overlapping. If done correctly, diagrams group relevant information together to make search more efficient, and use visual cues to make information more explicit [Diehl07]. Several arrangements of geometric shapes and geometric relations build a visual language with defined syntax and an associated semantics.

Since the early days of software development, diagrams have been used to represent the structure of programs. In 1946 and 1947, Goldstine and von Neumann introduced the flow diagrams, initially called control-flow graphs and later standardized and called flowcharts [Diehl07]. Other examples of diagrams used to depict the structure

of programs are: Nassi-Shneidermann diagrams [Nassi73], Jackson diagrams [Jackson75] and Control-Structure diagrams [Cross98].

With the development of software engineering, the need to build bigger software systems provoked the creation of notations that represent not only the structure of modules but also their behaviour and interactions. The most known of these is the Unified Modeling Language, UML [OMG\_UML]. UML is a standardized (general purpose modelling language. UML 2.0 has 13 different types of diagrams used to create abstract models of specific systems.

In the last fifteen years the predominant visualizations in the area of requirements engineering are either associated with UML diagrams or i\* goal models [Gotel08].

One example of notation associated with UML is the Systems Modeling Language, SysML [OMG\_SysML]. It is a general-purpose modelling language for systems engineering applications that is defined as a dialect of UML. SysML was built as a modelling language to specify systems that include non-software components and for this purpose is better suited than UML, which has a software-bias. It contains, among other diagrams, a Requirement diagram which shows system requirements and their relationships with other elements.

The i\* framework is an agent-based approach providing a visual diagrammatic modelling language [Yu97]. The i\* framework is an adaptation of NFRF framework, both of which are described in Section 2.5.2.

Glinz [Glinz00] investigated the suitability of UML as a semiformal requirements specification language and, identified and demonstrated several problems and deficiencies of UML.

Diagrams can also be used in software development for tasks other than the description of the software itself. Eppler [Eppler04] proposes three types of diagrams

(one of them uses a metaphor) for the transfer and creation of professional knowledge in organizational decision making contexts. A screenshot of two of these diagrammatic tools, the Synergy Map, and the Parameter Ruler is shown in Figures 3-1, and 3-2. Eppler claims that these types of interactive diagrammatic tools offer great potential for the improvement of (synchronous) knowledge communication, advising that future work should support not only tasks such as converging, evaluating, organizing or consensus building, but also facilitate criticizing, elaboration, and abstraction tasks [Eppler04]. These, in turn, can provide interesting tools to support the analysis phase and its relationships both with the design phase and the software management process.

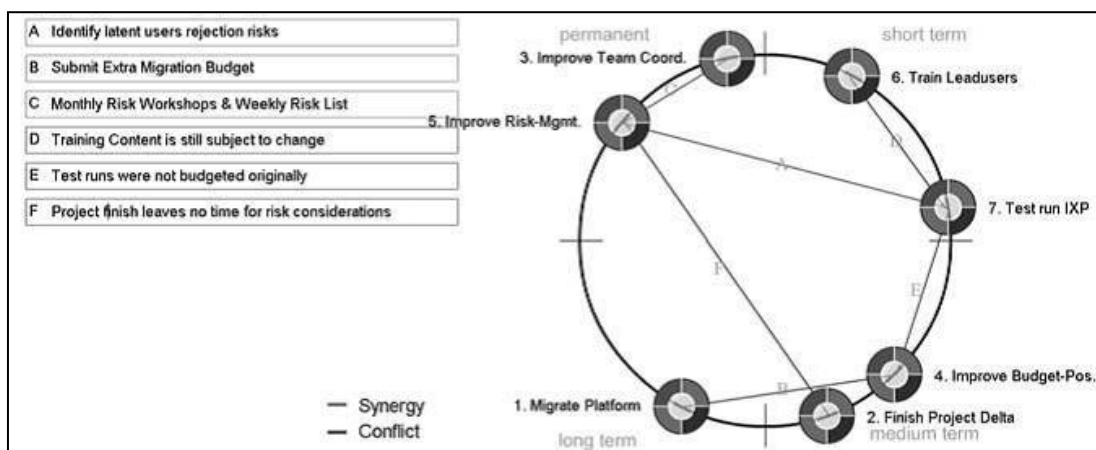


Figure 3-1 A screenshot of a synergy map [Eppler04].

Probably one of the most interesting characteristics of diagrammatical visualizations is the possibility to provide a sharable convention [Gotel08]. But the provision of a sharable convention also presents a problem. It is complex to define, syntactically and semantically, a universally applicable notation for describing software systems (as for example with UML) [Gotel08]. Field studies conducted by Bresciani et al [Bresciani08] confirm that diagram interpretation is an activity with high prerequisites, including learning. In fact, visualization's efficacy depends on the user's previous

experience and visual literacy. The size of UML language for example, which is perhaps unnecessarily large, contributes to this problem. Another aspect of diagrammatic languages that contributes to make interpretation more difficult is that they are attached to a particular methodology (e.g., UML for object-oriented development, flowcharts for structured development). In fact, the use of diagrammatic languages to describe software artefacts and development can present a barrier for communication and work among multidisciplinary user profiles, as well as, not being so suitable to the initial phases where the process is mainly creative and there is no organization yet.

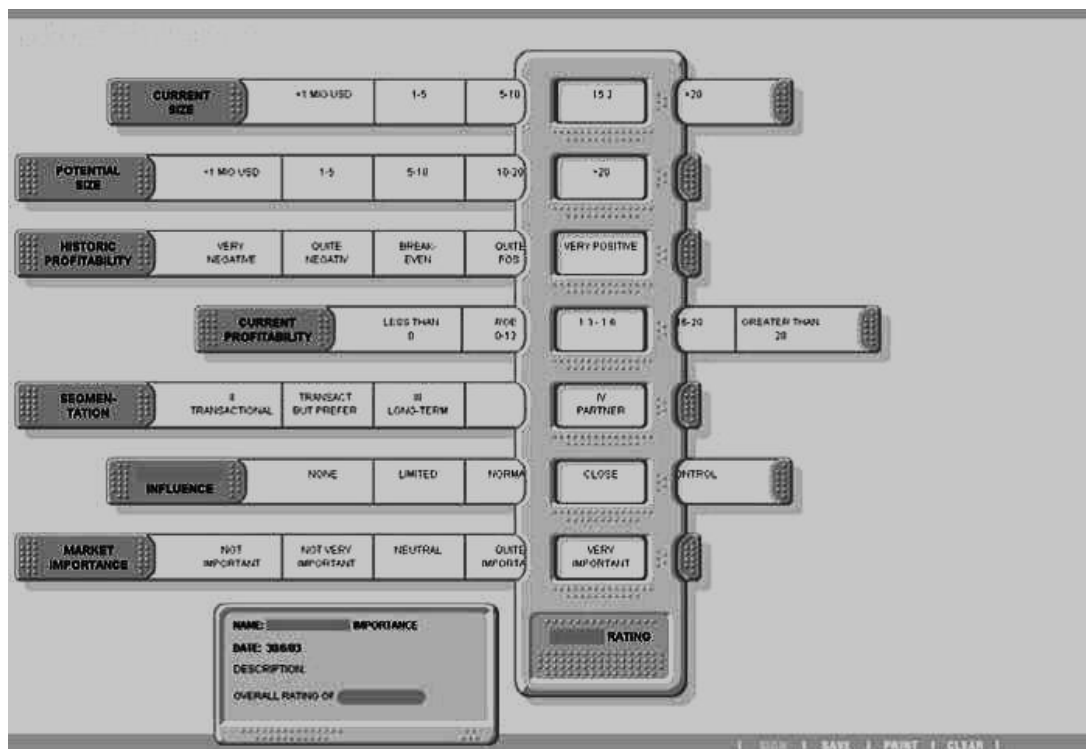


Figure 3-2 A screenshot of a Parameter Ruler session [Eppler04].

### 3.3.1 Graphs and trees

Graphs and trees (which are connected, directed graphs) are mathematical abstractions widely used to describe relationships between objects. The typical visualizations for graphs and trees (as mathematical abstractions) are node and edge diagrams, where each node is represented by a box, circle, or ellipse, while the edges are represented by lines. In fact, Goldstine and von Neumann flowcharts (see Section 3.3) are graphs. Other examples of graphs used in software engineering are Petri nets, syntax trees and finite-state diagrams.

Concerning the visualization of software development processes, graphs have been used to represent artefacts along the development life cycle and trees to represent design rationale. Noppen [Noppen07] also uses a directed graph (the Artifact Trace Model) where the nodes represent the intermediate design artefacts that result from the software development process. Noppen's Design Tree Model is a tracing model that creates a design tree (a tree structure) and uses it to describe the design issues that have been resolved, the order in which they have been addressed and the design alternatives that have been considered.

The use of node and edge diagrams is not suitable to represent artefacts and/or processes. Diagrams waste too much space if considered to represent a real life software system. Also being graph and trees a kind of diagram, they share the visualization problems already pointed for diagrams.

Screen-filling techniques have been developed to fit large hierarchies onto the screen [Diehl07]. These screen-filling techniques, instead of connecting nodes with lines, use geometrical relations between nodes such as containment or neighbourhood. Some examples are Treemaps [Johnson91], Information Pyramids (3D version of Treemaps) [Andrews97] and Information slices [Andrews98].

Recent Treemap algorithms by Bederson et al [Bederson02] have been applied by Feather et al [Feather06] to visualize requirements, risks and mitigations (what could be done to reduce the likelihood and/or impact of risks). The experience reported uses position, size and colour to convey interesting characteristics like grouping and hierarchy (through position), relative importance (through size) and requirement attainment status (through colour). This visualization is relevant, as it could be a fairly usable visualization for software artefacts in a development phase when the relation and organization of artefacts (requirements, design components) starts to become clearer. When such an organization is not yet known, as in the initial phases of analysis where developers and stakeholders are starting to build from nothing, trees and graphs are not the most appropriate, as their vocation is to represent relationship and thus organization. But it is also possible to think of an intermediate situation, when the development goes from artefacts that do not have any organization, to a situation when some organization starts to appear. The kind of requirements visualization that Feather et al [Feather06] produced with these recent Treemap algorithms can bring interesting insights in such cases.

### **3.4 Visual metaphors**

A visual metaphor is an analogy that underlies a graphical representation of an abstract entity or concept with the goal of transferring properties from the domain of the graphical representation to that of the abstract entity or concept [Diehl07]. Lakoff

and Johnson say: “The essence of metaphor is understanding and experience one kind of thing in terms of another” [Lakoff80].

Some of the visualization techniques, described in the previous sections, are based on visual metaphors. For instance, the diagrams are geometric-based metaphors and the tree (even the mathematical concept) is a metaphor. The Information Pyramids, which are a kind of 3D Treemap, can also be seen as a pyramid-based metaphor.

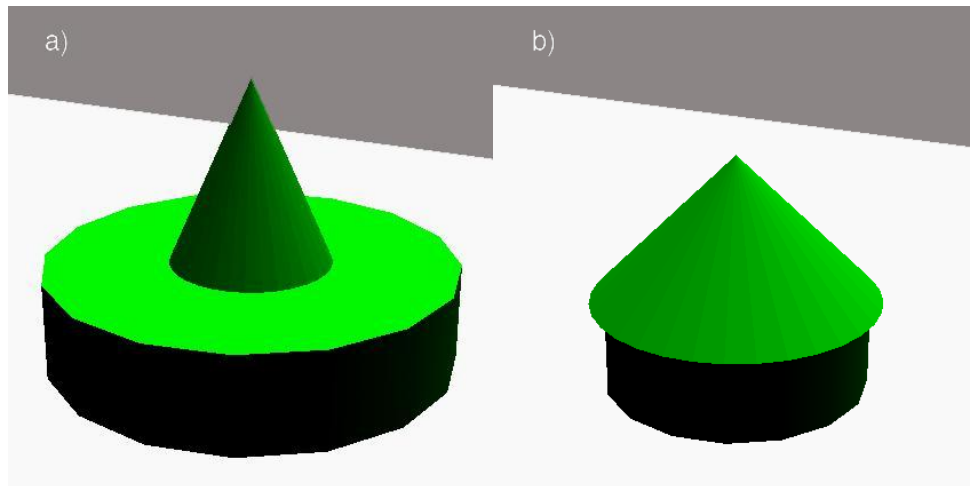
Some more sophisticated visual metaphors that have been recently developed to visualize software are described next.

One such example is the city metaphor, which was used to visualize software code by Knight et al [Knight00, Charters02], Panas et al [Panas03], and Wettel et al [Wettel07a, Wettel07b]. In the city metaphor, typically a building or a district represents an object-oriented class, and then visual characteristics are used to depict software characteristics and metrics. Another interesting metaphor is the landscape metaphor where landscapes are used to represent software systems [Balzer04].

Boccuzzo et al [Boccuzzo07] report on some experiences, with a software metrics configurator that handles different metaphors, and allows optimisation to their graphical representation. They report that, for metrics visualization, the usage of metaphor glyphs results in improved software comprehension compared to abstract graphical representations. Their approach presents two interesting aspects concerning interaction with the visualization: it enables the viewer to tag relevant elements for later analysis, enabling to quickly filter non-relevant elements out; and to use the interaction technique of walking through the views to analyse the software and in particular previous tagged elements. But probably, the most interesting idea is the usage of the concept of well-shaped graphical visualization to represent that the corresponding artefact is well designed. This shows that the metaphor “language” can be used to express information (quality aspects) about the artefacts that are



being represented. Figure 3-3 shows a house metaphor. Figure 3-3 a) shows a misshaped house and 3-3 b) a well-shaped house.



**Figure 3-3 House Metaphor showing a) a misshaped, and b) a well-shaped glyph [Boccuzzo07].**

Apprehension is a general problem in information visualization in which data of various degrees of abstraction, dimensions, degrees of freedom, and relatedness are correlated employing graphical means. According to Tversky [Tversky02], apprehension means: “structure and content of visualization should be readily perceived and comprehended”. Gotel [Gotel07, Gotel08] proposes the use of (good) metaphors to solve the apprehension problems that other types of visualization could not yet solve, when dealing with this type of data (containing various dimensions, degrees of abstraction, of freedom and relatedness). This thesis agrees with Gotel proposal, but takes the view that metaphors can also bring other advantages in the case of visualizations for software development.

One of the foundations, of this thesis, is the belief that the usage of (good) metaphors or combination of metaphors can provide good solutions, for the support of imperfect information in software development, in particular in requirements engineering. In

fact, metaphors may provide a good solution to address the needs of requirements engineering and its visualization. These characteristics were described both in Chapter 1 and in Section 3.1, and will be revisited now, focusing on the development of a communication mechanism.

In order to accommodate usage by different professional profiles that cooperate in RE, the metaphor developed should build on a well-known concept, to be easily used by the broadest professional profiles.

An important conclusion drawn from the readings in software visualization is that the more complex visual metaphors (more complex than two- or three-dimensional geometric ones) have been applied in their vast majority to artifacts and software that already exist (to show metrics, for program comprehension, for reverse engineering) and not to support software development [Diehl07]. This thesis is of the view that a well-assembled metaphor, making an analogy with a task where participants build something out of something, can provide an interesting tool for requirements engineering.

## **3.5 Visualizing characteristics of information in software engineering**

Gershon [Gershon98] in a short note entitled “Visualization of an Imperfect World” discusses several fundamental questions: what are the sources of imperfection in information, the degree of imperfection and how to represent it, intuitive visual

metaphors and cues for representing imperfection, and the imperfection in the presentation itself. Gershon says: “Life is not perfect. ... No two users are alike”; and explains the need to develop principles and methods of imperfection management. He proposes, as an initial approach, to present the degree of imperfection to the user as needed.

Skeels et al [Skeels08] performed a review of existing work from several domains to investigate what uncertainty is and how users explore it. They propose a classification of uncertainty representing commonalities in uncertainty across domains, in order to help develop appropriate visualizations for uncertainty.

### **3.6 Research agenda**

The general problem of software engineering visualization (SEV) is to find the best visual system that permits: to deploy information and data enabling tools to provide effective SE support; and, at the same time, enable usage by different types of professionals that cooperate in software development. There is no answer to what is the best information visualization (IV) technique. It depends on: “what for?”, and even when knowing for what, one can aim to get one good IV technique, usually not the best technique. All the techniques are good for something, and many times the best is to combine several techniques.

Concerning the application of information visualization techniques to support the specific aspect of imperfect information in RE, several interesting research questions arise:

- What types of imperfection or characteristics of information (meta-information, i.e. information on information, like imperfection, priority, and credibility) would be useful to represent in tools to support the management of imperfect information?
- What characteristics of RE activities would be useful to represent and promote?
- What are the modes and media usually used to convey information and meta-information in RE?
- How these visualizations can be integrated with other development support tools, without forgetting the issue of “visual” traceability for the other life-cycle phases?

Concerning the involvement of people in SE development, some more research questions arise:

- It would be relevant to study how people that are involved in development tasks represent (visually) imperfection and other characteristics such as credibility and relevancy.
  - Do people represent these different types of imperfection differently and if so how?
  - And does the way people represent imperfection vary by professional background?
  - Concerning decision making, do people represent or register decisions (in particular the decisions related with imperfect

information or other meta-level information) and its reasoning; and if so, how?

Also decision-making usually implies conflict, thus another question is:

- How do people usually interact with each other, and with the material used to work during the management of imperfect information?

Last, but not the least, the visualizations should be interactive, so it should be studied:

- What modes of interaction are most suitable and what visual analytics techniques are needed?

In this thesis one of the challenges is exactly to answer the question: what is the best visual technique(s) to deploy information and data (integrating what is called the communication mechanism) to provide effective handling of imperfection in requirements.

## **4 Proposed solution**

### **4.1 Introduction**

The previous chapters highlight that effective requirements engineering (RE) in the presence of conflict and ambiguity remains a major research problem and there is a lack of metaphors to aid communication during consultation with stakeholders.

This chapter proposes and describes an approach, based on a jigsaw puzzle metaphor to improve the identification, communication, and handling of conflict and ambiguity, inadvertently introduced during RE.

This approach offers heuristics to identify the most pertinent ambiguities and conflicts, which are worth discussing in a consultation meeting with stakeholders.

It also provides a jigsaw puzzle metaphor developed to make the identified conflicts and ambiguities explicit in key NFRs, during meetings with stakeholders. The chapter justifies why this metaphor constitutes a good instrument to promote and facilitate group consultation meetings with stakeholders, and explains the choice of a metaphor solution.

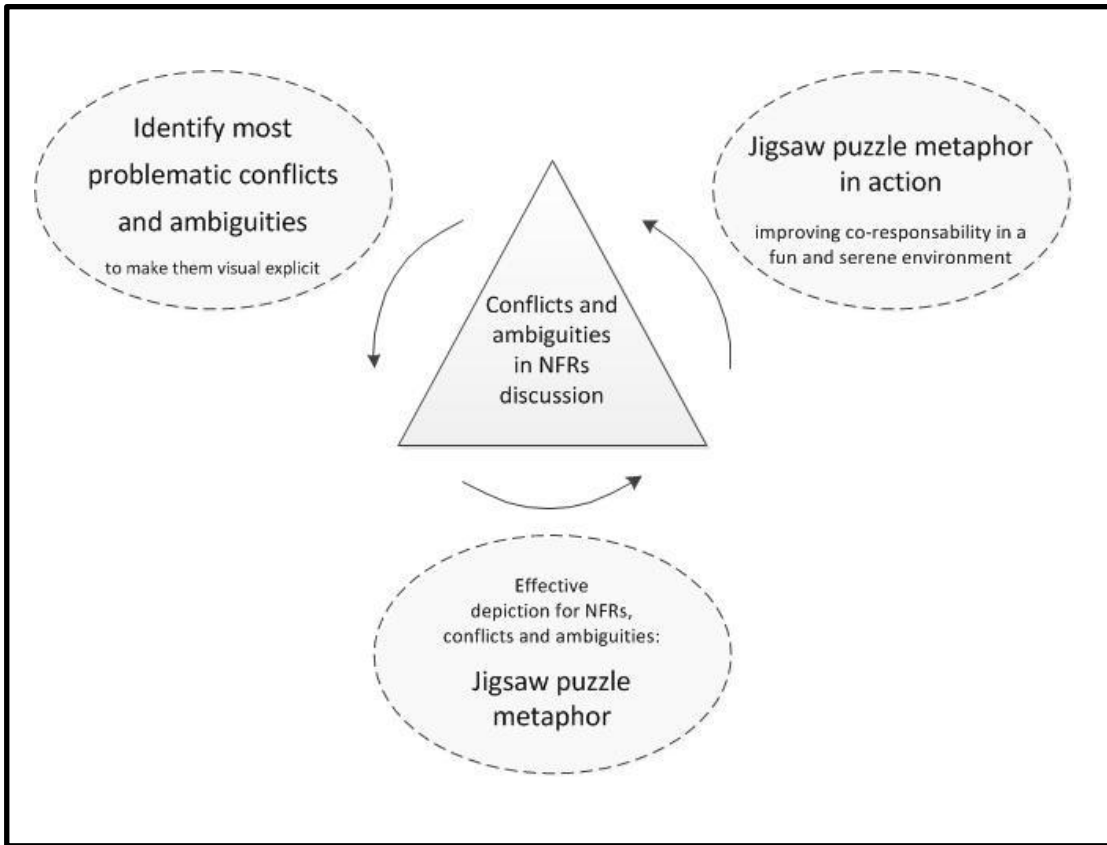
The chapter also presents a method for conducting the consultation meetings with stakeholders, and discusses tool support for the proposed approach.

## 4.2 Approach

In real-life system development, the requirements documentation may present thousands of individual requirements [Sommerville97, p. 9]. The approach assumes that the requirements are in natural language text, which, as seen before, is the most common form of presenting requirements documentation.

The three high-level activities of the approach are depicted in Figure 4.1. These are as follows:

1. Identify the most pertinent conflicts and associated ambiguities that ought to be made visually explicit. The approach proposes some heuristics and tools that can be used for conflict and ambiguity identification.
2. Generate an effective depiction for NFRs, the existing conflicts among them and associated ambiguities. This depiction uses a jigsaw puzzle metaphor. It is aimed at enabling discussion about conflicts and ambiguities, and promoting RE as an analytical/creative task, performed by a group of stakeholders.
3. A method for conducting the stakeholder meetings, bringing the jigsaw puzzle metaphor into action, promoting cooperation and co-responsibility towards the requirements document in a fun and relaxed environment.



**Figure 4-1 Diagram with the three high level activities of the approach.**

The approach is illustrated using the non-functional requirements (NFRs) from the Crisis Management Systems (CMS) requirements documentation [Kienzle09, pp. 8-10]. The documentation concerning these non-functional requirements can also be found in Appendix A. The requirements Reliability, Availability, Accuracy, and Real-time are shown in Figure 4-2.

- Reliability
  1. The system shall not exceed a maximum failure rate of 0.001%.
  2. The mobile units shall be able to communicate with other units on the crisis site and the control centre regardless of location, terrain and weather conditions.

**Figure 4-2 Text for Reliability as it is published [Kienzle09].**



- Availability
  1. The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
  2. The system shall recover in a maximum of 30 seconds upon failure.
  3. Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.
- Accuracy
  1. The system shall have access to map, terrain and weather data with a 99% accuracy.
  2. The system shall provide up-to-date information to rescue resources.
  3. The system shall record data upon receipt without modifications.
  4. The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.
- Real-time
  1. The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.
  2. The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
  3. The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.

**Figure 4-2 (cont.) Text for Availability, Accuracy, and Real-time as it is published [Kienzle09].**

## **4.3 Identification of the most problematic ambiguities and conflicts**

In order to identify the most problematic ambiguities and conflicts, worthy for discussion in a group consultation meeting, the approach proposes heuristics developed upon existing literature and our own insights.

These heuristics help to detect ambiguity and conflict in non-functional requirements (also called qualities). As the domain are non-functional requirements the heuristics advise the search for words (or word expressions) referring to qualities or breaking of qualities. The domain of qualities to be considered for such search can be taken from catalogues of qualities. Such catalogues have been developed in the NFR framework [Chung00], with continuing developments like the work of Cysneiros et al [Cysneiros03]. These catalogues may also be standards such as the standard ISO/IEC 25010:2011 [ISO11].

### **4.3.1 Ambiguity identification**

The heuristics proposed to identify ambiguity draw upon the definition of an unambiguous SRS<sup>15</sup> (discussed in Section 1.2.3), and the classification of ambiguity into: lexical, syntactic (also called structural), and semantic. This thesis adopts this classification and the definitions proposed by Berry et al [Berry03], and applies them to the domain of NFRs. The thesis presents this exercise for the higher level of the

---

<sup>15</sup> SRS is an acronym for Software Requirements Specification.

classification of ambiguity proposed by Berry et al [Berry03]. This exercise can be done downward the ambiguity classification tree presented by Berry et al. Such exercise was considered out of the scope of the thesis.

The definition of SRS of being unambiguous is defined as follows: «An SRS is **unambiguous** if, and only if, every requirement stated therein has only one interpretation» [IEEE-SA98].

### ***Lexical ambiguity heuristic***

Lexical ambiguity occurs at the level of words that may have different meanings. Berry et al define: «lexical ambiguity occurs when a word has several meanings» [Berry03].

### ***Textual description***

To find ambiguity in an SRS:

1. search for:
  - a. word expressions that belong to a catalogue of qualities (or their synonyms); e.g.: available, in operation, accurate;
  - or
  - b. word expressions meaning a violation of a quality (including antonyms of the qualities); e.g.: failure, downtime; and
2. check whether the word expression may have more than one meaning, enabling more than one interpretation.

Example: “Are control centre and system, referred in Real-time requirement, the same or different entities?”

The second phrase of the Real-time requirement defines that: “the delay in communication of information between control centre and rescue personnel...shall not exceed 500 milliseconds”; while the third phrase prescribes that “the system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.”

It is not clear if the expressions ‘control centre’, and ‘system’ refer to the same entity. There are at least two possible interpretations: one that ‘control centre’ and ‘system’ are the same entity; and the other that they are different entities.

The most frequent cases of lexical ambiguity are the ones where words (such as ‘control centre’ and ‘system’ as in the example above) may have more than one meaning, and in particular it is possible to interpret that the words of a pair refer (or not) to the same entity. Such pairs of words arise as problematic when used in the same requirement (as in the example above) or in two related requirements. This is in fact why ambiguity is quite often related with conflict.

### **Syntactic ambiguity heuristic**

Syntactic (or structural) ambiguity occurs at the level of sequence of words that can be given more than one grammatical structure, and each of the grammatical structures has a different meaning. Berry et al define: «syntactic ambiguity, also called structural ambiguity, occurs when a given sequence of words can be given more than one grammatical structure, and each has a different meaning» [Berry03].

### **Textual description**

To find ambiguity in an SRS:

1. search for:
  - a. sequence of words referring to a quality from a catalogue (or its synonyms); or
  - b. sequence of words referring to a violation of a quality from a catalogue (including antonyms of the qualities); and
2. check whether that sequence of words can be given more than one grammatical structure, each having a different meaning, enabling more than one interpretation on how that quality or break of quality has to be realized in the system.

*Example: “Is WAP a mobile user or an access channel for mobile users?”*

The above example is from the CAS system, which is a customer relationship management application [Ayed09]. It is the only example of heuristic that does not use the Crisis Management Systems (CMS) SRS and in particular Fig 4-2. In fact, we could not find syntactic ambiguity in the CMS requirements presented in Fig 4-2.

The CAS system SRS has the following phrase:

“SaaS applications offered through the Internet are usually supporting different interaction modes including classic page-oriented HTML GUIs, rich internet GUIs (e.g., AJAX) as well as access channels for mobile users such as WAP, data replication for offline use or speech control (e.g., to operate applications through a normal telephone)” [Ayed09].

This phrase presents situations that can be interpreted differently according to different grammatical structures.

From the part of phrase “as well as access channels for mobile users such as WAP” it is possible (if the person is not very knowledgeable in the domain) to interpret: a) WAP is an example of a mobile user; b) WAP is an example of an access channel.

It is possible to interpret that the phrase lists five “types of interaction modes”: a) classic page-oriented HTML GUIs, b) rich internet GUIs (e.g., AJAX), c) channels for mobile users such as WAP, d) data replication for offline use, e) data replication for speech control.

Or it is also plausible to interpret that the last three items listed are types of “access channels for mobile users”: a) WAP, b) data replication for offline use, and c) speech control.

### **Semantic ambiguity heuristic**

Semantic ambiguity occurs at the level of sentences, which may be read in more than one way within their context. Berry et al define: «semantic ambiguity occurs when a sentence has more than one way of reading it within its context although it contains no lexical or syntactic ambiguity» [Berry03].

The definition of context adopted is the one proposed by Berry et al: «The SRS context comprises the language context (i.e. the sentences before and after the sentence in which the quality word expression occurs) and the context beyond the language (i.e. the situation, the background knowledge, and expectations of the speaker or hearer and the writer or reader) » [Berry03].

### **Textual description**

To find ambiguity in an SRS:

1. search for:
  - a. word expressions that belong to a catalogue of qualities (or their synonyms); or

- b. word expressions meaning a violation of a quality (including antonyms of the qualities); and
2. check whether the sentences using these word expressions have more than one way of reading it within the SRS context, enabling more than one interpretation on how that quality or breaking of quality has to be realized in the system.

*Example: "Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?"*

The first sentences of the Availability and the Reliability requirements show an example of semantic ambiguity. Considering that each of these sentences is part of the context of the other; and considering also the background knowledge and the expectations of the readers, there are two ways of reading the following sentences.

From the Availability requirement:

"1. The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance",

and from the Reliability requirement:

"1. The system shall not exceed a maximum failure rate of 0.001%".

These two ways of reading are: 1) what is required in the second sentence concerning the 'maximum failure rate of the system' has to take into consideration the 'maximum downtime' allowed in the first sentence or; 2) these concepts are not related with each other. The ambiguity can be expressed with the question: should 'downtime for maintenance' be included in the situations considered for the calculation of 'failure rate', and thus the times allowed for 'downtime' have to be

accounted in the 'failure rate' value; or 'downtime ... for maintenance' is not to be accounted for 'failure rate'?

In fact this ambiguity case has to do with the semantics attached to the expressions 'downtime...for maintenance' and 'failure rate', which is dependent on the semantics of the sentences, the broader context of the SRS and the background knowledge and expectations of the readers.

### 4.3.2 Conflict identification

The heuristics proposed to identify conflict draw upon the definition of an internally consistent SRS (discussed in Section 1.2.3), and our own insights inspired in the existing RE approaches namely: Preview [Sommerville97a], and the NFR framework [Chung00].

This thesis has defined that:

An SRS is **internally consistent** if, and only if, no subset of individual requirements described in it define a feature of the system in an incompatible way.

#### **Synonymous/antonymous quality conflict heuristic**

When searching for pairs of NFRs<sup>16</sup> defining features (qualities, in this case) of the system in an incompatible way, one should search for two phrases in the SRS referring to the same quality. Picking a quality 'X' referred with the word expression

---

<sup>16</sup> For simplicity we use pairs, but the subsets can have higher cardinality.



'X1', other reference to the quality 'X' can be done using a synonym of 'X1'; or in a negative form, using an antonym of 'X1'.

Preview [Sommerville97a] uses the concept of "focus" to find probable conflicts. Viewpoints whose foci intersect are the most likely sources of conflict. In fact these viewpoints with intersecting foci are the ones that impose requirements on the same system components or features, and thus the ones where conflicts are more likely to appear. When for a quality 'X' referred with a word expression 'X1', an antonym or synonym (say 'X2') of 'X1' is found in another part of the SRS, these two parts of the SRS are describing the same quality (i.e. impose requirements on the same features) and thus they may conflict. In the Preview approach such a pair ('X1','X2') would belong to two viewpoints which have intersecting foci, and thus would be selected as probable conflicts.

### Textual description

To find a conflict situation in an SRS:

1. search two word expressions from a catalogue of qualities (or violation of qualities), which are synonym or antonym of each other, and
2. check whether their descriptions are incompatible with each other.

Example: "Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement."

The first phrase of the Real-time requirement and the second phrase of Accuracy requirement have the word expressions "update the following information" and "up-to-date information", respectively. The possible conflicting situation has to do with the fact that the Real-time requirement describes that "The control centre shall receive and update the following information ... at intervals not exceeding 30 seconds." Thus

it offers some detail about the “update of information”. But the Accuracy requirement just says: “The system shall provide up-to-date information...” The issue is: “are these requirements compatible?” and “How can they be made compatible?”

### **Actions operationalizing quality conflict heuristic**

In computer science and engineering it is known that an abstract concept such as a quality (e.g. Real-time) has to be operationalized in less abstract concepts usually described by actions (e.g. receive and update information). The description of the abstract concept and its operationalizations should be compatible.

In Preview it is sometimes helpful to consider the decomposition of a viewpoint into sub-viewpoints. This is advised in the presence of conflict among the requirements of a viewpoint, especially if the sources of the requirements have imperfectly matched foci. This heuristic (as well as the Quality dependency heuristic) aims to identify these cases of conflict among requirements of a viewpoint.

### **Textual description**

To find a conflict situation in an SRS:

1. search two word expressions of actions describing the operationalization of the same quality (from a quality catalogue), and
2. check whether these descriptions are incompatible with each other.

**Example: “Real-time description requiring times possibly incompatible.”**

In the Real-time requirement the first phrase says: “receive and update ... information ... at intervals not exceeding 30 seconds”. The second phrase says: “the delay in

communication of information ... shall not exceed 500 milliseconds.” And, the third says: “able to retrieve any stored information with a maximum delay of 500 milliseconds.” The word expressions underlined with only one line describe actions that need to be done to operationalize the Real-time requirement. It is pertinent to raise the questions: “Does communication of information (in the second phrase) require its retrieval (in the third phrase)? And if yes, are the values for maximum delay required in these phrases compatible? Another pertinent issue is to know if the maximum delays of 500ms allowed for “the delay in communication of information” (second phrase) and the ability to “retrieve any stored information” (third phrase) are already accounted for and compatible with the first phrase that demands “receive and update ... information ... at intervals not exceeding 30 seconds”.

In this case again it is questionable how to interpret correctly the interconnection of the actions describing the operationalization of the Real-time requirement (underlined with one line). There is again the presence of possible ambiguity in a situation of possible conflict.

### **Quality dependency conflict heuristic**

It is pertinent to organise system qualities in hierarchies, with more general qualities at the top and more specific ones at lower levels of the hierarchy. ‘Availability’ is certainly a quality belonging to the top level. If a system is not available it is useless to discuss other qualities. At a lower level of abstraction it is useful (although arguable on how to do it best) to organize, for instance, in a tree below ‘Performance’: ‘speed’, ‘efficiency’, ‘resource consumption’, ‘throughput’, ‘response time’. Such hierarchies can be found in standards (e.g. ISO/IEC 25010:2011 [ISO11]).

The NFR framework [Chung00] (and the developments on this work) proposes such hierarchies through catalogues that organise previously accumulated design knowledge. The NFR type catalogues organise the knowledge per type of NFRs (e.g. a catalogue about security, another about performance). In particular, NFR catalogues support the discovery of conflicts when there are qualities that require other qualities.

### Textual description

To find a conflict situation in an SRS:

1. search two word expressions from a catalogue of qualities (or violation of qualities), and one of these qualities requires the fulfillment of the other quality, and
2. check whether their descriptions are incompatible with each other.

### Example: "Real-time dependency on Availability incompatible."

The Real-time requirement has the word expressions: "receive and update ... information", "communication of information", and "retrieve any stored information". For these situations to happen, the system must be available, thus they are related to the following word expressions of Availability: "in operation except for a maximum downtime" and "recover ... upon failure". The possible conflicting situation resides in the following. If the system is allowed a downtime for 2 hours (as allowed by the first phrase of Availability requirement), how can it guarantee the Real-time requirement, i.e., "receive and update ... information ... at intervals not exceeding 30 seconds", "the delay in communication of information ... shall not exceed 500 milliseconds", and "retrieve any stored information with a maximum delay of 500 milliseconds". The same type of question can be posed in relation to the second phrase of Availability

requirement, which says “The system shall recover in a maximum of 30 seconds upon failure”. But, thus if it may be up to 30 seconds down how can the system guarantee what the Real-time requirement demands?

This type of heuristic happens frequently with qualities related with the Availability of the system (e.g. to be in operation) and other qualities, for instance communication of information.

### **4.3.3 Selection of ambiguities and conflicts to be discussed in stakeholder consultations**

The focus of this work is to provide a communication mechanism to help software engineers and stakeholders clarify, if possible, requirements text hampered by ambiguity and conflicts. This communication mechanism is to be used in stakeholder consultation meetings which are time-consuming to organise and perform, and are thus only worthwhile for highly pertinent situations. The issue is how to select these situations.

As surveyed in chapter 2, the output of ambiguity and conflict models and detection tools is not a two-valued logic classification of ambiguity and conflict (i.e. the output is not a list of 100% probability of ambiguity/conflict cases). For conflict identification this led to the proposal of classifications for the relationships between requirements that have more classes than just “in conflict” and “not in conflict”.

### *Selection of ambiguity cases*

Concerning ambiguity, the goal of a stakeholder consultation is to achieve a consensus, first if certain parts of the SRS are perceived as ambiguous. It may indeed happen that even if a part of the SRS is marked as potentially ambiguous a certain group of stakeholders don't find it ambiguous (or find a benefit to leave ambiguity). If the ambiguity is confirmed and deemed undesirable, the focus is on handling it, which can be to solve through rewording the text, or through more elicitation and analysis, etc.

Section 4.3.1 above proposes a set of heuristics to identify ambiguity. The development of a tool to "code" these heuristics is left for future work.

From the existing tools in ambiguity identification and handling surveyed (in Section 2.4), two sets of tools called attention because of their pertinence to the focus of this thesis, and added value that they would bring to the enterprise of building a tool to "code" the proposed heuristics. The strategies used in these two sets of tools would bring advantages if integrated into a tool "coding" the heuristics proposed. Also, in the absence of such a tool, the method to perform stakeholder consultations can be performed using the following sets of tools to identify and select the most pertinent ambiguities.

The work about nocuous and innocuous ambiguity [Chantree06, Yang11] described in Section 2.4 is very useful to reduce the number of ambiguity cases in consideration to the nocuous cases. As stakeholder meetings are time consuming the goal is to select the most pertinent cases. The most pertinent cases would be the more nocuous. The starting point of the tools [Chantree06, Yang11] is a dataset of ambiguous phrases from a requirements corpus, and associated human judgements about their interpretation. It is plausible that a refinement to the sub-domain of NFRs of the requirements corpus used and the human judgements about their

interpretation, can improve the identification of the ambiguities and classification into nocuous and innocuous (for the NFR domain).

Gleich et al's tool [Gleich10] provides reliable ambiguity detection, in the sense that it detects four times as many genuine ambiguities as an average human analyst. For every sentence that contains a detected ambiguity, the tool provides an explanation why the detection result represents a potential problem. This tool is very well suited to the aim pursued by this thesis, because of its reliability but also because it provides an explanation of why a certain part of the SRS has been marked as an ambiguity. As it is the responsibility of the analysts to select, from identified ambiguities, which ones are pertinent to discuss in a stakeholder consultation, such an explanation is helpful to judge the relevance of a certain detected ambiguity.

In the current state of the art in ambiguity identification tools, this thesis would advise the usage of:

1. [Yang11] classifier, which selects the most nocuous ambiguities, and/or
2. [Gleich10] tool, which selects the cases that have higher potentiality to be ambiguous.

For the same SRS, the results of both tools can be used as input for the analysts' selection of pertinent ambiguity cases for discussion in stakeholder meetings.

### **Selection of conflict cases**

Concerning conflict, the goal of a stakeholder consultation is to clarify if certain relationships between requirements are conflicts. If a conflict is confirmed it might also be a goal (depending on the analysts objectives for the meeting) to rewrite the SRS so that the conflict disappears or the conflict level is diminished. In fact, the most pertinent relations, between requirements, for clarification in group consultation

meetings are the ones where the requirements influence each other but the type of influence is not known: if it is positive, negative, or neutral; and in particular if it is probable that there may be a negative influence.

The definition of ambiguity as being a property of both the text and the interpretations held by a group of readers of the text [Chantree06, Yang11] enables to establish a connection between ambiguity and relations between requirements with unknown influence but with a potential to be a conflict. In such cases, the analysis by the stakeholders of the two pieces of the SRS to decide if there is a conflict promotes the clarification of the interpretation each person gives to the parts of the SRS in appreciation. This clarification of the interpretation given to each part of the SRS is what is needed to resolve the ambiguity, or at least to boost the discussion on the ambiguity itself. It can happen the other way round, i.e., when analysing ambiguity in two parts of the SRS, the resolution of the ambiguity may clarify if certain relations between the requirements in those two parts of the SRS are a conflict.

Section 4.3.2 above proposes a set of heuristics to identify conflict. The development of a tool to “code” these heuristics is considered future work.

From the existing tools and methods in conflict identification and handling surveyed (in Section 2.5), one method and a tool called attention because of their pertinence for the focus of this thesis, and added value that they would bring to the enterprise of building a tool to “code” the proposed conflict heuristics.

The concepts of viewpoint and focus, used in Sommerville and Sawyer [Sommerville97a] method and the search for viewpoints whose foci intersect, have as goal to find the sub-parts of the SRS that impose requirements on the same system components or features. These sub-parts of the SRS are exactly the most likely sources of conflict. The heuristics proposed have exactly the same goal of finding sub-parts of the SRS that impose requirements on the same system components or



features. This is in fact the goal pursued when searching for pairs of references to the same quality (either using synonymous or antonymous), when searching for abstract quality concepts and its operationalizations, as well as, when searching for qualities in dependency of other qualities. Thus, it is foreseen that when implementing Sommerville and Sawyer's method as well as the heuristics proposed in this thesis, both works may be combined to produce a useful tool.

In the absence of a tool to "code" the proposed conflict heuristics, the method to perform stakeholder consultations can be conducted using the following tool to identify and select the requirements with words that have higher probability to be in conflict, and that are thus pertinent to discuss in a stakeholder consultation.

In the current state of the art in conflict identification tools, this thesis would advise the usage of:

1. EA-Analyzer tool [Sardinha09] that lists the words with its associated probability (in decreasing order) of pertaining to a conflict between requirements.

It is important to notice that the final selection of the sets of requirements, to be used in a group consultation meeting, belongs to the team of requirements engineers in charge of the requirements engineering process. The heuristics, guidelines or tool results are a means to support the engineers' decision.

## 4.4 Jigsaw puzzle metaphor

### 4.4.1 Description

In the jigsaw puzzle metaphor each puzzle piece represents a requirement. When the requirement's text potentially leads to conflicts with other requirements, the respective puzzle pieces have a matching edge that almost fits but not perfectly. The background of each jigsaw puzzle piece contains part of a picture. When all the pieces of a puzzle are assembled correctly each piece contributes to a bigger complete image, just like in commonly used jigsaw puzzles. Figure 4-3 shows a picture of the four pieces of such a jigsaw puzzle representing Availability, Reliability, Real-time, and Accuracy NFRs of the CMS [Kienzle09]. The interlocking shapes between these pieces do not match perfectly, depicting conflicts between the requirements, represented by the pieces. For instance the bad fitting between the Availability piece and the Real-time piece represents the conflict *"Real-time dependency on Availability incompatible"* described in Section 4.3.2.

The text published in the original documentation for these requirements was previously shown in Figure 4-2 in Section 4.2. The text written in a piece representing a requirement is the same as in the requirements documentation. But some changes are introduced. To improve readability some unnecessary text is cut out, some words are abbreviated, the text is displayed in list mode, and upper case letters are used to stress the "topic" of each requirement.

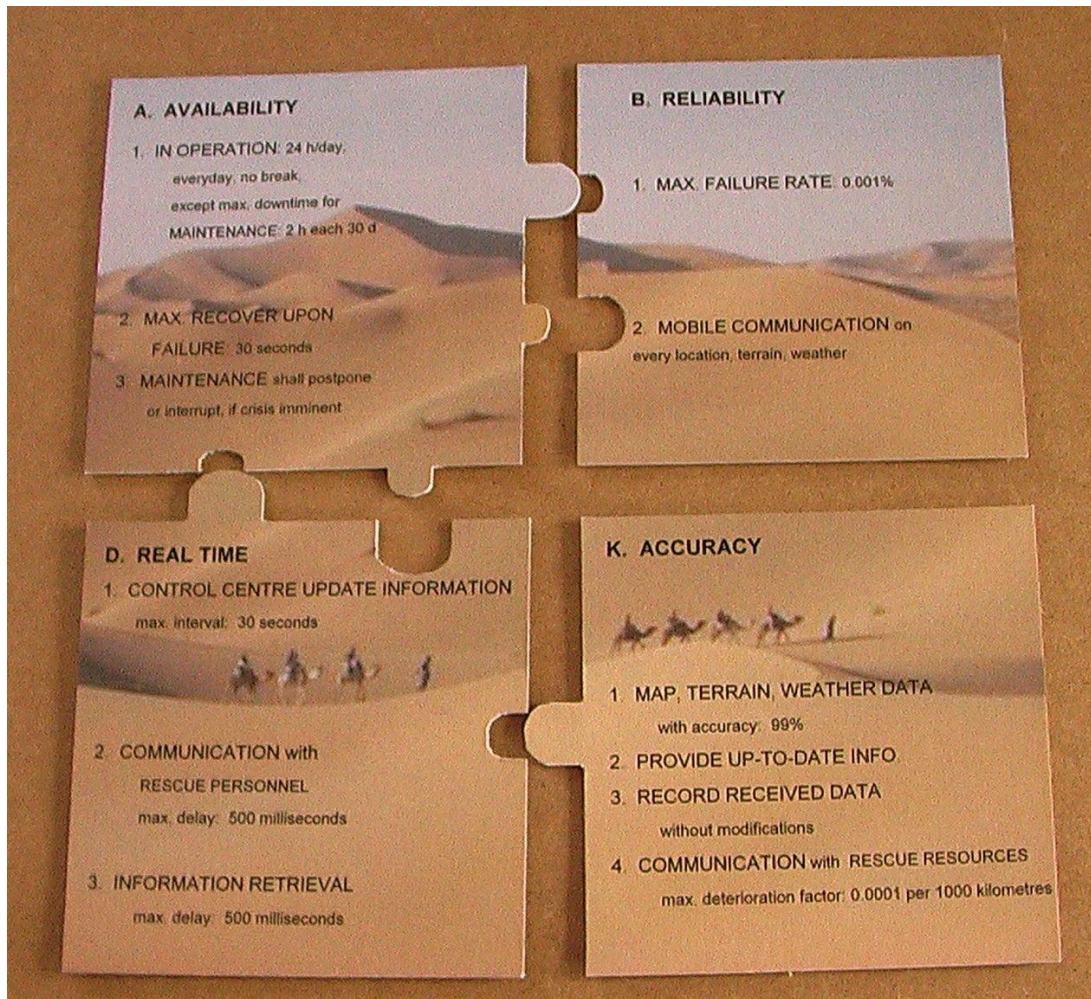


Figure 4-3 The jigsaw puzzle for the requirements Availability, Reliability, Real-time, and Accuracy of the CMS case study [Kienzie09].

The conflicts and ambiguities that occur among the text description of a requirement (represented by a jigsaw puzzle piece) do not have any visualization associated with them. An example of this is the conflict example *“Real-time description requiring times possibly incompatible”*, and the ambiguity example *“Are control centre and system, referred in Real-time requirement, the same or different entities?”* But, in the experiments the participants discovered these conflicts and ambiguities as described in Chapter 5.

In the first experiment (described in Section 5.4.1) using CMS, the fact that in Accuracy description nothing was said about the meaning of the word ‘up-to-date’,

leading to ambiguity on how the 'up-to-date' was to be interpreted, was visualized using a dotted border around the word 'up-to-date'. This will be referred as ambiguity case *"Nothing described concerning up-to-date in Accuracy requirement"*. After experiment 1 the hypothesis arose that it would not be very relevant to visualize this specific ambiguity, once together with the ambiguity there is also a conflict: *"Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement."* The ambiguity was rather seen as a sub-problem of the conflict. This hypothesis was confirmed in experiments 2 and 3 (described in Sections 5.4.2 and 5.4.3), where ambiguity in 'up-to-date' was not visually marked, and the participants discovered both the fact that nothing was said about how to interpret the word 'up-to-date', and the conflict *"Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement."*

The ambiguity example *"Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?"* was not marked because both Availability and Reliability requirements are not ambiguous when each is considered in isolation. The discussion of what interpretation should be considered (i.e. whether downtime should be accounted for failure rate, or not) becomes relevant, in face of the relation between Availability and Reliability. This relation would be the conflict: *"Downtime in Availability requirement and failure rate in Reliability requirement demanding incompatible values"*, if one considers that downtime for maintenance is to be accounted in failure rate. Availability text says: "The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance". If one takes the hours in a year,  $365 \cdot 24 = 8760$  hours and considers the system can be down for 2 hours every 30 days, i.e.,  $2 \cdot 12 = 24$ h, one achieves a failure

rate of 0.2740%. This value contradicts the Reliability requirement: “The system shall not exceed a maximum failure rate of 0.001%”.

#### **4.4.2 Rationale**

The approach presents a visual metaphor as an effective communication mechanism to make the identified conflicts and ambiguities explicit, during meetings with stakeholders organised for handling these imperfections. The quality of the communication mechanism depends on its adequacy to the task at hand in the consultation meetings and its wider context (i.e., RE), as well as its users (described in Section 1.2.2).

This approach employs a visual metaphor because metaphors may (if well defined) provide a good solution to address the needs of making conflicts in requirements explicit via visualization. A visual metaphor is an analogy that underlies a graphical representation of an abstract entity or concept with the goal of transferring properties from the domain of the graphical representation to that of the abstract entity or concept [Lakoff80, Diehl07].

As described in Chapter 3, the most common visualizations used in software engineering are diagrams (including graphs, trees). Diagrams use geometric shapes and geometric relations to build a visual language with defined syntax and an associated semantics. These visualizations present apprehension problems (i.e., the structure and content of visualization is not readily perceived and comprehended [Tversky02]). Thus, they need to be learned. With these characteristics, the use of diagrammatic languages to describe requirements and conflicts is not appropriate

when the users are a heterogeneous community with multidisciplinary (not necessarily computing) backgrounds. This becomes even more important when the goal is to improve co-responsibility and co-authoring of the requirements documentation. These facts lead to the need for a non-diagrammatic visual metaphor, one not relying on the definition of a defined syntax and an associated semantics to communicate, or where such syntax and semantics are intuitive (as it is the case of jigsaw puzzle metaphor).

The more complex visual metaphors (more complex than two- or three-dimensional geometric ones) have been applied in their vast majority to artifacts and software that already exist (to show metrics, for program comprehension, for reverse engineering) [Diehl07]. Using visualization to support requirements engineering, is a task with a strong creative facet, requires a different mind-set: to provide visualizations for artifacts that are being built.

The jigsaw puzzle game provides a good metaphor for requirements engineering, supported by the analogy of building an object, the jigsaw puzzle (in RE, the system), out from different pieces (in RE, the different requirements) that have to be correctly assembled to yield a final “correct” product, the background image (in RE, a system with “no” conflicts, or more correctly with as few conflicts as possible). This analogy emphasizes the creative facet of RE.

In order to accommodate usage by different professional profiles and promote cooperation and co-responsibility, a metaphor is needed that builds on a well-known concept to be easily used by the broadest professional profiles. The jigsaw puzzle, a game widely known and very easy to learn and play, provides a good metaphor for group work among persons with heterogeneous background.

The work of Boccuzzo et al [Boccuzzo07] (described in Section 3.4) proposing visual metaphors, that used the analogy of a badly-shaped visualization, to represent that

the corresponding artefact is badly designed, was an inspiration for the usage of badly fitting interlocking shapes representing badly fitting requirements (i.e. requirements with a conflict relation).

## **4.5 Jigsaw puzzle metaphor in action**

### **4.5.1 Description**

Together with the jigsaw puzzle metaphor, the approach proposes a method that uses the metaphor for conducting the consultation meetings with stakeholders. The challenge is to use the jigsaw puzzle metaphor in consultation meetings with requirements engineers and stakeholders to effectively promote cooperation, to perform analytical and creative work towards the requirements document, in a fun and relaxed environment.

The goal of the consultation meeting is to analyse conflicts and ambiguities in and among the requirements presented, the ultimate goal being to handle these imperfections, e.g. resolve or defer the resolution. The participants in the meeting will typically be engineer(s) responsible for requirements and appropriate stakeholders (will probably include but not limited to engineers involved in other development phases) chosen by the requirements engineer(s), according to the requirements, ambiguities, and conflicts under analysis. The requirements engineer(s) are the manager(s) of the meeting and also play the role of meeting facilitators.

In these meetings, typically requirements engineers want to focus on a small number of requirements (say 4 to 6, max. 9) and facilitate involvement of stakeholders.

The meeting begins with participants being asked to perform group work to assemble the jigsaw puzzle. This is arranged to be easy and quick. The goal is to “break the initial ice”, call the participants’ attention to focus on the same “object” and promote cooperation, and communication. The approach uses the cues commonly used in jigsaw puzzle pieces: pieces that have straight borders, and thus belong to corners and jigsaw borders; interlocking shapes that match pairs of pieces; and an image in the background, with each piece containing a part of it. The specific type of jigsaw puzzle used in this approach has another cue: the orientation of the text of the requirements.

While assembling the jigsaw puzzle participants discover that there is a way the pieces fit but not perfectly. The fact that the pieces don’t fit means that the way the requirements are written at the moment, contains conflicts (and possibly associated ambiguities) which do not allow such a system to be built.

Once the puzzle is assembled, the participants are asked to read the text in the pieces, and scan what could be the possible sources of conflict and ambiguity. Participants are encouraged to allow themselves to be questioned by the information the pieces convey, and to freely speak about any comments, and questions that came to their minds, as well as, to interact with other participants. Obviously, if the participants have difficulty in trying to identify the conflict and ambiguity cases, the managers of the meeting should inform what are the cases in discussion.

Upon the discovery of a possible conflict or ambiguity, the group is challenged by the managers and facilitators of the meeting to analyse and discuss that imperfection (including if it is really an imperfection) and achieve a consensus on how to handle that imperfection: it could be to require information that could resolve the



imperfection, it could be to reword the requirements text (for instance to remove ambiguity), or defer the resolution, etc.

## **4.5.2 Rationale**

The organization of meetings with stakeholders to inspect and/or negotiate problems in requirements (both conflicts and ambiguities, and the conflicts due to different stakeholders' interests) is a common and established practice in requirements engineering [Sommerville97].

The method proposed to conduct the stakeholders meetings, using the jigsaw puzzle metaphor, presents key features introduced with specific goals, namely to promote cooperation and team building, to improve communication and co-responsibility, as well as to induce creativity. All these goals are known to be favoured in fun and relaxed environments.

It is known that the organization of meetings to inspect and/or negotiate problems in requirements is cost-effective, and reduces the time required to an agreed set of requirements. But it is also known that problem resolution meetings should only concern the resolution of outstanding requirements problems [Sommerville97, pp.125-126, 195]. This is why the proposed approach prescribes the organisation of these group meetings when requirements engineers want to focus on a small number of requirements (say 4 to 6, max. 9).

An important aspect to respect in these meetings is the involvement of all (pertinent) stakeholders and analysts with different backgrounds [Sommerville97, pp.125-126,

195]. One of the key features introduced, in the organization of these meetings, is the choice of a visual representation that does not require computing knowledge (the jigsaw puzzle metaphor). This should improve communication, and thus co-responsibility and co-authoring.

Sommerville et al [Sommerville97] propose that these group meetings should be conducted in three stages: information stage, discussion stage, and resolution stage. This is in fact the schedule for the meetings that use the jigsaw puzzle metaphor with a difference. A key feature introduced is that, in the information stage, instead of informing the participants what the problems are and their nature, the approach proposes that, a priori, the participants scan for the conflicts and ambiguities using the cues provided by the non-fitting interlocking shapes. The introduction of a fun aspect is one of the reasons for this adaptation. As discussed, a fun and relaxed environment improves team cooperation and creativity. The other reason is to increase the sense that handling ambiguity and conflict in requirements require the cooperation of everyone involved. If the problems are discovered by participants in the meeting (in particular stakeholders), instead of being readout by the requirements engineers in charge, it will improve stakeholders' awareness that conflict and ambiguity in requirements are their problem too, and thus increase their commitment to cooperate in handling those imperfections.

The goals of promoting commitment and cooperation among (and amongst) stakeholders' and engineers' are pursued through two other key features. One is that participants are challenged to analyse and discuss the ambiguities and conflicts (including if they are really ambiguities and/or conflicts) and achieve a consensus on how to solve the imperfection. Another key aspect is that participants have to work with the same common "document", i.e. the jigsaw puzzle, and not each with his/her own copy of requirements text.

Sommerville et al. [Sommerville97, p.195] propose that in problem resolution meetings “there should be no blame attached to any problems discovered”. This approach is based on the same belief. Sommerville et al propose it as a measure to avoid confrontations between stakeholders. In fact, in these meetings, there are usually misunderstandings, and conflicts, which pitch people against each other. These misunderstandings and conflicts tend to transform working meetings into non-pleasant and boring ones.

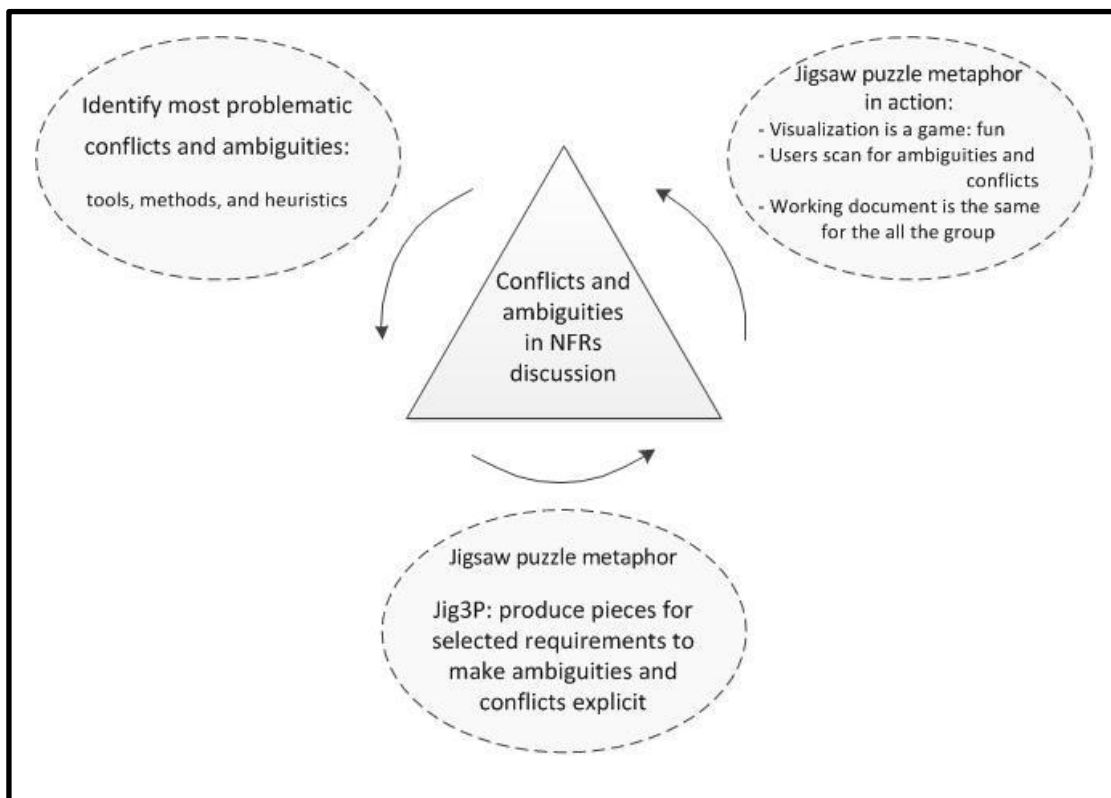
One more key feature of the proposed method is that being the jigsaw puzzle a game, it introduces fun in this work and fun has proven to increase creativity [Maiden07, Maiden08]. Having these group meetings perceived as fun and relaxed, can help ease the recruitment of stakeholders and engineers with different responsibilities, which is one of the problems often reported by those who have to organise them [Sommerville97, p.197].

Section 4.6, next, describes the synthesis of the approach, as well as, how the approach can be integrated in the wider context of ambiguity and conflict support in RE.

## **4.6 Synthesis of the approach**

This section presents a synthesis of this thesis approach to improve the identification, communication, and handling of ambiguity and conflict in requirements, inadvertently introduced during the RE process.

This approach receives requirements textual documents written in natural language, and integrates existing tools alongside heuristics to detect, classify, and prioritize imperfections, and a new tool to produce the jigsaw puzzle for a set of selected requirements. The approach is illustrated by Figure 4-4, which is similar to Figure 4-1, but now each of the ellipses of the three high level activities describe the activities listing the most important elements to realise them.



**Figure 4-4 Illustration of the realisation for the high level activities of the approach.**

Figure 4-5 illustrates the synthesis of the approach, showing its iterative nature.

As said, the approach receives requirements textual documents written in natural language. Other documentation collected and produced by the analysts, during elicitation and analysis, is also relevant for the approach.

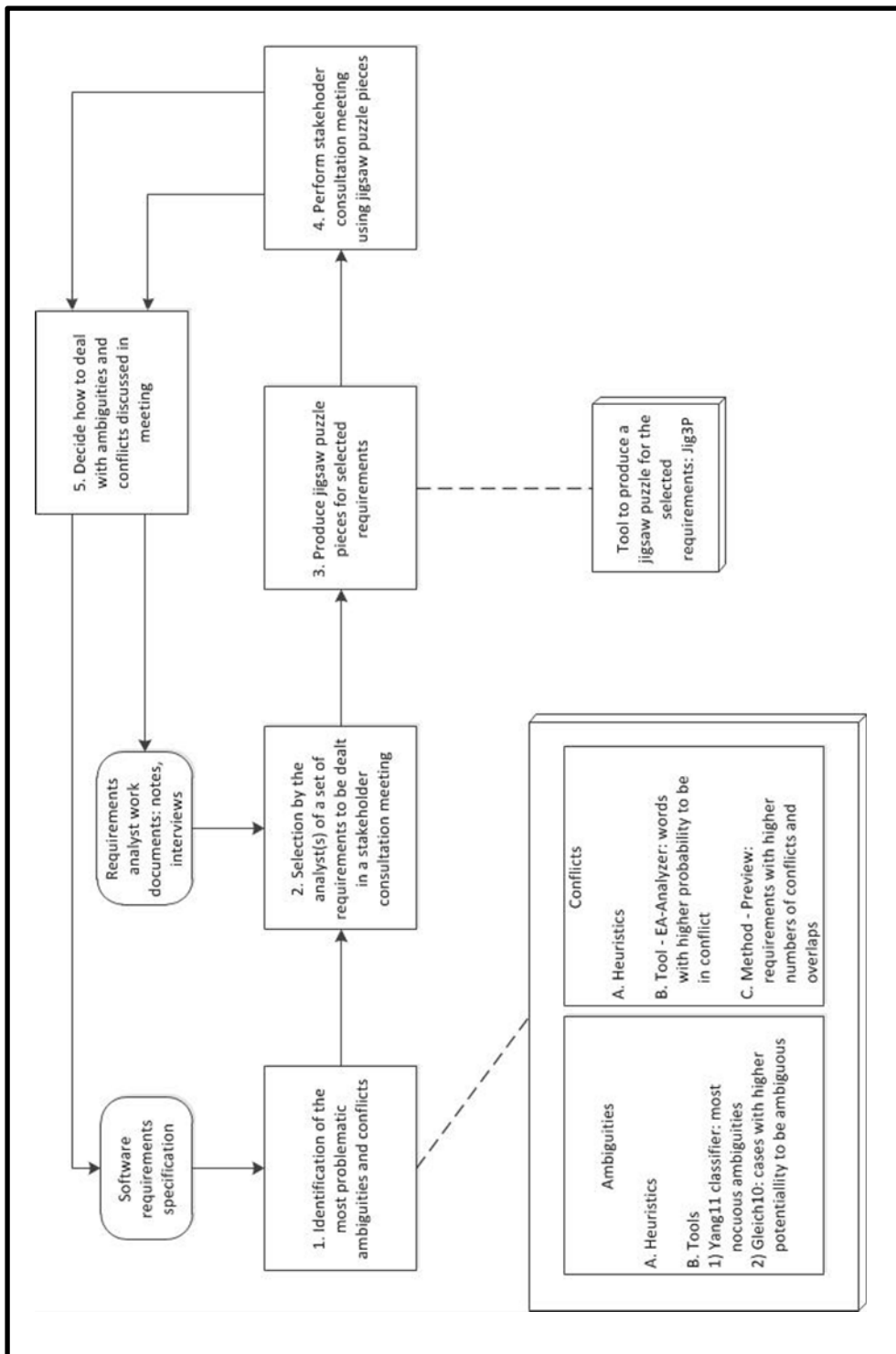


Figure 4-5 Illustration of the synthesis of the approach, showing its iterative nature.

The activities of the approach are:

1. Identification of the most problematic ambiguities and conflicts.

This thesis proposes, in Sections 4.3.1 and 4.3.2 a set of heuristics to identify

ambiguity and conflict. Section 4.3.3 offers a selection of the existing tools (described in Chapter 2) that best serve the goal of this thesis, presenting the analyst with a ranking of ambiguity and/or conflict cases. Section 4.3.3 also discusses how the proposed heuristics and existing tools and methods could be integrated to build a tool to “code” the proposed heuristics. Such a tool is left for future work.

2. Selection by the analyst(s) of a set of requirements to be dealt in a stakeholder consultation meeting. This decision is based on the output of one or several of the tools in task 1, taking into consideration project constraints such as time or budget constraints. The choice is entirely the responsibility of the analyst and s/he can, for instance, choose requirements which are not in the top lists of higher probability for conflict and/or ambiguity. It is possible and would be valuable to have tools that support the decision through the integration of the state-of-the art tools and methods that identify and rank conflict and/or ambiguity.
3. Produce the jigsaw puzzle pieces for the selected requirements. Jig3P is a (prototype) tool to produce a set of jigsaw puzzle pieces that represent a set of requirements, from the SRS, according to the jigsaw puzzle metaphor described in Section 4.4. Jig3P receives requirements documents in RDL (Requirements Definition Language), which are in fact XML documents. The prototype was developed in MATLAB [MATLAB] and uses a function to import XML documents. This function reads an XML file and returns the content of the file in a Document Object Model (DOM) node. Using MATLAB functions to deal with DOM it is possible to select just some parts of the RDL document, according to XML label contents. Thus, it is possible to extract the text of chosen requirements. MATLAB's geometric capabilities are then used to draw the puzzle pieces (at this moment just 4 and in a grid) with the respective requirement text inside. For the pairs of pieces (representing requirements)

that share a border and have conflicts (detected), the prototype draws interlocking shapes that do not fit well with each other. At this moment no treatment of the text of the requirements is done by the prototype. The code of Jig3P can be consulted in Appendix D.

4. Perform a consultation meeting with stakeholders using the jigsaw puzzle to promote analysis and handling of ambiguities and conflicts, according to the method proposed in Section 4.5. This method follows the tradition of stakeholder's meetings but is improved with key features aligned with the goals of the approach, which are:
  - a. The participants in the meeting have to assemble the jigsaw puzzle, which is itself a game (thus it is not necessary to begin the meeting with a game as it is commonly used in RE meetings) with the purpose of inducing a fun and serene environment.
  - b. It is the participants that have to first assemble the jigsaw puzzle: this is important because it is fun and improves relaxation; and secondly, to scan visually for the imperfections, instead of the imperfections being readout: this is important to improve the sense that the imperfections are everyone's problem and require cooperation and co-responsibility.
  - c. It is to work with the same common "document": the jigsaw puzzle, instead of each with his/her own copy: this is important to improve the sense that imperfections are problems common to all, and require cooperation and co-responsibility.
5. Decide how to deal with the ambiguities and conflicts discussed in the meeting. There are several decision alternatives, apart from resolution, of dealing with the identified ambiguities and conflicts. This decision process is a responsibility of the analysts. As this is not the focus of this thesis, we briefly describe the various options:

- a. Resolve; it may require more elicitation and/or analysis. The next Section describes existing approaches to ambiguity and conflict resolution.
- b. Work with imperfection making it explicit. This includes deferring the resolution of imperfection, and leaving the imperfection not seeking its resolution – this can be done using probability/fuzzy techniques described in Chapter 2 [Akşit01, Marcelloni01, Marcelloni04, Marcelloni04a, Noppen07].
- c. Work with imperfection not making it explicit. In fact, there might be cases/reasons when/why it may be useful to keep the imperfection implicit. For instance, a system provider may choose to leave, in a SRS, a system response time value ambiguous to allow flexibility and no commitment (enabling to reduce costs).

As Figure 4-5 depicts, this approach is iterative and the output from step 5 should be input into the SRS and/or the analysts' work documentation.

## **4.7 Resolution of ambiguity and conflict**

One of the hypotheses to deal with the ambiguities and conflicts, discussed in the stakeholders' consultation meetings is the resolution of such imperfections. This can be achieved:

- Through the input of information collected in the meeting, that eliminates the ambiguity or conflict. For instance, people in the meeting may agree to reword



or even change one or several requirements so that the imperfection(s) disappear.

- Through the help of resolution methods or tools. The next two Sections point some examples of tools and methods for ambiguity and conflict resolution in requirements. These methods and tools might provide suggestions to correct ambiguity and/or conflict cases. The decision of adopting a specific correction must be of the requirements engineer, taking into consideration the opinion of the stakeholders.

#### **4.7.1 Resolution of ambiguity in natural language SRS**

Resolution of ambiguity (disambiguation) and preventing (or avoiding) of ambiguity in natural language text are commonly seen as two faces of the same problem: the solution to the problem of ambiguity interfering with satisfactory natural language communication [Pool04].

Concerning ambiguity in software requirements specification (SRS) written in natural language, the preventing approaches can be categorised in [Berry08]:

- Write less ambiguously, avoiding those constructions that tend to create ambiguities. Berry et al [Berry03, Berry08] provide references for these approaches.
- Detect ambiguity
  - Manually [Kamsties01, Tjong07, Tjong08]. Kamsties et al work [Kamsties01] was already discussed in Section 2.4. Tjong et al work

- is briefly described below. Berry [Berry08] provides more references for these techniques.
- With the help of tools [Tjong08, Kiyavitskaya08, Gleich10]. Kiyavitskaya et al and Gleich et al tools were already discussed in Section 2.4. Again Berry [Berry08] provides more references for tools.
  - Use restricted or controlled languages for specifying requirements in an almost natural language [Fuchs99, Mitamura99, Mitamura01].

Surely it is always good the requirement analysts learn to write less ambiguously, but it is not possible to guarantee that a SRS text written by stakeholders respect the rules and guides to write less ambiguously. Restricted or controlled languages are not so natural and have not been adopted by requirements analysts in practice.

Concerning the disambiguation approaches, as described above, we share with Daniel Berry the conviction that, in the context of RE SRS, the best approach is not to build a tool that resolves ambiguity automatically, taking the requirements analyst off the process. Automation should stop with the suggestion of possible ways to resolve ambiguity. Anyhow in 1993, Kevin Ryan concluded that foreseeable future, AI approaches to language understanding do not work well for RE work [Ryan93].

Some of the tools categorized as tools to detect ambiguity also provide suggestions. Tjong et al work [Tjong07, Tjong08] present rules to guide a SRS writer in writing a less ambiguous and more precise SRSs, by analysing a few sets of requirements documents from different domains. The guiding rules can serve also as an inspection checklist that helps find ambiguities in SRSs. Tjong in his thesis [Tjong08] describes an experimental tool: Systemised Requirements Engineering Environment (SREE) that searches for requirement statements and offers suggestions for rewriting each potentially ambiguous requirement statement it finds. When SREE finds an instance

of potential ambiguity, SREE reports the instance to the user, so that the user can decide if the instance is truly ambiguous and to disambiguate the instance if desired.

Christophe et al [Christophe11, Christophe12] claim that computer tools can support the disambiguation process associated with requirements at elicitation and representation stages. They have developed an experimental process that uses the Recursive Object Model (ROM) [Zeng08] to clarify the syntactic ambiguities in requirements together with a protocol for semantic disambiguation.

Another hypothesis of solution for the ambiguity problem may be postponing resolution and working with ambiguity, such as described in the work of Noppen [Noppen07] in Chapter 2.

#### **4.7.2 Resolution of conflict in requirements**

Robinson et al [Robinson03] report the study of 29 methods for conflict resolution. They propose a categorization of these methods into six categories: 1) relaxation (includes generalization and value-range extension); 2) refinement (or specialization); 3) compromise; 4) restructuring (includes reinforcement and replanning); 5) postponement; and 6) abandonment. Postponement can be done using probability/fuzzy techniques as described in the previous Section 4.6.

Concerning methods that support conflict resolution WinWin [Boehm88, Boehm89] provide some manual support for all the above categories of conflict resolution. Quite often, conflict resolution approaches satisfy only a subset of all stakeholders, the “winners”. The WinWin tool offers software support for multistakeholder requirements

analysis and integrates such analysis into the software development life cycle. WinWin approach goal is to have a winning outcome for all stakeholders. Its activity model combines the risk reduction strategy of the spiral model [Boehm88] with the negotiation-oriented philosophy of Theory-W [Bohem89]. Concerning conflict resolution WinWin presents users, through QARCC (a component of WinWin), with a list of predefined text strategies that may apply to the given conflict. A user may use this information to define a resolution.

Deficiency-driven requirements analysis (DDRA) methods provide some automated support to conflict resolution. These methods rely on AI techniques to construct automated assistants that critique requirements as part of a deficiency-driven design activity. In DDRA violations of system requirements or environmental constraints drive the state-based search that generates alternative designs through the application of design operators [Fickas92]. Several prototypes of this kind of methods were built, being KATE [Fickas85] the first one. The tools Critic [Fickas88], Oz [Robinson93, Robinson94], and DealMaker [Robinson96, Robinson97] generate conflict resolutions. Critic recognizes some design fragments as bad (i.e., should be absent), and others as good (i.e., should be present). It can map design fragments to requirements patterns. Thus, when it recognizes a design fragment it suggests elements to remove or add in order to satisfy the specified requirements, and remove the recognized conflicts. Oz and DealMaker support many of the conflict resolution categories listed above. Oz and DealMaker support an interactive search procedure that (1) iteratively shows a multi-criteria evaluation of resolution alternatives, and (2) allows for the application of new resolution methods.

KAOS (Knowledge Acquisition in automated Specification of software) [Lamsweerde91] describes some requirements conflict resolution methods. These methods are not implemented, and thus an analyst has to apply them manually. A summary list of these methods is: avoiding boundary condition, restore goal,

anticipate conflict, weaken goal, instantiate resolution patterns, select alternative goal refinement, apply informal resolution heuristics, and refine object.

Other approaches that describe some support for conflict resolution (but not implemented) are ViewPoints and VIM [Easterbrook93, Easterbrook94, Easterbrook96], and NFR framework [Mylopoulos92, Chung00]. These approaches were already described in Chapter 2. In the ViewPoints approach an analyst may apply a resolution rule that is associated with a violated consistency rule [Easterbrook94]. Alternatives can be considered in the scope of a hierarchy of consistent ViewPoints [Easterbrook93]; however resolution alternatives are not explicitly represented. In the NFR framework when an analyst identifies a negative interaction, he can generate alternative resolutions and add them to the requirements OR nodes. An analyst can select a requirement as a resolution by marking it as selected.

The conflict resolution approaches discussed so far use domain models. Another hypothesis to provide conflict resolution is through tools based in argumentation structuring. Examples of such tools are gIBIS [Conklin88, Conklin89], and Synoptic [Easterbrook91a]. In Chapter 3 the gIBIS system was already mentioned because of its use of hypertext techniques upon the IBIS (Issue-Based Information Systems) [Kunz70], an argumentative approach to design rationale. gIBIS represents an argumentation process as an hypertextual graph, enabling to identify issues, identify positions that one can adapt with respect to the issues, and link arguments that can support or refute positions. Considering the issues may represent conflicts, such a system can be helpful in representing different solutions to solve a conflict, with the advantage of registering the different positions and arguments involved.

Easterbrook [Easterbrook91a] proposes a model of computer-supported negotiation which can be used to address conflicts in systems analysis. This model has been

used to develop the Synoptic system. This forms part of a larger model of requirements engineering based on the representation of multiple viewpoints, as described in Easterbrook thesis [Easterbrook91]. The model used to develop Synoptic consists of three phases: exploration of the participants' perspectives; the generation of suggestions for resolving the conflict, and the evaluation of these suggestions. During the exploration phase, the initial conflict is broken down into its components, represented as specific correspondences and differences between items in the viewpoint descriptions. These are annotated with comments describing any assumptions they make and issues they raise. These links and annotations act as a map of the conflict to guide the later stages. Resolution takes the form of designing novel ways of satisfying the issues. In the final phase, the ideas generated are then compared with one another and measured against the issues to determine the level of satisfaction. The option or combination of options, which best satisfies the issues, is chosen as a resolution.

## **4.8 Conclusion**

This Chapter describes and discusses the jigsaw puzzle metaphor as an effective communication mechanism to make explicit identified conflicts and ambiguities in NFRs, during stakeholders' consultation meetings. The proposal includes a method to conduct the meetings, using the communication mechanism to improve cooperation, co-responsibility towards the SRS, as well as promoting a fun and relaxing environment. The Chapter presents heuristics, tools and methods to identify

and select the most pertinent conflicts and ambiguities. The Chapter ends with a synthesis of the approach and an overview of conflict and ambiguity resolution techniques.

The next Chapter will describe and analyse the experiments performed to verify that the proposed approach does confirm the hypothesis of:

- Increasing effectiveness when compared with text presentation.
- Fostering team work and communication, and improving commitment of stakeholders in co-authoring of requirements and co-responsibility in resolution of ambiguity and conflict.
- Promoting a relaxed environment to improve team cooperation and creativity.

# 5 Evaluation

## 5.1 Introduction

This Chapter presents the evaluation of the jigsaw puzzle metaphor and its effectiveness in handling of requirements ambiguities and conflicts in stakeholders' consultation meetings, compared with the usual mode of performing this task with text.

The evaluation was done through three experiments, each one emulating the "real-life context" of a consultation meeting between requirements engineers and stakeholders. The set up used for the experiments is presented, as well, as the threats to validity, the experiments' results, and their analysis. The description of the experiments also shows how they were used to iteratively develop the approach.

The Chapter concludes with a synthesis of results and a discussion of the insights gained from the evaluation.

## 5.2 Goals

The goal of the evaluation was to test the following hypotheses, concerning the **jigsaw puzzle metaphor and the method for conducting the stakeholders'**



**consultation meetings**, when used to communicate and handle requirements ambiguities and conflicts in those meetings:

#### Hypothesis H1

- The **effectiveness** in **communication** and **handling** of requirements ambiguities and conflicts is **increased**, when compared with the same tasks performed with a text presentation.

#### Hypothesis H2

- **Team work** and **communication** are fostered, **improving** commitment of stakeholders in **co-authoring of requirements** and **co-responsibility in handling of ambiguity and conflict**.

#### Hypothesis H3

- A relaxed environment is **promoted**, and thus **team cooperation** and **creativity**, as the metaphor builds upon a game (the jigsaw puzzle).

## **5.3 Research methodology**

The ideal research methodology would have been a series of case studies, since the context is expected to play a role in meetings between engineers and stakeholders [Easterbrook07]. But, case studies [Yin02] would have required the use of the jigsaw

puzzle metaphor in real consultation meetings. As the access to real consultation meetings was not possible, the experiments emulating real-life meetings were chosen as the best possible evaluation method.

These experiments had a mixed philosophical stance: positivist and constructivist [Easterbrook07]. The experiments were confirmatory since they were used to test the hypotheses described in Section 5.2. The experiments were also exploratory because they were used to understand the capabilities and problems of the proposed metaphor and method of working with it, leading to improvements (and new hypotheses). The description of the results also shows how the earlier experiments results were used to iteratively develop the approach.

### **5.3.1 General design of the experiments**

The unit of analysis in the experiments was a small group (3 to 5) of participants.

The preparation of the experiments, as well as of a survey undertaken to establish the type of background image participants prefer for the jigsaw puzzle included the writing of a research protocol form and a consent form. The research protocol form and the consent form for the experiments are presented in Appendices E. and F., respectively.

Since the goal is to assess effectiveness and improvement of qualities (stressed with bold in the hypotheses in Section 5.2), such as communication, team work, commitment, and creativity, it is only appropriate to perform qualitative analysis.

During the course of the experiments some quantitative data was also collected, for instance, the time taken by the participants using the metaphor and method proposed in this thesis, as well as, the time taken by those who did not use the jigsaw puzzle and associated method. However, this data does not present statistically significant results given the small sample size of the qualitative experiments (sizes which are nevertheless acceptable for qualitative studies [Marshall96]).

The data collection techniques used were participant observation, mainly indirectly through audio and video record analysis (experiment 1 was only partially audio recorded); reports written by one of the group members with the descriptions of ambiguities and conflicts identified by the participants, a possible solution or other information required to handle the conflict or ambiguity; and questionnaires (audio and video recorded) to inquire preference between working with the proposed approach versus the usual textual presentation of requirements documentation.

During the experiment sessions the participants were invited to brainstorm and think-aloud. This enabled to collect richer data about aspects like participants' threads of reasoning, modes of interaction among participants, the tasks being performed, and the ways in which the puzzle pieces were used.

## **5.4 Experiments description**

This section presents the three experiments, describing the set-up, results, and threats to validity, for each.

## 5.4.1 Experiment 1

### 5.4.1.1 Set up

As the first experiment with the jigsaw puzzle metaphor there was a particular exploratory focus on testing the use of a jigsaw puzzle set of pieces, to understand which features work well, and which ones could be improved and how.

The session emulated the “real-life context” of a consultation meeting between requirements engineers and stakeholders aimed at reviewing a specific part of the requirements documentation for the Crisis Management Systems (CMS) [Kienzle09]<sup>17</sup>, presented using the jigsaw puzzle metaphor. In this meeting the investigators performed the role of facilitators and engineers in charge of the requirements documentation, while participants played the role of stakeholders and engineers from other areas.

In experiment 1 the “emulated” meeting was attended by five participants with requirements engineering background and two researchers acting as facilitators. There was only one session of 2 hours.

The investigators described the system briefly. The participants worked with four requirements of the CMS: Availability, Reliability, Real-time, and Accuracy. These were represented in a jigsaw puzzle set composed of four cardboard pieces with size 10 cm x 10 cm. The background of the pieces had no image, and the pieces did not match perfectly due to the presence of badly fitting interlocking shapes, representing conflicts. The fact that in Accuracy requirement nothing was said about the meaning

---

<sup>17</sup> Appendix A presents all the non-functional requirements of the CMS example.

of the word 'up-to-date', leading to ambiguity on how the 'up-to-date' was to be interpreted, was visualized using a dotted border around the word 'up-to-date'. This ambiguity case will be referred to as "Nothing described concerning up-to-date in Accuracy requirement" in the Table 5-1, listing the ambiguity and conflict cases with a visual cue in the jigsaw puzzle.

The participants were asked to work together to assemble the jigsaw puzzle. When doing this they discovered that there is a way the pieces fit but not perfectly. Participants were then informed that this means that there are conflicts and ambiguities in that part of requirements documentation. At this moment participants were asked to read the text in the pieces, and scan what could be the possible sources of conflict and ambiguity. The participants were encouraged to allow themselves to be questioned by any information that could seem problematic, and speak freely.

Each time a participant raised a possible conflict or ambiguity, the whole group was asked to brainstorm and search for an agreement on whether if that possible conflict/ambiguity was indeed a conflict/ambiguity, a description of the imperfection according to their interpretation, and if possible a proposal on how to remedy it. After achieving a consensus, one participant hand-wrote a work report with the results of the brainstorm about that possible ambiguity or conflict.

The pieces were designed to promote identification and analysis, of the conflicts (with ID 1, 2 and 4) and ambiguity (with ID 3) in the Table 5-1:

ID	Name	Description
1	Real-time dependency on Availability incompatible	See Section 4.3.2
2	Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement	See Section 4.3.2
3	Nothing described concerning up-to-date in Accuracy requirement	Described above and in Section 4.4.1
4	Downtime in Availability requirement and failure rate in Reliability requirement demanding incompatible values	Described below and in Section 4.4.1

**Table 5-1 Conflicts and ambiguities with a visual cue in the jigsaw puzzle for the CMS system in experiment 1.**

The possibility of a conflict in the relationship between Availability and Reliability was already discussed in Section 4.4.1. It was decided to represent in the jigsaw puzzle, the possible conflict between Availability and Reliability requirements to bring forward the discussion of the relation between these requirements and the associated ambiguity. ‘Downtime’ in Availability requirement and ‘failure rate’ in Reliability requirement demand incompatible values, only if it is interpreted that ‘downtime for maintenance’ has to be accounted for in the ‘failure rate’. The ambiguity (if ‘downtime for maintenance’ has, or not to be accounted for ‘failure rate’) was not marked because both Availability and Reliability requirements are not ambiguous when each is considered in isolation.

Figure 5-1 shows a picture of the cardboard puzzle set after being used in the experiment.



Figure 5-1 The cardboard jigsaw puzzle for part of CMS, after being used in the first experiment.

#### 5.4.1.2 Description of experiment and results

##### Hypothesis H1 - Increase effectiveness when compared with text presentation

##### 1. Number of imperfections detected

In this experiment participants **discovered all the four imperfections** listed in Table 5-1 (see previous Section 5.4.1.1), and for which there were cues through badly fitting interlocking shapes.

The participants detected **four other imperfections**, for which there were no cues (through badly fitting interlocking shapes). Appendix H in Section H.1.1 lists these imperfections.

## *2. Preferences between jigsaw puzzle and text presentations*

When the participants were asked to evaluate the jigsaw puzzle compared to the usual SRS text presentation, to identify and analyse the ambiguities and conflicts, they concluded that “the jigsaw puzzle-based presentation is really good in helping to identify problems and conflicts”.

Hypothesis H2 - Foster team work and communication, improving commitment of stakeholders in co-authoring of requirements and co-responsibility in handling of ambiguities and conflict and

Hypothesis H3 - Promote a relaxed environment, and thus team cooperation and creativity, once the metaphor builds upon a game (the jigsaw puzzle)

Hypotheses H2 and H3 can be observed in the following steps of the meeting when the jigsaw puzzle metaphor was put in action:

### *1. Assembling the jigsaw puzzle*

When trying to build the puzzle, the participants started to propose to others what tactic to use, like: “make first the corners” and we could perceive they were exploring how to work in group. After this initial phase, participants collaborated as a team, not having problems in posing their questions or making comments, and saying things like: “wait a minute...” and then explaining their reasoning and offering their comments. The participants were handling the pieces, even taking them up off the table, and showing them to others. They used the direction of the text written on the pieces as another visual cue (in addition to the interlocking shapes) that helped to understand how the pieces should go together.



## *2. Scanning for conflicts/ambiguities*

During the phase of scanning for conflicts and ambiguities, the participants kept working as a team, and when faced with a possible conflict/ambiguity, participants discussed among themselves. After achieving a consensus, one participant hand-wrote the group conclusion on the piece using numbers and/or letters to refer to the different pieces and phrases inside a piece. In some conflict/ambiguity cases the participants proposed a common remedy for the imperfection.

### Exploratory perspective - use insights from an earlier experiment to inform the design of the subsequent experiment

The participants suggested the following improvements, concerning the jigsaw puzzle pieces:

- Have the possibility to identify, univocally (with letters, numbers,..), each requirement and each phrase in a piece, so that it can be referred to in comments anywhere in the puzzle;
- Make the pieces bigger, so that there is more free space for hand-written comments; and
- Reinforce the jigsaw puzzle metaphor common cues such as the use of colour on the surface of the puzzle pieces.

### 5.4.1.3 Threats to validity

#### Constructivist view

The evaluation used in this work is qualitative and “perhaps more” constructivist than positivist, Easterbrook et al express what was experienced, concerning the assessment of its validity: “In the constructivist stance, assessing validity is more complex. Many researchers who adopt this stance believe that the whole concept of validity is too positivist, and does not accurately reflect the nature of qualitative research. That is, as the constructivist stance assumes that reality is ‘multiple and constructed’, then repeatability is simply not possible. Assessment of validity requires a level of objectivity that is not possible” [Easterbrook07].

Anyhow, frameworks were developed to evaluate the contribution of constructivist research. Creswell [Creswell02] identifies eight strategies for improving validity of constructivist research. From this list the following three were selected as they are applicable (and were possible to apply) to our research methodology:

1. Clarify bias<sup>18</sup>: be honest with respect to the biases brought by the researchers to the study, and use this self-reflection when reporting findings.
2. Rich, detailed descriptions: to convey the setting and findings of the research.
3. Report (also) discrepant information.

#### 1. Clarify bias

The following situations may have introduced biases brought by the researchers:

- During the preparation of the experiment, when formatting the text in the pieces (cutting some words, introducing abbreviations, etc.).

---

<sup>18</sup> Clarify bias is called reliability, under the positivist view [Easterbrook07].

- During the experiment sessions, as the investigators also acted as facilitators, there was the possibility, even when trying to avoid it, that they conducted the participants to the solution. As experiment 1 was conducted with two researchers this probability was reduced, compared to the presence of only one facilitator in experiments 2 and 3.
- When reporting and analysing the results, which this thesis text makes every effort to avoid. Strategies 2 and 3, prescribed by Creswell, may also help in reducing this type of researchers' bias.

## 2. Rich, detailed descriptions and 3. Report discrepant information

When reporting the experiments results, this thesis aims to provide rich descriptions with as much detail as possible. It will include, should it be the case, data that contradicts the hypotheses/propositions.

### **Positivist view**

#### 1. Construct validity

With regard to construct validity [Easterbrook07], it was already acknowledged in Section 5.3 that the best methodology to evaluate this approach would have been a case study, but that this was not possible. Anyhow the experiments performed as emulations of real meetings between stakeholders and engineers provided a reasonable approximation of the context of a real meeting, and it is plausible that results obtained would be confirmed in real meetings.

## 5.4.2 Experiment 2

### 5.4.2.1 Set up

This experiment introduced a control group, asking the participants to analyse two different systems: one with a jigsaw puzzle presentation, the other with textual documentation, in each session. Experiment 2 comprised 3 sessions, with 3 different systems being worked by 3 different groups of participants. In each session the meeting performed with the textual documentation was used as a control group for the session where the same system was worked with the jigsaw puzzle presentation.

The three requirements system specification documents used were: the Crisis Management System (CMS) [Kienzle09] (the requirements Availability, Reliability, Real-time, and Accuracy); the Health-Watcher system (HW), which is a web-based system that manages health-related complaints [Soares06] (the requirements Availability, Performance, Security, and Standards); and the CAS system, which is a customer relationship management application [Ayed09] (the requirements Availability, Security, Multi-channel access, and Accurate and Up-to-date information)<sup>19</sup>.

Each session, as in experiment 1, emulated the “real-life context” of a consultation meeting between requirements engineers and stakeholders to review a specific part of the requirements documentation. Each session lasted 2 hours, and was attended by a group of three to four participants and one researcher, acting as facilitator. In

---

<sup>19</sup> Appendices A, B and C present the text versions of the subset of requirements used in the experiments from the CMS, the Health-Watcher, and the CAS system, respectively. In each session, the system presented as a jigsaw puzzle, was accompanied by a piece of text explaining the main goal of the system, and in the case of CAS containing also a description of the abbreviations. These introductory descriptions are presented in the appendices A, B, and C.

every session at least one participant was an RE expert, while the others were engineers specialised in issues ranging from computing, telecommunications, networks, and electronics. Among these others there was at least one with no RE knowledge to emulate the role of external stakeholder. The other participants played the role of stakeholders, which could be internal or external. Care was taken to organize the groups in a balanced way, achieving equilibrium between experts and novices in the area. The organization of sessions, and the number of participants per session was as Table 5-2 shows:

	<b>Jigsaw puzzle presentation</b>	<b>Text presentation</b>	<b>Number of participants</b>
<b>Session 1</b>	CMS	HW	4
<b>Session 2</b>	HW	CAS	4
<b>Session 3</b>	CAS	CMS	3

**Table 5-2 Systems, type of presentation and number of participants in each session of experiment 2.**

Figures 5-2, 5-3, and 5-4 show pictures of the jigsaw puzzles used in experiment 2, representing the CMS, the Health-Watcher (HW), and the CAS system. The jigsaw puzzle sets were designed to be assembled in a 2x2 grid to enable the adding of a background picture. All had the desert picture with text printed on a paper that was then glued to the support.

The following improvements were made to the design of the jigsaw puzzle metaphor:

- Concerning the suggestions made by participants in experiment 1 (see Section 5.4.1.2):
  - The pieces were larger: 12 cm x 12 cm (for session 1 and 2), and in session 3, two pieces measured 12 cm x 20 cm and the other two 12 cm x 12 cm.

- The pieces were labelled for reference in the discussion and report.
- The pieces had an image in the background to reinforce the jigsaw puzzle metaphor.



Figure 5-2 The cardboard jigsaw puzzle for part of CMS, used in experiment 2.

- Test the hypothesis: it is not relevant to have a visualization (the dotted border) to mark an ambiguity that is associated with a conflict.
  - An example of this situation is the ambiguity; “Nothing described concerning up-to-date in Accuracy requirement”, and the relation

between Real-time and Accuracy: “Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement”. To decide if this relation is a conflict it is necessary to decide how to interpret the word ‘up-to-date’. The ambiguity was rather seen as a sub-problem of the conflict.

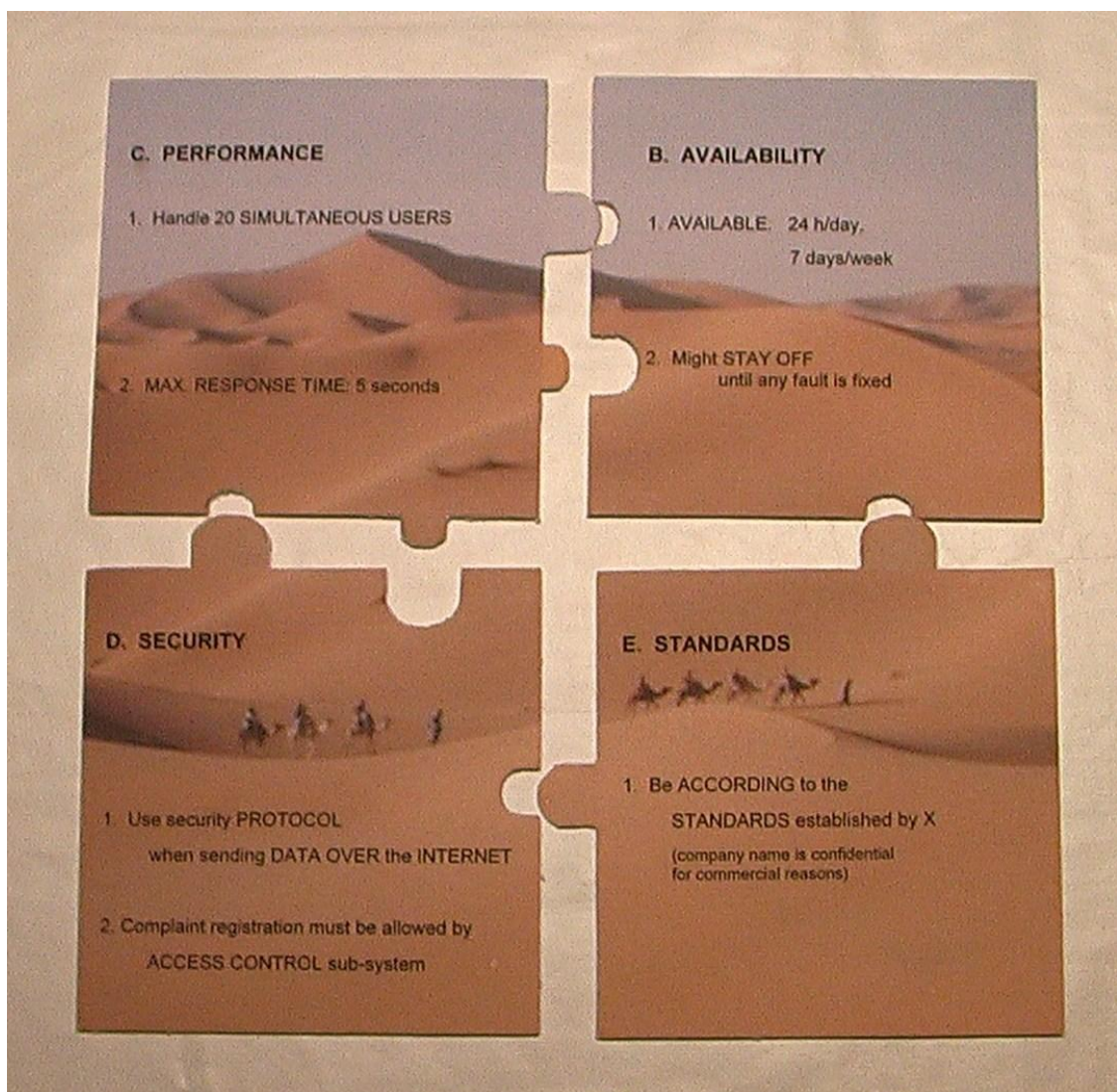


Figure 5-3 The K-line jigsaw puzzle for part of Health-Watcher, used in experiment 2.

- One other advantage to not visually mark ambiguity was to unclutter the presentation. The design decisions about the jigsaw puzzle

metaphor constantly strived to provide what was thought to be necessary cues but no more than that.

- In the CAS example, shown in Figure 5-4, rectangular pieces were used so that one piece positioned in the row below could be connected with the two upper pieces.
  - In the cases when a requirement has possible conflicts with three other requirements, the layout with 2x2 cells all with the same size does not allow placing interlocking shapes for one of the possible conflicts.
  - In experiment 3 this design was also applied to the Health-Watcher example.

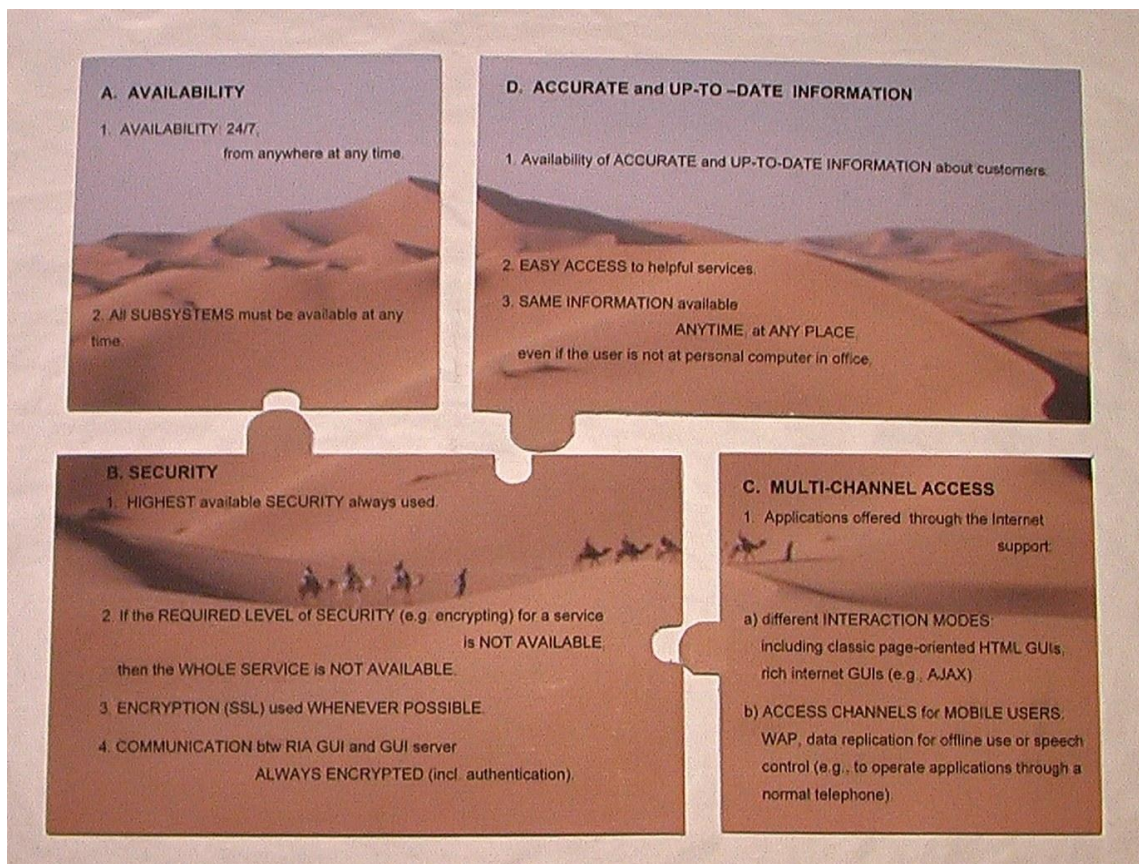


Figure 5-4 The K-line jigsaw puzzle for part of CAS example, used in experiment 2.



In each session, the investigators briefly described the first system, and presented a jigsaw puzzle set representing a part of that first system. The meeting continued, with the assemblage of the jigsaw puzzle and analysis of the first system, exactly in the same way as it was described in Section 5.4.1.1 for the first experiment.

After scanning and reporting all the possible conflicts/ambiguities for the first system, the session continued with the presentation of part of the requirements for a second system. This system requirements documentation was presented in textual form. Concerning this second system, the participants were also asked to perform the same job of scanning for the possible conflicts/ambiguities, brainstorm about them, and write a report after achieving a consensus on the possible imperfections.

After working with a subset of the requirements for two systems, the participants were asked which one they preferred to work with. They were also encouraged to provide whatever comments and suggestions for improvements they thought might be useful. The sessions ended with a brainstorming to discuss the possibility and eventual preference of using the jigsaw puzzle metaphor with an interactive digital support, with collaborative functionalities, instead of the presented physical (cardboard or K-line) pieces.

### 5.4.2.2 Description of experiment and results

#### Hypothesis H1 - Increase effectiveness when compared with text presentation

##### 1. Number of imperfections detected

Tables 5-3, 5-4, and 5-5 show how many conflicts, with visual cues, were totally or partially discovered by the participants in session 1, 2 and 3, respectively. The tables also show the number of other imperfections (ambiguities and conflicts), for which there were no visual cues (through badly fitting interlocking shapes) but also detected by participants. Appendix H lists these imperfections.

	<b>N conflicts with visual cue</b>	<b>N conflicts with visual cue detected</b>	<b>N imperfections with no visual cue detected</b>
<b>CMS jigsaw puzzle</b>	3	1 – totally 2 -partially	4
<b>HW text</b>	NA/4 <sup>20</sup>	1	2

**Table 5-3 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 1.**

In session 1, from the four imperfections, with no visual cue, detected in the CMS jigsaw puzzle, two were the following ambiguities:

- *“Nothing described concerning up-to-date in Accuracy requirement”.*
- *“Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?”.*

---

<sup>20</sup> In the HW system presented in text, no imperfection had a visual cue. The number 4 represents the number of imperfections with cue in the HW presented in jigsaw puzzle, for comparison reasons. In the next column, the number of imperfections detected, with visual cue, refers to the number detected in the set of 4 that present a cue in the jigsaw puzzle.

	N conflicts with visual cue	N conflicts with visual cue detected	N imperfections with no visual cue detected
HW jigsaw puzzle	4 <sup>21</sup>	2	3
CAS text	NA/3 <sup>22</sup>	2	6

**Table 5-4 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 2.**

	N conflicts with visual cue	N conflicts with visual cue detected	N imperfections with no visual cue detected
CAS jigsaw puzzle	3	1 – talked (0 - written <sup>23</sup> )	6
CMS text	NA/3 <sup>24</sup>	0	4

**Table 5-5 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 2, session 3.**

## *2. Preferences between jigsaw puzzle and text presentations*

Participants in session 1 unanimously preferred to work with the jigsaw puzzle presentation, rather than with the text.

In session 2, the first reaction when asked about their preference between working with the jigsaw puzzle presentation, rather than with the text, was that this comparison was unfair. They justified it by saying that the text in the jigsaw puzzle had a treatment that makes it much better and quicker to work: it is formatted making

<sup>21</sup> One of the 4 conflicts marked with a visual cue in the HW, has a visual cue with “poor” quality as it is explained in Appendix H.

<sup>22</sup> The number 3 represents the number of imperfections with cue in the CAS presented in jigsaw puzzle, for comparison reasons.

<sup>23</sup> In general, the analysis only considered as detected, the imperfections written in the report. In this case no imperfection was written in the report, but the participants talked about one.

<sup>24</sup> The number 3 represents the number of imperfections with cue in the CMS presented in jigsaw puzzle, for comparison reasons.

it clear, important words are marked, and garbage text is deleted. Anyway, the unanimously conclusion was that it was better to work with the jigsaw puzzle than with the text. One participant suggested it could be fairer to compare the jigsaw puzzle metaphor with templates that are sometimes used for requirements.

In session 3, participants' first reaction was that CAS is not a good example to compare the jigsaw puzzle presentation with a text presentation, as the puzzle did not work well for the CAS example. The three participants acknowledge that they did not work with CAS example as a puzzle but as four pieces of text separately. From the three participants, one expressed his preference to work with the text<sup>25</sup>. Another one's opinion was that it was the same, but stressed that this example was not fair to assess this comparison. He pointed out that in general if there are few requirements he would prefer the text and to be able to write, draw arrows, move. The researcher pointed out that this could also be achieved with the puzzle in an interactive digital table, which was well accepted. A third participant sees this kind of idea to be more useful for a more technical-oriented tool (to be used by analysts, not customers), like a dashboard and "there may be a puzzle"<sup>26</sup>.

---

<sup>25</sup> This participant reviewed his preference towards the jigsaw puzzle presentation, when interviewed about image background preference.

<sup>26</sup> This participant reviewed his preference towards the jigsaw puzzle presentation, when interviewed about image background preference.

Hypothesis H2 - Foster team work and communication, improving commitment of stakeholders in co-authoring of requirements and co-responsibility in handling of ambiguities and conflict and

Hypothesis H3 - Promote a relaxed environment, and thus team cooperation and creativity, once the metaphor builds upon a game (the jigsaw puzzle)

Hypotheses H2 and H3 can be observed in the following steps of sessions when the jigsaw puzzle metaphor was put in action:

*1. Assembling the jigsaw puzzle*

In session 1, when trying to assemble the CMS jigsaw puzzle (Figure 5-2), the participants first tried to set them according to the capital letter used to identify each concern. Thus in the beginning they thought there were pieces missing. Afterwards they followed the jigsaw puzzle cues of interlocking shapes and the picture in the background.

In session 2, while one participant quickly assembled the HW jigsaw puzzle (Figure 5-3) using the picture in the background as a cue; another one commented that the picture gave the wrong idea on how to set the jigsaw, because the pieces did not fit perfectly when assembled using the picture cue. Then participants set the pieces apart and tried to find a different organization for the pieces according to its textual contents. At this moment, the researcher explained that the pieces are to be assembled following the picture cue; and the fact that the interlocking shapes do not fit well was designed on purpose. After this, participants turned their attention to scan for the conflicts/ambiguities.

In session 3, when trying to assemble the CAS jigsaw puzzle (Figure 5-4), participants queried if they needed to read the text. The researcher reminded them

that this was a jigsaw puzzle and asked what cues participants usually use to assemble a jigsaw puzzle. Then the group quickly made the jigsaw puzzle. After discovering that the pieces did not fit well, the researcher explained why and asked the group to work as a team in scanning for conflicts and ambiguities. From this moment participants started to raise possible conflicts and ambiguities and discussing about these.

## *2. Scanning for conflicts/ambiguities (both in jigsaw puzzle and text)*

In session 1, once the CMS jigsaw puzzle was assembled one participant raised, quite instantly, a possible conflict and the group started to discuss it. In all sessions when working with the jigsaw puzzle, participants scanned the conflicts/ambiguities through reading the text in the jigsaw puzzle pieces, and collaborated as a team, searching for a consensus on what should be the conflicts/ambiguities to report. This scanning for conflicts/ambiguities went on with a good team work atmosphere.

In session 1, but now working with Health-Watcher in text, there was, in the beginning quite a long period when each participant read his/her piece of paper, and almost everyone underlined or made comments on the text. Only after a call from the researcher/facilitator the participants started to speak out about the possible conflicts/ambiguities they found.

In session 2 and 3, the meeting events and atmosphere were similar to what happened in session 1. Each participant read in silence his/her own documentation, but in these sessions none of them underlined or made any comments on the text.

Exploratory perspective - use insights from an earlier experiment to inform the design of the subsequent experiment

At the beginning of experiment 2 the following hypothesis was raised:

- It is not relevant to have a visualization (the dotted border) to mark an ambiguity that is associated with a conflict.

This hypothesis was confirmed in experiment 2 where ambiguity in 'up-to-date' was not visually marked, and the participants discovered both the conflict *"Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement"* and the ambiguity case: *"Nothing described concerning up-to-date in Accuracy requirement"*.

Between the second and third experiment, a survey was undertaken to establish the type of background image participants prefer for the jigsaw puzzle. This survey was performed between experiment 2 and experiment 3 with the participants of experiment 2. During the survey some participants asked an explanation on some aspects they had not fully understand during their participation in experiment 2. Then, they spontaneously re-formulated their comments and opinions on the issues discussed during their session of experiment 2. The description of the survey and its results are in Appendix G. These results were implemented and evaluated in experiment 3.

It was not part of the plans for this experimental evaluation to use the constructivist research strategy of prolonged contact with participants, i.e., "to make sure that exposure to the subject population is long enough to ensure a reasonable understanding of the issues and phenomenon under study" [Creswell02]. But the

performance of the survey provided a second contact with all the participants in experiment 2. Some participants wished to offer, during the survey, more suggestions after checking their initial interpretation of the jigsaw puzzle tool was not correct. In particular, the participant who initially thought the jigsaw puzzle would only be useful for a more technical tool could now understand its usefulness to work with stakeholders. This participant proposed the jigsaw puzzle background could use the colours of the customers company.

#### **5.4.2.3 Threats to validity**

##### **Constructivist view**

Concerning the Creswell framework [Creswell02] strategies for improving validity of constructivist research, the possible introduced bias are described below.

##### 1. Clarify bias

When designing the jigsaw puzzle piece for the CAS requirement Multi-channel access, the investigator formatted this requirement's phrase, assuming implicitly one of the possible interpretations for the grammatical structure of the phrase that has syntactic ambiguity (as explained in section 4.3.1). When doing this, the investigator might have introduced a bias. The original phrase in the CAS SRS text is:

“SaaS applications offered through the Internet are usually supporting different interaction modes including classic page-oriented HTML GUIs, rich internet GUIs (e.g., AJAX) as well as access channels for mobile users such as WAP, data



replication for offline use or speech control (e.g., to operate applications through a normal telephone)” [Ayed09].

The investigator formatted it, using a list style (as it was done for all the other requirements) in the following way:

“C. MULTI-CHANNEL ACCESS

1. Applications offered through the Internet support:

a) different INTERACTION MODES:

including classic page-oriented HTML GUIs,

rich internet GUIs (e.g., AJAX)

b) ACCESS CHANNELS for MOBILE USERS:

WAP, data replication for offline use or speech

control (e.g., to operate applications through a

normal telephone).”

One threat to validity that was more noticed in experiment 2, session 3 was the tiredness of the researcher, which might have introduced some bias. This tiredness did not enable the researcher to perform as a facilitator with the same quality in all sessions. One example of this was the fact that in experiment 2, session 3, the researcher was not able to spot the conflict detected and spoken by the participants and ask the participants to include it in the report.

## 2. Rich, detailed descriptions and 3. Report discrepant information

As described for experiment 1, rich, detailed descriptions and discrepant information were included in the report.

## **Positivist view**

### 1. Construct validity

In respect to construct validity [Easterbrook07], it was already acknowledged that these experiments, performed as emulations of real meetings between stakeholders and engineers, provided a reasonable approximation of the context of a real meeting.

### 2. Internal validity due to confounding factors

Concerning internal validity due to confounding factors [Easterbrook07], familiarity and learning were avoided using, for each session, two different systems. The system that was used for the jigsaw puzzle was different from the one used with textual requirements documentation.

Still concerning confounding factors, after experiment 2, the hypothesis was raised that participants were not given the same information (treatment) while working with the jigsaw puzzle, compared to working with the textual documentation. The difference had to do with the information about conflicts, which was provided (as jigsaw puzzle cues) when requirements were presented through the jigsaw puzzle, but not in the textual documentation.

Another possible confounding factor detected when analysing experiment 2, was the tiredness effect in the participants. In fact, in all three sessions of experiment 2 the first system was always presented using the jigsaw puzzle metaphor, and the second using the textual representation. The participants' preference for analysing conflicts and ambiguity in requirements, with textual versus jigsaw puzzle metaphor, could very well be biased in favour of the jigsaw puzzle. This could happen just because when participants worked with system presented in text (always in the second half of a two hour session) they were already too tired.

### 5.4.3 Experiment 3

#### 5.4.3.1 Set up

This experiment contained two sessions, which lasted 2 hours each, with two different groups using two systems (one with a jigsaw puzzle, the other with textual documentation), and each group worked as control group for the other group. The examples used were the Crisis Management System (CMS) [Kienzle09], the requirements Availability, Reliability, Real-time, and Accuracy; and the Health-Watcher (HW) [Soares06], the requirements Availability, Performance, Security, and Standards.

The third experiment aimed to further confirm the hypotheses described in Section 5.2. Concerning the improvement of the experiment design the following changes were introduced:

- Better design of the information given to the control group, i.e., give the “same amount and type” of information for the system that is described in text as the one presented through the jigsaw puzzle. This was done by introducing in the text presentation phrases saying “Detected conflicts between requirement X and requirement Y”, exactly for the same situations that in the jigsaw puzzle were signaled with interlocking shapes.
- To avoid a possible tiredness effect so that the jigsaw puzzle was not always the first presentation in the session. Table 5-6 shows for each session the systems used, their presentation and order of usage.

	<b>Jigsaw puzzle presentation</b>	<b>Text presentation</b>
<b>Session 1</b>	1st - CMS	2nd - HW
<b>Session 2</b>	2nd - HW	1st - CMS

**Table 5-6 Systems, type of presentation and order of usage in each session of experiment 3.**

- To include in the teams participants with backgrounds other than engineering. It was possible to include participants with management background, which brought interesting insights into the evaluation.

Concerning the improvement in the design of the jigsaw puzzle metaphor, the following was done:

- Use in the Health-Watcher system rectangular pieces so that one piece positioned in the row below (Performance requirement) could be connected with the two upper pieces (Security and Standards requirements).

The two sessions performed in experiment 3 were attended by three participants and one researcher, acting as facilitator. Each group had one participant with management background, one engineer with some RE expertise and one engineer with no RE expertise.

Each session emulated the “real-life context” of a consultation meeting between requirements engineers and stakeholders to review a specific part of the requirements documentation. The design of the sessions, including the tasks asked to the participants, was the same as for experiment 2 (see Section 5.4.2), except for the improvements already described.

The jigsaw puzzle pieces size was the same as in experiment 2: 12 cm x 12 cm, for session 1, and in session 2, two pieces measured 12 cm x 20 cm and the other two 12 cm x 12 cm. The requirements in the jigsaw puzzle were labelled for reference in discussion and report. The pieces had in background the image that won the preference in the survey described in Appendix G (picture D, “orange, yellow abstract”). As in that survey there was a picture that won the second place with the difference of one point, it was decided to show in experiment 3, this other picture (picture A, “clouds and water”) and query the participants of this experiment which picture they preferred. Both the CMS and the Health-Watcher jigsaw puzzles had the “orange, yellow abstract” picture with text printed on a paper that was then glued to the K-line support.

Figures 5-5 and 5-6 show pictures of the jigsaw puzzles used in experiment 3, representing the CMS, and the Health-Watcher systems.

An interlocking shape was added to the CMS puzzle (compared to experiment 1 and 2) between the “Reliability” and “Accuracy” pieces as a cue for the conflicts between these requirements that were raised by the participants in experiment 1 and 2, even if conflicts between these requirements were not initially considered by the researchers.



Figure 5-5 The jigsaw puzzle for CMS example, used in experiment 3.

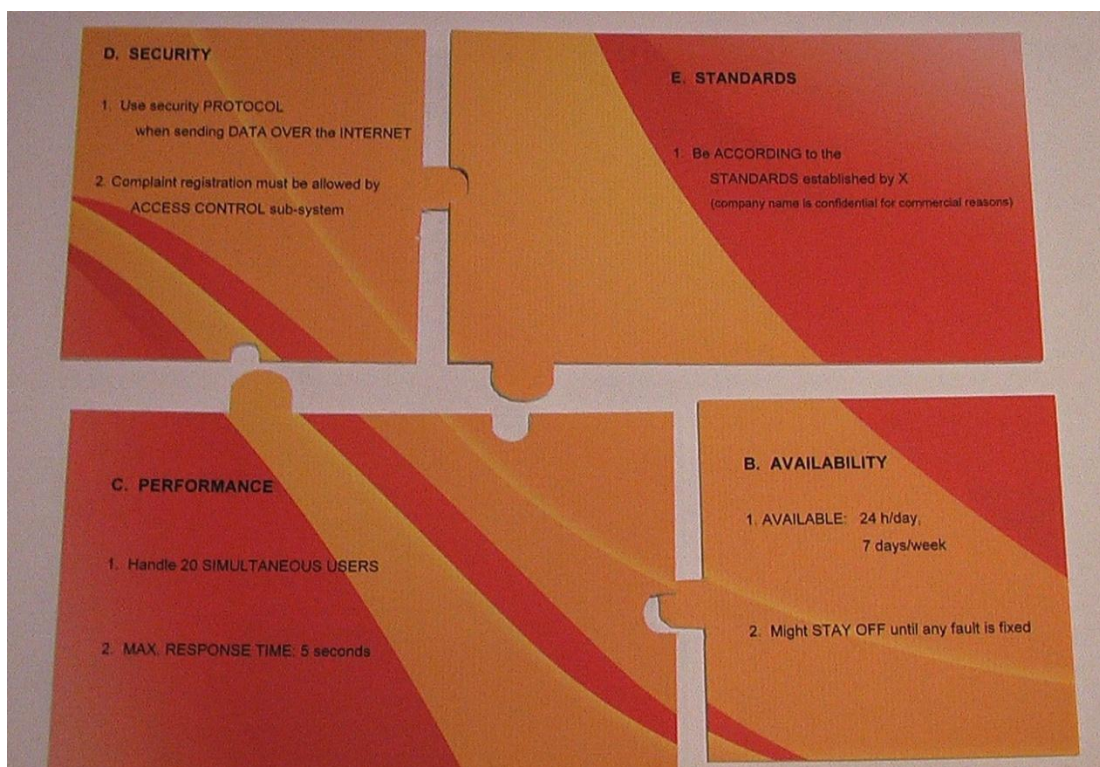


Figure 5-6 The jigsaw puzzle for the Health-Watcher example, used in experiment 3.

### 5.4.3.2 Description of experiment and results

#### Hypothesis H1 - Increase effectiveness when compared with text presentation

##### 1. Number of imperfections detected

Tables 5-7 and 5-8 show how many conflicts, with cues, were totally or partially discovered by the participants in session 1 and 2, respectively. The tables also show the number of other imperfections (ambiguities and conflicts), for which there were no cues (through badly fitting interlocking shapes, or in text) but also detected by participants. Appendix H lists these imperfections.

	N conflicts with cue	N conflicts with cue detected	N imperfections with no cue but detected
<b>CMS jigsaw puzzle</b>	4 <sup>27</sup>	2 - totally 1 -partially	8
<b>HW text</b>	4	4	3

**Table 5-7 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 3, session 1.**

	N conflicts with visual cue	N conflicts with cue detected	N imperfections with no cue but detected
<b>CMS text</b>	4 <sup>28</sup>	1 -partially	10
<b>HW jigsaw puzzle</b>	4	2 – totally 2- partially	5

**Table 5-8 Number of conflicts with visual cue, detected with visual cue, and imperfections detected with no visual cue, in experiment 3, session 2.**

<sup>27</sup> CMS puzzle in experience 3 considered four conflicts. In experiments 1 and 2 just three.

<sup>28</sup> CMS text in experience 3 considered four conflicts. In experiments 1 and 2 just three.

Concerning the CMS in text, the participants worked very hard and for quite a long time, and found initially many ambiguity cases. When trying to reason on possible conflicts they found the barrier of not knowing the answer for the questions on ambiguity. After about one hour and a half working with the CMS in text the researcher proposed to switch for the second system (the meeting was becoming too long).

## *2. Preferences between jigsaw puzzle and text presentations*

Concerning the preference to perform these RE tasks with the jigsaw puzzle presentation or with text, the participants in experiment 3 preferred unanimously the jigsaw puzzle. The reasons presented were: the jigsaw puzzle is colourful, it presents less information, and both the relevant topics and the relations between pieces are more direct and easier to spot. The text inside the jigsaw puzzle pieces is presented in a simpler and easier way to see and understand the possible connections: there are keywords, smaller sentences, and the letters are bigger. It is also possible to move the different parts, and it is more flexible.

Even agreeing with the above paragraph, one participant raised the issue that the examples worked might not be comparable, and in particular that the Health-Watcher could be considered simpler, i.e., having requirements easier to examine. This participant pointed that they might have felt the Health-Watcher (with the jigsaw puzzle) easier because with the first system (CMS in text) they had to learn how to perform the tasks they were asked to perform.

The main disadvantages pointed out when commenting on the text presentation were the need to do a lot of reading and work to understand the system. And that, as in text they had a tendency to work vertically, it was not so easy to study the relations between the pairs of requirements.



Their general conclusion was that the jigsaw puzzle can be a good tool, much better than the text, but it assigns responsibility to the person who treats the information for the jigsaw.

Hypothesis H2 - Foster team work and communication, improving commitment of stakeholders in co-authoring of requirements and co-responsibility in handling of ambiguities and conflict and

Hypothesis H3 - Promote a relaxed environment, and thus team cooperation and creativity, once the metaphor builds upon a game (the jigsaw puzzle)

In the discussion part of the session, one participant noted that one advantage of the jigsaw puzzle was that with the puzzle they worked collectively, all focused on the same time on only one object.

Hypotheses H2 and H3 can be observed in the following steps of sessions when the jigsaw puzzle metaphor was put in action:

#### *1. Assembling the jigsaw puzzle*

In the first session of experiment 3, when trying to assemble the CMS jigsaw puzzle (Figure 5-5), one of the participants inverted some pieces to make the interlocking shapes fit better. The participants were trying to find a strategy in a relaxed and fun mood. In this strategy it appeared that making the interlocking shapes fit was an important goal. The researcher reminded that this was a jigsaw puzzle where the pieces are to fit with the picture side up. S/he tried to facilitate work asking what cues participants usually use to assemble a jigsaw puzzle. Participants tried through ordering the capital letters in the pieces and then through the text. The researcher

insisted to try to assemble the jigsaw puzzle as usual and not to get worried if the interlocking shapes do not fit. The participants were then able to set well the jigsaw puzzle.

In the second session of experiment 3, at the moment of assembling the HW jigsaw puzzle (Figure 5-6), participants used the colour as a cue and assembled the puzzle easily.

## *2. Scanning for conflicts/ambiguities (both in jigsaw puzzle and text)*

In session 1, when starting to search for conflicts in CMS jigsaw puzzle, the participant with management background interpreted the requirements considering a broader view than just those specific requirements; in fact s/he was concerned with what had, in the broader system, to be assured so that the requirements in analysis could be fulfilled. This interpretation was at the beginning difficult to understand by the engineering background participants, who were thinking more “inside the box” of those specific requirements. But with the facilitation of the researcher the group could understand the two different ways to reason and from that point worked as a team.

In the second session of experiment 3, after assembling the HW jigsaw puzzle, participants started to feel curious about what connections there were among pieces (they had already their minds trained to search for relations from the exercise they performed before with text) and from then on they worked as a team searching for conflicts/ambiguities.

### 5.4.3.3 Threats to validity

#### Constructivist view

Concerning the Creswell framework [Creswell02] strategies for improving validity of constructivist research, the possible introduced bias are described below.

##### 1. Clarify bias

When designing the Availability jigsaw puzzle piece for CMS system, during the formatting of text the investigator removed the word “system”, which appears in the text version. In the session with the CMS system as text, the participants pointed the ambiguity case:

- The word “system” in Availability requirement and the word “control center” in Real-time requirement present ambiguity (“system” only appears in the text version): are these words to be interpreted as meaning the same, or not?

This ambiguity case could not have been described by the participants in the session with the CMS system using the jigsaw puzzle, as the word “system” does not appear in it.

This case shows the possibility that while formatting the requirements phrases to produce the jigsaw puzzle (justified with the willingness to make the text more readable), the investigator implicitly chooses one interpretation (in this case assumed that “system” and “control center” referred to the same entity) and thus might introduce bias. One way to avoid this kind of bias is to use the output of the ambiguity detection tools to inform the formatting of the requirements text that will be written in the jigsaw puzzle pieces. The choice of one possible interpretation is just one example of bias that the person or tool that treats information may introduce.

## 2. Rich, detailed descriptions and 3. Report discrepant information

As described for experiments 1 and 2, rich, detailed descriptions and discrepant information were included in the report.

### **Positivist view**

#### 1. Construct validity

In respect to construct validity [Easterbrook07], it was already acknowledged that these experiments performed as emulations of real meetings between stakeholders and engineers, provided a reasonable approximation of the context of a real meeting.

In experiment 3 the quality of the real meeting emulation was improved through the introduction of participants with management background.

#### 2. Internal validity due to confounding factors

Concerning internal validity due to confounding factors [Easterbrook07], familiarity and learning were avoided using, for each session, two different systems presented one in text and the other in jigsaw puzzle, as was already done for experiment 2.

Still concerning confounding factors, after experiment 2, the hypothesis was raised that participants were not given the same information about conflicts, which was provided (as jigsaw puzzle cues) when requirements were presented through the jigsaw puzzle, but not in the textual documentation. Thus, in experiment 3, for the requirements documentation presented in text, it phrases were added indicating the presence of possible conflicts, like for instance for the CMS example: "Detected conflicts between Availability and Real-time". This improvement has reduced, as

much as possible, the difference in the information provided (which is always present when there are two different visual presentations).

Another possible confounding factor detected when analysing experiment 2, was the tiredness effect. The participants' preference for analysing conflicts and ambiguity in requirements with textual versus jigsaw puzzle metaphor, could very well be biased in favour of the jigsaw puzzle, because this was presented always first when participants were less tired. To avoid this possible factor, in the second session of experiment 3, the mode of presentation of the system requirements was the inverse of the one used in the first session.

One other possible confounding factor, raised by one participant, is that the examples worked might not be comparable, and in particular that the Health-Watcher could be considered simpler when compared to the CMS, i.e., having requirements easier to examine. It is difficult to ensure that the requirements documentation for two systems is "similar" in "handling difficulty".

#### **5.4.4 Synthesis of discussions about possible developments**

This section provides a synthesis of the main results collected during discussions about the aspects participants liked in the jigsaw puzzle metaphor, as well as possible developments of the jigsaw puzzle metaphor.

#### 5.4.4.1 Design of the pieces and structure of the jigsaw puzzle design

The aspects participants liked in the design of the jigsaw puzzle metaphor, in comparison to the usual text presentation of requirements documentation, were as follows:

- The text inside the jigsaw puzzle pieces is presented in a simpler and easier to read fashion, there are keywords, smaller sentences, the letters are bigger, the “important” information is bullet-pointed; some parts are highlighted (through upper case).
  - As a consequence both the relevant topics and the relations between pieces are more direct and easier to spot. This enables to accentuate some aspects, which make the users think if, for instance, the numbers (shown in more than one piece for the same or related aspects and that appear to be related) should match up;
- The jigsaw puzzle is colourful; someone said the orange and yellow colours used in experiment 3 lead into action;
- Users can move the different pieces (representing requirements), making it a more flexible presentation to work with;
- It promotes group work: all focused at the same time on only one object;
- Through presenting the requirements connected amongst them, makes users understand (visually) that requirements have impact on each other.

The structure, that the jigsaw puzzle provides, was discussed in some sessions. The way the puzzle pieces are laid down imposes/proposes an order to scan the possible conflicts/ambiguities. In experiment 1 with the L-shape, users are conducted to start the analysis by one of the ends. In experiments 2 and 3 with the rectangular 2x2 grid

shape, users are conducted to start typically with the top left piece. In all cases users tend to analyse the first piece they pick with the ones that are adjacent. The question raised was: is this a good or bad effect? Participants' opinion was that the way the puzzle pieces are disposed allows controlling the focus of the discussion. It is important to note that, according to participants' opinion, the structure provided by the puzzle if on one hand promotes analysis in a certain order, on the other does not limit people to find imperfections among pieces that are not directly connected. This opinion was supported by the results collected, since participants reported conflicts among pieces not directly connected or not having badly fitting interlocking shapes.

One participant commented that the jigsaw puzzle helps to think about the pairs of pieces that are side by side but not in diagonal. He suggested it would be good to allow more connections for interlocking shapes. To accommodate more connections, another participant suggested using pieces in shape of hexagon instead of rectangle (which is something the researchers also considered but did not evaluate). Another suggestion was to have bigger pieces

The design of the jigsaw puzzle can be better or worse, and this is a responsibility of the engineers in charge. More evaluation on the effects different shapes can have is also needed.

#### **5.4.4.2 Possibility to present the jigsaw puzzle in a digital form**

When the participants were queried on the possibility to support the jigsaw puzzle metaphor in a digital form, like in an interactive digital table, there were two types of reactions (except for one participant): some accepted the idea enthusiastically, some

others liked or even preferred the physical pieces but would also like (or in some cases accept) to have it in a digital table with all the advantages of the digital support. From the 22 participants in the 3 experiments, one participant definitely assumed his preference for the physical mode of presenting the jigsaw puzzle.

The participants that accepted enthusiastically the possibility of having the jigsaw puzzle presented in an interactive table foresaw that this type of digital support could enable to work in pretty much the same way, as they did with the physical pieces, with the advantage of providing extra functionalities. One participant called it the “i-jigsaw puzzle”. Some of these advantages that were suggested by both groups are:

- Direct manipulation: hand writing (notes with comments and rationale with show/hide) and drawing (e.g. arrows, links between different words in different requirements, these links could have different widths if there are more conflicts in a link), move the pieces, change the size of pieces;
- Visual helpers: to have icons in each piece to illustrate the requirement that is being considered; highlight the words/expressions that provoked the conflicts;
- Intelligent real-time interaction (like changing a requirement text and see the interlocking shape due to a conflict disappearing);
  - o this functionality would be very important to provide users work progress assessment in real-time;
- Have 3D jigsaw puzzles to enable visualization of a bigger number of relationships among requirements, and possibly with several levels of abstraction;
- Organization and storage;
- Easy distribution (e.g. possibility to send the jigsaw puzzle before/after the meeting)
- Collaborative functionalities (e.g. recording with different colors the contributions from different participants); and



- Technology is fun and this would have a positive impact in creativity: for technology fans the jigsaw in digital format would probably be an extra motivation compared with text.

The group of participants that wished to retain the physical pieces noted that it is a disadvantage to lose the possibility to touch the pieces physically, and lose the initial challenge of putting the pieces together physically and realize how they are physically linked. This group of participants accepted to work with the jigsaw puzzle in a computer, but they still would like to have the physical form.

One of participants pointed out that he was not feeling comfortable with the fact that the physical pieces did not match, and stressed he would not use a puzzle like this to discuss with his customers. When asked if having an interactive digital table that shows pieces fitting well when the problems are solved would make the jigsaw puzzle more usable, he replied that in that case he might use it.

One very interesting suggestion concerning the type of support for the jigsaw puzzle is to adjust the support according to participants' profile. According to their preference the meeting could run with the jigsaw puzzle physically or digitally supported.

#### **5.4.4.3 Picture in the background of the jigsaw puzzle pieces**

In the debate about the picture background, some participants found it distracting. They reported a tendency to make a connection between the picture and the system domain. Having this in mind one participant suggested this could be explored positively if the picture has something to do with the domain of the system. Another

idea was to have the background of the requirements with colours conveying information like priority/importance (e.g. red for highly important).

A very interesting suggestion and with marketing value, is to use the colours of customers in the picture background of the jigsaw puzzle. The idea is to use an abstract picture like the one used in experiment 3 (Figures 5-5 and 5-6) but with the colours of customer's company.

Referring to the preference between the background images shown in Figure G-1 (the "clouds and water" picture) and 5-5 or G-4 (the "orange, yellow abstract" picture) used in the two sessions of experiment 3, all the six participants except one voted for the picture of Figure 5-5. There was one participant who preferred the "clouds and water" picture, because this is a natural, more serene picture. The ones that preferred the "orange, yellow abstract" picture pointed out the following reasons: it is colourful, it has hot colours and contrast, it has lines, and the connections are easier to spot. They also pointed out that the "orange, yellow abstract" picture is more dynamic: this type of colours will lead into action. They said that the text in the picture of Figure G-1 is more difficult to read, since it contains blinds. With cold colours it is more difficult to make contrast.

When queried why they did not write on the pieces (experiment 3), participants said they would like to but did not do it in order to not deface the picture.

#### **5.4.4.4 Possibility to identify conflicts and ambiguities that were not considered a priori**

The participants debated if users would be able to identify conflicts that were not considered a priori by the engineers in charge and that are not visualized through the jigsaw puzzle. The general conclusion was that finding imperfections not “marked” is not prohibited by this jigsaw puzzle representation. Participants reported that initially they focused on the pairs that were given, but afterwards they went beyond the original limits of the given structure.

#### **5.4.4.5 Attachment of semantics to the interlocking shapes of the jigsaw puzzle**

In one of the sessions, participants tried to attach semantics to the interlocking shapes of the jigsaw puzzle: between the number of interlocking shapes and the number of phrases (in each requirement) with imperfections. They soon concluded that from the presented puzzle they could not draw this type of association. Later on in the meeting, participants came back again to the discussion of adding more semantics to the interlocking shapes. Some other meanings proposed for the bubbles were “what the requirement might provide (to other requirements)” or “what the requirement might need”.

One participant stressed that in his opinion this metaphor would not work without a semantics attached to the interlocking shapes. He stated: “we need to know the semantics for each interlocking to go any further in exploring the conflicts/ambiguities”. Nevertheless this participant worked with the material

presented and collaborated with the others in the task proposed. On the contrary another participant expressed clearly that he did not find anything special in the interlocking shapes, he just looked at the material presented as a metaphor to detect and analyse conflicts and ambiguities. For this participant the spatial relations among pieces (proximity can mean more interaction) are the ones that can benefit from having semantics attached. And in particular if this semantics is associated to spatial relations it can be used to direct people's focus in different ways (with advantages but also with risks).

## **5.5 Analysis and discussion**

This section provides an analysis and discussion of the main results collected during the evaluation, and other issues such as the explorative perspective of the evaluation method, and the researcher and the facilitator roles.

## **5.5.1 Increase effectiveness when compared with text presentation**

### **5.5.1.1 Number of imperfections detected**

#### The CAS example

The group that worked with the CAS example in text (experiment 2, session 2) was able to detect two out of three conflicts but complained that this detection was difficult because the example contained too much ambiguity. The group that worked with the CAS example in jigsaw puzzle (experiment 2, session 3) found and talked about one conflict (not reported) and lots of ambiguities. It is plausible to conclude that the group that worked with text might have found the conflicts because they knew from the previous task they performed with the other system (with jigsaw puzzle) that the purpose was to scan for conflicts.

The main lesson from the CAS example is that it is not adequate (and thus not worthwhile) to present a set of requirements with too much ambiguity using a jigsaw puzzle. It might help to present the requirements in squares/rectangles and give to the text the same treatment that was done for the jigsaw puzzle. In fact, in the performed experiments (not only with CAS system) it happened sometimes that, if there would be a clarification in an ambiguity problem, then conflicts referring the same requirements, were deemed not pertinent to discuss.

### The Health-Watcher example

With the Health-Watcher system there were significant improvements between the numbers of conflicts detected from experiment 2 to experiment 3. Concerning the jigsaw puzzle presentation from 2 conflicts detected in experiment 2 to 2 conflicts and part of another 2 in experiment 3. Concerning the text, from 1 conflict detected in experiment 2 to all (the 4) conflicts detected in experiment 3. In fact, both the jigsaw puzzle presentation and the text presentation were improved.

The jigsaw puzzle pieces shapes were changed to allow a direct interlocking shape between Performance and Standards, evolving from what is in Figure 5-3 to Figure 5-6. This enabled the group that worked with the puzzle in Figure 5-6 (experiment 3) to detect the conflict between Performance and Standards, and part of another conflict. This seems to indicate that the interlocking shapes are important in helping to detect conflicts.

In the text versions the change was to introduce phrases indicating the possible existence of conflicts like: "Detected conflicts between Security and Standards", with the goal to provide as much as possible the same information in both presentations.

It is interesting to note that for Health-Watcher a slightly better result was achieved in experiment 3 with the text presentation (all the 4 conflicts detected) than with the jigsaw puzzle (2 conflicts detected and part of the other 2). A plausible explanation for the slightly poorer performance of the group working with the jigsaw might be tiredness. In fact this work was performed after this group worked for one hour and a half with another system in text. Anyhow, the group that performed so well with the text presentation, detecting all the four conflicts, were unanimous in preferring to work with the jigsaw puzzle.

### The CMS example

With the CMS example, there was much better performance, in the number of conflicts detected, with the jigsaw puzzle presentation compared to the text presentation. In fact with the text presentation, in one session (experiment 2) no conflict was detected and in the other (experiment 3) just part of one conflict was detected. In this case the fact that the text presentation was improved for experiment 3 with the phrases indicating the presence of conflicts seems to have had no positive effect (or a marginal one).

With the jigsaw puzzle presentation in experiment 1 every conflict was detected (from the three conflicts with visual cue). In experiment 2 the participants detected one conflict and part of two other conflicts (there were three conflicts with visual cue). In experiment 3 the participants detected two conflicts and part of a third one (there were four conflicts with visual cue).

#### **5.5.1.2 Preferences between jigsaw puzzle and text presentations**

Concerning the preference of performing the identification and analysis of conflicts and ambiguities, with text or with the jigsaw puzzle presentation, all the 22 participants except one (who showed no preference), preferred to work with the jigsaw puzzle. A number of participants could explain why they definitely preferred the jigsaw puzzle presentation. The main reasons are described in Section 5.4.4.1.

Interestingly, some participants could also perceive the danger that the jigsaw puzzle presentation brings: if the person that treats information is biased then participants in

the meeting would also be biased. This is not a danger in itself but indeed a responsibility of the requirements engineers in charge.

The two groups that worked with the CAS example (with whatever presentation) believed that with this example the comparison was unfair, because the CAS example had too much ambiguity. They provided their opinion while making an effort to abstract from this problem. The improvement of text with the phrases indicating the presence of conflicts did not influence the result, since in experiment 3 (the one where the text was improved) the preference for the jigsaw puzzle was unanimous.

One participant reported having no preference for either presentation. Another preferred text, but when interviewed for the survey on the picture background, wished to change his opinion saying “the puzzle could provide an interesting tool in an interactive table”. A third participant initially thought the jigsaw puzzle would be useful for analysts only, in a more technical-oriented tool “with the puzzle inside it”. This third participant, when interviewed for the survey on the picture background, changed his position providing suggestions on how the jigsaw could be used with customers, for instance using the colours of the company in the picture background.

### **5.5.1.3 Conclusion**

One of the hypothesis evaluated in this research work, was that the jigsaw puzzle metaphor and the method to work in stakeholders group consultation meetings, increases effectiveness in the communication and handling of ambiguity and conflict in requirements, compared with using text presentation. The experiments' results demonstrate that both the jigsaw puzzle metaphor and the method exhibit a very



good potential to increase their effectiveness in communicating and handling ambiguity and conflict in stakeholders meetings, when compared to what is the common mode of performing this task in text presentation.

The most significant result sustaining this statement is that all participants (17 in experiments 2 and 3), except one (who showed no preference), preferred to work with the jigsaw puzzle than with the text presentation. In experiment 1 (where 5 of the participants were knowledgeable in requirements engineering and thus able to evaluate the comparison of the proposed approach against text) the participants concluded: “the jigsaw puzzle-based presentation is really good in helping identifying problems and conflicts”. It is also significant that a number of participants are aware of and explain the jigsaw puzzle features that make them prefer to work with it.

The examples studied were few (thus with no statistical significance). However, both in the CMS example and the Health-Watcher, the number of conflicts detected with the jigsaw puzzle was, in general, greater than the number detected in the textual presentation. There is only a slightly better result with the text presentation for Health-Watcher in experiment 3. This might be explained by tiredness as was sustained in the analysis of Health-Watcher example (Section 5.5.1.1). It is relevant to note that, in this experiment 3, the textual presentation was not in normal plain text, but text improved with the phrases saying “Detected imperfections between concern X and concern Y”.

The number of conflicts and ambiguities well communicated and thus identified, as well as, the suggestions for handling the imperfections can improve even more significantly, with an interactive digital support. This is the researchers’ belief, together with the (enthusiastic) positive opinion of the participants towards having the jigsaw puzzle metaphor digitally and interactively supported.

In itself just the fact that there is a digital support (other than pure text) for analysis and rational descriptions (e.g. comments, arrows relating requirements) can bring enormous benefits: these analysis and rationale are not lost (are recorded), and can easily be distributed.

If the digital support also provides intelligent real-time interaction, enabling that the user changes, for instance, the text of a requirement and the interlocking shapes are redesigned according to a recalculation of possibility of conflicts, then we can say the (digitally supported) tool is enabling us to move towards minimisation of conflicts. Furthermore, if this progress in work is automatically recorded, its effects are visually observable by the users involved and can be discussed “now”, instead of in the next meeting, after the ones involved “calculate” the possible side-effects. This would represent a huge improvement in handling of requirements conflicts.

However, the enormous potential foreseen for a digitally supported jigsaw puzzle can only be effective if this tool is well designed, using the best practices of usability, and in particular respecting the characteristics of RE task and its users.

## **5.5.2 Improve co-authoring of requirements and co-responsibility in ambiguity and conflict handling and Promote team cooperation and creativity**

### **5.5.2.1 Assembling the jigsaw puzzle and scanning for ambiguities and conflicts**

The fact that participants were given a physical set of objects to work on together, that none of them knew before, and that it implied a challenge (a game) did work as an initial “break of ice”. People were excited and having fun. As this task of assembling the jigsaw puzzle was not perceived as a technical one, people felt free to offer their comments on issues like strategies to assemble a puzzle.

In the only session (experiment 3 session 2) when the first system to be discussed was in text, participants started to perform the job they were asked and worked as a team but there was not the initial excitement and participants were behaving like in a normal professional meeting. The fun part came after when they were asked to assemble the jigsaw puzzle: they seemed excited and behaved more informally.

It is interesting to note that different groups behaved quite differently and used diverse strategies to assemble the jigsaw puzzle. The session of experiment 1 is partially comparable with the others as there was no picture in the background. In this session the interlocking shapes and the direction of the text was the cue used to assemble the puzzle.

Interestingly two groups (in experiments 2 and 3) tried to assemble the pieces through the alphabetical order of the letter used to identify the piece, and two other groups pondered if one needed to read the text to be able to set the pieces. These

“problems” were easily solved when the researcher asked the participants to use the typical jigsaw puzzle cues and strategies.

Two groups behaved as if making the interlocking shapes fit was an important goal. One group inverted some pieces (positioning the picture upside down) to try to make the interlocking shapes fit perfectly. The other assembled the puzzle using the colour cue but then set the pieces apart to try again, because they were not fitting well. Again these “problems” were easily solved when the researcher asked the participants to use the typical jigsaw puzzle cues and strategies, and not to worry about the bad fittings because these were on purpose and had a meaning, which was going to be explained.

#### **5.5.2.2 Conclusion about hypothesis 2 - Improve co-authoring of requirements and co-responsibility in ambiguity and conflict handling**

It was also hypothesized that the jigsaw puzzle metaphor and the method of work would foster team work in groups having elements with different backgrounds, improving both cooperation, co-responsibility and co-authoring.

It was clearly observed that electrical engineers, and managers with no software engineer background, did not have any problem in understanding both the task to perform and the mode the requirements were conveyed with the jigsaw puzzle presentation. Beginning the meeting with assembling a jigsaw puzzle<sup>29</sup> also contributed to make everyone feel at the “same level”. The jigsaw is something that

---

<sup>29</sup> Except in experiment 3 session 2.

everyone can do; it permits that users enter in contact with what is going to be discussed in a smooth way.

In fact the contributions, in terms of conflicts and ambiguities identified as well as possible modes of handling them, came from participants in spite of its background. This means co-responsibility and co-authoring.

### **5.5.2.3 Conclusion about hypothesis 3 - Promote team cooperation and creativity through a relaxed environment**

The fun side of assembling the jigsaw puzzle clearly contributed to a more informal and relaxed environment. Participants had fun and enjoyed the sessions. Some tried to assemble the jigsaw puzzle pieces with the picture up-side down, which indicates they had their minds open to do things in an unusual way. The group that worked with the text presentation first (for one hour and half), when presented with yet another example to work showed tiredness but then the jigsaw puzzle boosted the work and the group showed enjoyment up to the end of the meeting.

It is important to acknowledge that what was observed and described concerning the ability of the proposed approach to foster team work in groups having elements with different backgrounds, improving both cooperation, co-responsibility and co-authoring, provides a conclusion with limitations. These limitations concern the fact that the evaluation was not done with real stakeholders. The participants did not have a priori any reason to create an undesirable environment.

## **6 Conclusion and future work**

### **6.1 Summary and contributions**

The aim of the research, described in this thesis, was to investigate, and contribute to the improvement of identification, communication, and handling of ambiguity and conflict in non-functional requirements, inadvertently introduced during the RE process.

The thesis proposes a jigsaw puzzle metaphor and an associated method that increase effectiveness of detection and handling of ambiguity and conflict in requirements, when compared with the usual textual presentation of requirements. This approach also fosters team work and communication, and improves commitment of stakeholders in co-authoring of requirements and co-responsibility in resolution of imperfection. As the proposed solution builds upon a game (the jigsaw puzzle), it promotes a relaxed environment and thus creativity.

The next section clarifies the approach, through the objectives it pursued.

## 6.2 Aims and objectives revisited

The objectives of the thesis were as follows:

1. Develop guidelines/heuristics to identify the ambiguities and conflicts in requirements that are most pertinent to be discussed during stakeholder consultation meetings.
2. Develop an effective communication mechanism that makes the ambiguity and conflict in key NFRs explicit. In order to be effective, the communication mechanism must consider the task at hand in the consultation meetings as an analytical/creative task, performed by a group of heterogeneous stakeholders.
3. Develop and investigate a method for conducting the consultation meetings with the stakeholders, using the communication mechanism to promote cooperation and co-responsibility towards the requirements document. The method should make the analysis and search for conflict and ambiguity resolution an enriching experience, through a fun and relaxed environment.

We next discuss how the work presented in the thesis addressed the above objectives.

### 1. Develop guidelines/heuristics to identify the most pertinent ambiguities and conflicts to discuss in consultation meetings

In order to identify the most pertinent ambiguities and conflicts, to enable making them visually explicit (marked in some way), the approach proposed heuristics and guidelines. The final selection of the sets of requirements, to be used in a group

consultation meeting, belongs to the team of requirements engineers in charge of the requirements engineering process.

For the experiments performed, the identification of the most pertinent ambiguities and conflicts was done manually. Participants were unanimous (except one) to say that it helped to work faster to have ambiguities and conflicts visually explicit (even if not all were marked).

Considering the automation of the process to identify the most pertinent imperfections this thesis contributed with a proposal for a realisation with existing tools. This proposal shows how a complete implementation for this automation can be achieved.

### *2. Development of an effective communication mechanism for the requirements making ambiguities and conflicts explicit*

and

### *3. Development and investigation of a method to conduct the consultation meetings using the communication mechanism*

The approach proposed the jigsaw puzzle metaphor and a method, to detect and handle requirements ambiguities and conflicts in stakeholders' group consultation meetings.

Having a list of (the most pertinent) ambiguities and conflicts in requirements, the jigsaw puzzle metaphor proved to be a good and effective communication mechanism for these imperfections in RE, by making the conflicts explicit. The quality of the jigsaw puzzle mechanism, as a communication mechanism, and the proposed method to work with the jigsaw puzzle, was enabled since both took into consideration the RE characteristics.



The jigsaw puzzle metaphor proved to be a good communication mechanism because it is a visual metaphor that possesses in particular the following characteristics, important to achieve the goals listed below:

- It is not a technical notation; instead it represents requirements through jigsaw puzzle pieces and conveys the information about conflict through the analogy with a badly fitting interlocking shape; this is important to respect the possible different backgrounds of participants; as it does not need to be learned<sup>30</sup> all the participants “feel at the same level”, and thus with the same ability to be co-responsible for the requirements documentation.
  - It was clearly observed that electrical engineers, and managers with no software engineer background, did not have any problem in understanding both the task to perform and the mode the requirements were conveyed with the jigsaw puzzle presentation.
  - Beginning the meeting with assembling a jigsaw puzzle also contributed to make everyone feel at the “same level”. The jigsaw is something that everyone can do; it permits that users enter in contact with what is going to be discussed in a smooth way.
  - In fact the contributions in terms of imperfections detected, and suggestions on how to handle those imperfections, came from participants in spite of its background. This means co-responsibility and co-authoring.
- It enables to use the analogy of building an object, the jigsaw puzzle (in RE, the system), out from different pieces (in RE, the different requirements) that have to be correctly assembled to yield a final “correct” product; this analogy emphasizes the creative facet of RE.

---

<sup>30</sup> In the event someone does not know how to play a jigsaw puzzle, it can be learned quickly and easily.

- The participants in the meeting have to assemble the jigsaw puzzle, which is itself a game (thus it is not necessary to begin the meeting with a game as it is commonly used in RE elicitation meetings) with the purpose of inducing a fun and serene environment.
  - Participants had fun and laugh. Dared to try to assemble the jigsaw puzzle pieces with the picture up-side down, which indicates they had their minds open to do things out of the usual way. The group that worked with the text presentation first (for one hour and half), when presented with yet another example to work showed tiredness but then the jigsaw puzzle boosted the work and the group showed enjoyment up to the end of the meeting.

The proposed method of working with jigsaw puzzle metaphor is based in the common modus operandi for stakeholder's consultation meetings improved in some aspects, related with the goals listed below. These aspects are:

- The participants have to first assemble the jigsaw puzzle: this is important because it is fun and improves relaxation; and secondly, to scan visually for the ambiguities and conflicts, instead of the imperfections being readout: this is important to improve the sense that imperfections are everyone's problem and require cooperation and co-responsibility.
  - The experiments showed that the fun side of assembling the jigsaw puzzle clearly contributed to a more informal and relaxed environment, which contrasted with a formal environment while working with text presentations.
- All stakeholders work with the same common "document": the jigsaw puzzle, instead of each with his/her own copy: this is important to improve the sense

that imperfections are problems common to all, and require cooperation and co-responsibility.

- The experiments showed a clear difference between working with text and with the jigsaw puzzle. With the text presentation each participant performed the scan for conflicts and ambiguities and the analysis, lonely and quietly. While with jigsaw puzzle participants worked together, cooperating and discussing both how to assemble the jigsaw puzzle, and what is and is not an imperfection, and why.

Concerning increase of effectiveness (in comparison to textual presentation), both the jigsaw puzzle metaphor and the method to put it into action have shown a very good potential to increase effectiveness in the handling of ambiguity and conflict in requirements, when compared to what is the common mode of performing this task in text presentation. All participants except one (who showed no preference) preferred to work with the jigsaw puzzle than with the text presentation. Participants concluded: “the jigsaw puzzle-based presentation is really good in helping identifying problems and conflicts”. It is also significant that quite some participants are aware and explain the jigsaw puzzle aspects that make them prefer to work with it. The results of the experiments indicate that this approach is more useful when the requirements document is in a certain level of maturity.

The interest, adequacy, effectiveness, and potentiality of the jigsaw puzzle metaphor and the method to handle ambiguity and conflict in RE, during stakeholders' consultation meetings is not only foreseen by the researcher, and tested through the experiments. Indeed all the 22 participants except one, expressed opinions, after working with the jigsaw puzzle through the method, that show they also believe in this approach. It is relevant to stress that both the good aspects, as well as potential

developments expressed in Section 5.4.4 were pointed by the participants and not by the researcher<sup>31</sup>.

Considering the automation of the process to produce the jigsaw puzzle pieces for a set of selected requirements, this thesis contributed with a prototype tool, Jig3P. This prototype, together with the heuristics, methods and tools proposed in the thesis and synthesised in Section 4.6, show how a complete implementation for this automation can be achieved.

## **6.3 Future work**

### **6.3.1 Realisation of the approach**

Some proposals for the realisation of the approach presented in the thesis are described in Section 4.6. Those tools and methods were not built to work together. This is usually not straightforward, because what the imperfection detection tool gives is not exactly what the tool to construct the communication mechanism needs.

Usually before the challenge of “how to present” there is yet another to “make” the ambiguity and conflict detection and support tools provide the information (about

---

<sup>31</sup> We agree with the participants in almost everything.

imperfection) that can be usable and useful for the task at hands. It would be interesting to explore:

- How can the interlocking shapes represent (e.g. through different sizes of the pieces bubble) the “size” of the conflict?

Due to the possible connection, raised by the experiments done in this research work, between the concomitant existences of ambiguity with conflicts:

- It would be useful to investigate if the hypothesis of this connection between ambiguity and conflict is or not confirmed and in what circumstances it appears (if always, or in certain circumstances);
- If there is a connection it would be interesting to combine the results from ambiguity detection with conflict detection. For instance in the case that an ambiguity is detected but is not connected with a conflict it may be worthwhile to visually mark it.

Jig3P is a prototype tool, thus to be really useful has to be fully implemented. This implementation would certainly benefit from a study of design principles about the following aspects:

- What treatment of text would be better? In our experiments we applied some specific treatments in the requirements text (e.g. bulleted lists, cut non relevant text); thus before implementing a module to perform text treatment it would be correct to investigate first what that treatment should be.
- How to lay down the pieces? This can be automated, using the results from the rank and detection of imperfections (even if it is left to the requirements engineer to choose the layout of the pieces); but it would be useful to

investigate beforehand if and how the puzzle pieces are laid down affects (positively or negatively) the effectiveness and usability of work.

An important consideration for future developments of the implementation of the approach, with methods and automated tools (when appropriate), should take into consideration the integration of RE in the broader context of the development life cycle of software systems, in particular the connections with management requirements and architecture of the system.

Certainly future work should include more experiments. It would be valuable to perform experiments with real stakeholders and software engineers in a system under development.

### **6.3.2 Jigsaw puzzle supported in a digital media**

The possibility to have the jigsaw puzzle pieces in a digital interactive real-time support was very well accepted by the participants, and we do believe it would bring enormous potential to the approach proposed in this thesis.

The jigsaw puzzle metaphor could be implemented with a digital table where anyone could move the pieces around, people could show (or hide) interlocking shapes (as a cue for potential conflicts). It would be possible to write and draw lines on, with different colours for different participants. More sophisticated collaborative functionalities would also be useful. It would be useful to provide functionalities to show/to hide several types of relevant information, which may clutter the visual space

if presented all at the same type. It is known that the input material for RE does not necessarily have to be text. Despite the fact that text is the most common, participants in RE like to draw sketches, scenarios. It can still be useful to have the possibility to attach audio or video material.

Another interesting set of functionalities would be real-time intelligent interaction enabling, for instance, that if people add a requirement, they agree is missing, and this resolves a conflict, then the interlocking shape(s) that showed that conflict could be transformed in perfectly fitting interlocking and the pieces would fit perfectly. Or if people change a requirement this could make a conflict more probable and the respective interlocking shape could become worse fitting.

There are several variations of this idea that could involve tablets, like iPad. Another hypothesis (wished by some participants that do not want to abandon the physical pieces) is to work with both the physical and digital pieces. Recognition (to introduce in digital support) of what is done (for instance written) in physical pieces might be possible.

The exploration of the digitally supported puzzle pieces provides interesting questions for future work. Some of these questions are:

- How will the virtual puzzle adjust according to decisions participants make, like delete a requirement and thus a puzzle piece? In particular, how do interlocking shapes (representing conflicts will adjust)? And what is the impact of changing one requirement on others, even not directly connected?
- How can the interlocking shapes represent, in real-time, the “size” of the conflict?
- What are the kinds of decisions allowed?
- How can the aspects participants appreciated in the cardboard pieces be maintained and adapted to the digital media? For instance, is it

interesting/possible to virtually grab a pair of pieces from the table to a personal tablet (so that that person can focus on those two requirements aside from the main discussion)?

Some other suggestions related to the digital support are listed below. These suggestions have to be carefully studied (and evaluated) to see if and how they may be supported:

- Attach more semantics to the bubbles and dips of the interlocking shapes as “what the concern might provide (to others)” or “what the concern might need”);
- Attach semantics to the spatial relations between pieces, like proximity (can mean more “interaction” in the relationships); and/ or attach semantics to the width of the links drawn between two words (meaning that there are more conflicts);
- Have icons in each piece to illustrate requirements;
- Highlight words/expressions that provoke the conflicts (we initially thought about this but decided to not include, privileging less cluttered pieces), and provide the functionality to hide/to show this information.

### **6.3.3 How much information and how much perfection?**

There are some very interesting and pertinent questions to recall from the work of Easterbrook et al [Easterbrook96] and Nuseibeh et al [Nuseibeh01]. It would be useful to understand how information is used in the decision process, how



imperfection affects the decision problems, and what kind of impact it brings. In particular it would be necessary to study how much information and how much “perfection” is needed in information to make risk-acceptable, informed and cost-effective decisions. A related issue to address would be to know when to invest in information (for example, by collecting more information or prototyping) to make it more “perfect”, or when to tolerate imperfection and continue the development with it but being aware of it.

The answers to these questions would be relevant to the management of imperfection in RE and to the goal of integrating management of imperfection throughout the development life cycle.

#### **6.3.4 Other uses for the jigsaw puzzle metaphor**

Concerning scalability issues, a jigsaw puzzle-based metaphor can be considered as a solution to represent and communicate all the system requirements, as a big map of the system. An interesting extension to the proposed approach would be to represent, using digital support (e.g. a screen, a digital table), system requirements as 2D geometric shapes (rectangles or polygons with more sides to allow more connections), or even 3D geometric shapes. The elicitation of RE could begin with such a polygon representation of the system in an interactive table. The “jigsaw puzzle” could be a view with the possibility to show/hide the interlocking shapes between the geometric shapes representing requirements. Also interesting would be to allow the user to pick some geometric shapes (representing requirements) and the interlocking shapes between the borders of those requirements would be “automatically calculated”. The problem of representing visually all the requirements

of a system is a different one from the problem that is the focus of this thesis, of providing a communication mechanism to handle ambiguity and conflict in stakeholders' consultation meetings. In these meetings the goal is to discuss a small subset of requirements: the jigsaw puzzle metaphor and the method presented were proposed and shown to be effective to serve this goal. To use the idea of the jigsaw puzzle metaphor to represent all the system requirements is a different problem. This has to be studied from the point of view of the needs and utility of such a global view, analyse if and how a jigsaw puzzle metaphor could be adequate, and to be evaluated.

## **6.4 Final remarks**

Since I started to work in software engineering, and in particular in CASE (computer-aided software engineering), I had difficulty in dealing with the imperfection of tools. It appeared that the authors of tools had a more or less vague idea of the users and the task to be performed.

To illustrate the type of contribution, I believe this thesis gives to the state of the art in software engineering; I would like to borrow the following phrases from Nahum Gershon [Gershon98]:

«Life is not perfect. No two users are alike.

Need methods for visualization management that allow tailoring visualization to particular problems and users and thus to make them less imperfect».

I believe this research work on handling of ambiguity and conflict in RE consultation meetings offers a stepping stone by making visually explicit, the usually implicit meta-information (in this thesis ambiguity and conflict) about the system to be built.

The jigsaw puzzle metaphor and associated method have shown to be useful and effective, for the handling of ambiguity and conflict in RE. This happened exactly because they were designed to respect both the characteristics of the task, to be used for, and its users.

## **Appendix A Crisis Management Systems example**

This appendix contains the complete text of the Crisis Management Systems non-functional requirements [Kienzle09] in its original format, the introductory text (extracted from the original CMS documentation) used in experiments 2 and 3 in jigsaw puzzle and text presentations, and the text used in experiment 2, and in experiment 3 when it was presented as text.

### **A.1 Text used to introduce the jigsaw puzzle set**

The domain of the case study is crisis management systems, i.e., systems that help in identifying, assessing, and handling a crisis situation by orchestrating the communication between all parties involved in handling the crisis, by allocating and managing resources, and by providing access to relevant crisis-related information to authorized users.

### **A.2 Complete text of the non-functional requirements**

The crisis management system shall exhibit the following non-functional properties:

- Availability
  - The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
  - The system shall recover in a maximum of 30 seconds upon failure.
  - Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.
- Reliability
  - The system shall not exceed a maximum failure rate of 0.001%.
  - The mobile units shall be able to communicate with other units on the crisis site and the control centre regardless of location, terrain and weather conditions.
- Persistence
  - The system shall provide support for storing, updating and accessing the following information on both resolved and on-going crises: type of crisis; location of crisis; witness report; witness location; witness data; time reported; duration of resolution; resources deployed; civilian casualties; crisis management personnel casualties; strategies used; missions used; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; log of communications; log of decisions; log of problems encountered.
  - The system shall provide support for storing, updating and accessing the following information on available and deployed resources (both internal and external): type of resource (human or equipment); capability; rescue team; location; estimated time of arrival (ETA) on crisis site.
  - The system shall provide support for storing, updating and accessing the following information on crisis resolution strategies: type of crisis; step-by-step guide to resolve crisis; configuration of missions required; links to alternate strategies; applications to previous crises; success rate.
- Real-time

- The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.
- The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
- The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.
- Security
  - The system shall define access policies for various classes of users. The access policy shall describe the components and information each class may add, access and update.
  - The system shall authenticate users on the basis of the access policies when they first access any components or information. If a user remains idle for 30 minutes or longer, the system shall require them to re-authenticate.
  - All communications in the system shall use secure channels compliant with AES-128 standard encryption.
- Mobility
  - Rescue resources shall be able to access information on the move.
  - The system shall provide location-sensitive information to rescue resources.
  - Rescue resources shall communicate their location to the control centre.
  - The system shall have access to detailed maps, terrain data and weather conditions for the crisis location and the routes leading to it.
- Statistic Logging
  - The system shall record the following statistical information on both on-going and resolved crises: rate of progression; average response time of rescue teams; individual response time of each rescue team;

- success rate of each rescue team; rate of casualties; success rate of missions.
- The system shall provide statistical analysis tools to analyse individual crisis data and data on multiple crises.
  - Multi-Access
    - The system shall support at least 1000 witnesses calling in at a time.
    - The system shall support communication, coordination and information access for at least 20000 rescue resources in deployment at a time.
    - The system shall support management of at least 100 crises at a time.
    - The system shall support management of at least 200 missions per crisis at a time.
  - Safety
    - The system shall monitor emissions from crisis site to determine safe operating distances for rescue resources.
    - The system shall monitor weather and terrain conditions at crisis site to ensure safe operation and withdrawal of rescue resources, and removal of civilians and casualties.
    - The system shall determine a perimeter for the crisis site to ensure safety of civilians and removal of casualties to a safe distance.
    - The system shall monitor criminal activity to ensure safety of rescue resources, civilians and casualties.
    - The safety of rescue personnel shall take top priority for the system.
  - Adaptability
    - The system shall recommend alternate strategies for dealing with a crisis as the crisis conditions (e.g., weather conditions, terrain conditions, civilian or criminal activity) change.
    - The system shall recommend or enlist alternate resources in case of unavailability or shortage of suitable resources.
    - The system shall be able to use alternate communication channels in case of unavailability or shortage of existing channels.
    - The system shall be able to maintain effective communication in areas of high disruption or noise at the crisis site.

- Accuracy
  - The system shall have access to map, terrain and weather data with a 99% accuracy.
  - The system shall provide up-to-date information to rescue resources.
  - The system shall record data upon receipt without modifications.
  - The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.

## **A.3 Text used in experiment 2**

- **Availability**

- The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
- The system shall recover in a maximum of 30 seconds upon failure.
- Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.

- **Reliability**

- The system shall not exceed a maximum failure rate of 0.001%.
- The mobile units shall be able to communicate with other units on the crisis site and the control centre regardless of location, terrain and weather conditions.

- **Real-time**

- The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.



- The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
- The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.
- **Accuracy**
  - The system shall have access to map, terrain and weather data with a 99% accuracy.
  - The system shall provide up-to-date information to rescue resources.
  - The system shall record data upon receipt without modifications.
  - The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.

## **A.4 Text used in experiment 3**

- **Availability**
  - The system shall be in operation 24 hours a day, everyday, without break, throughout the year except for a maximum downtime of 2 hours every 30 days for maintenance.
  - The system shall recover in a maximum of 30 seconds upon failure.
  - Maintenance shall be postponed or interrupted if a crisis is imminent without affecting the systems capabilities.
- **Real-time**
  - The control centre shall receive and update the following information on an on-going crisis at intervals not exceeding 30 seconds: resources deployed; civilian casualties; crisis management personnel casualties; location of super observer; crisis perimeter; location of rescue teams on crisis site; level of emissions from crisis site; estimated time of arrival (ETA) of rescue teams on crisis site.

- The delay in communication of information between control centre and rescue personnel as well as amongst rescue personnel shall not exceed 500 milliseconds.
- The system shall be able to retrieve any stored information with a maximum delay of 500 milliseconds.

*Detected conflicts between Availability and Real-time*

- **Reliability**

- The system shall not exceed a maximum failure rate of 0.001%.
- The mobile units shall be able to communicate with other units on the crisis site and the control centre regardless of location, terrain and weather conditions.

*Detected conflicts between Availability and Reliability*

- **Accuracy**

- The system shall have access to map, terrain and weather data with a 99% accuracy.
- The system shall provide up-to-date information to rescue resources.
- The system shall record data upon receipt without modifications.
- The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.

*Detected conflicts between Reliability and Accuracy*

*Detected conflicts between Real-time and Accuracy*

## **Appendix B Health Watcher example**

This appendix contains the introductory text, extracted from the original Health-Watcher documentation [Soares06], used in experiments 2 and 3 in jigsaw puzzle and text presentations, and the text for the requirements used in experiment 2, and in experiment 3 when it was presented as text.

### **B.1 Text used to introduce the jigsaw puzzle set**

This document specifies the requirements for the City Hall Public Health System named HEALTH-WATCHER.

The purpose of the system is to collect then manage public health related complaints and notifications. The system is also used to notify people about important information regarding the Health System.

### **B.2 Text used in experiment 2**

This document specifies the requirements for the City Hall Public Health System named HEALTH-WATCHER.

The purpose of the system is to collect then manage public health related complaints and notifications. The system is also used to notify people about important information regarding the Health System.

### **Availability**

The system should be available 24 hours a day, 7 days a week. The nature of the system not being a critical system, the system might stay off until any fault is fixed.

### **Performance**

The system must be capable to handle 20 simultaneous users.  
The response time must not exceed 5 seconds.

### **Security**

The system should use a security protocol when sending data over the internet.  
To have access to the complaint registration features, access must be allowed by the access control sub-system.

### **Standards**

The system must be developed according to the standards established by X<sup>1</sup>, responsible for the norms and standardization of systems for the City Hall.

<sup>1</sup> The company name is confidential due to commercial reasons.

## **B.3 Text used in experiment 3**

This document specifies the requirements for the City Hall Public Health System named HEALTH-WATCHER.

The purpose of the system is to collect then manage public health related complaints and notifications. The system is also used to notify people about important information regarding the Health System.

### **Availability**

The system should be available 24 hours a day, 7 days a week. The nature of the system not being a critical system, the system might stay off until any fault is fixed.

### **Performance**

The system must be capable to handle 20 simultaneous users.

The response time must not exceed 5 seconds.

*Detected conflicts between Availability and Performance*

### **Security**

The system should use a security protocol when sending data over the internet.

To have access to the complaint registration features, access must be allowed by the access control sub-system.

*Detected conflicts between Performance and Security*

## **Standards**

The system must be developed according to the standards established by X<sup>1</sup>, responsible for the norms and standardization of systems for the City Hall.

<sup>1</sup> The company name is confidential due to commercial reasons.

*Detected conflicts between Standards and Security and Standards and Performance*

## Appendix C CAS example

This appendix contains the introductory text, extracted from the CAS system [Ayed09], used in experiments 2 in jigsaw puzzle and text presentations, and the text for the requirements used in experiment 2 when CAS was presented as text. Both for the introductory text and for the complete text, the researcher introduced the explanation of some abbreviations that were not clear. These are marked.

### C.1 Text used to introduce the jigsaw puzzle set

#### CAS industrial case: service mash-ups in a hosted SaaS CRM application

##### 1. CAS acronyms and terminology

- **SaaS:** Software-as-a-Service, a software delivery model, where clients use hosted software (usually over the Web) and pay a regular fee for the actual use of the software.
- **Client:** A commercial party paying a regular fee to use a SaaS system
- **User:** Representative of the client who is entitled to use the SaaS system. She has an account and password.
- **CRM:** (customer relationship management) is a type of business software that standardises sales processes and customer services.
- **Hosting company:** Company offering hosting infrastructure like server machines, data centre facilities etc. as well as hosting services as bandwidth, operation support and surveillance, software update and maintenance etc.

<Added by Maria Albuquerque:

**Mash-up:** Web page or application that uses and combines data, presentation or functionality from 2 or more sources to create new services

**RIA:** Rich Internet Application

**WAP:** Wireless Application Protocol>

## 2. Purpose

The main purpose of CRM is to make customer information transparently available – always and anywhere throughout the entire company.

## C.2 Text used in experiment 2

### CAS industrial case: service mash-ups in a hosted SaaS CRM application

#### 1. CAS acronyms and terminology

- **SaaS:** Software-as-a-Service, a software delivery model, where clients use hosted software (usually over the Web) and pay a regular fee for the actual use of the software.
- **Client:** A commercial party paying a regular fee to use a SaaS system
- **User:** Representative of the client who is entitled to use the SaaS system. She has an account and password.
- **CRM:** (customer relationship management) is a type of business software that standardises sales processes and customer services.



- **Hosting company:** Company offering hosting infrastructure like server machines, data centre facilities etc. as well as hosting services as bandwidth, operation support and surveillance, software update and maintenance etc.

<Added by Maria Albuquerque:

**Mash-up:** Web page or application that uses and combines data, presentation or functionality from 2 or more sources to create new services

**RIA:** Rich Internet Application

**WAP:** Wireless Application Protocol>

## 2. Purpose

The main purpose of CRM is to make customer information transparently available – always and anywhere throughout the entire company.

## 3. Requirements

**Availability:** SaaS applications promise a 24/7 availability from anywhere at any time.

All subsystems must be available at any time.

**Security:** Security and privacy are of high importance. The highest available security is always used. If a service requires a certain level of security (e.g. encrypting), which is not available, then the whole service is not available.

Encryption (SSL) is used whenever possible. Communication between RIA GUI and GUI server is always encrypted (including authentication).

**Multi-channel access:** SaaS applications offered through the Internet are usually supporting different interaction modes including classic page-oriented HTML GUIs, rich internet GUIs (e.g., AJAX) as well as access channels for

mobile users such as WAP, data replication for offline use or speech control (e.g., to operate applications through a normal telephone).

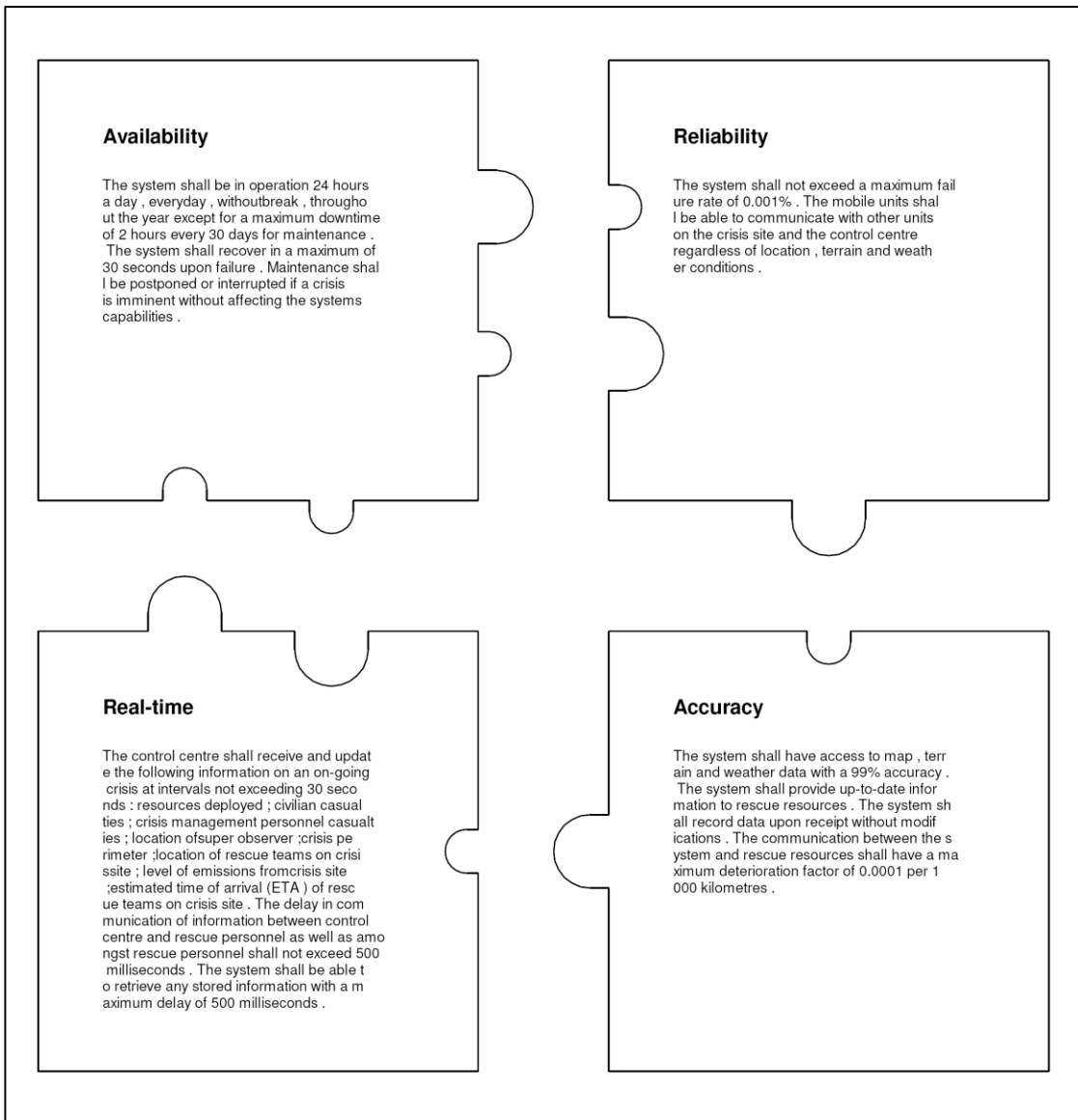
**Accurate and up-to-date information:** CRM systems have to support a broad range of working patterns, ranging from back-office support at company premises to mobile or on site work at the customer. The success of CRM is strongly dependent on the availability of accurate and up-to-date information about customers and easy access to helpful services. It is very important to the user to have the same information available anytime, at any place, even if he is not at his personal computer in his office, for example because he is on a business trip.

## Appendix D Jig3P tool prototype

Jig3P is a (prototype) tool to produce a set of jigsaw puzzle pieces that represent a set of requirements from the documentation of a system, according to the jigsaw puzzle metaphor described in Section 4.4.

Jig3P receives requirements documents in RDL (Requirements Definition Language), which are in fact XML documents. The prototype was developed in MATLAB [MATLAB] and uses a function to import XML documents. This function reads a XML file and returns the content of the file in a Document Object Model (DOM) node. Using MATLAB functions to deal with DOM it is possible to select just some parts of the RDL document, according to XML label contents. This way in Jig3P, it is possible to extract the text that belongs to specific requirements. MATLAB geometric capabilities are then used to draw the puzzle pieces (at this moment just 4 and in a grid) with the respective requirement text inside. For the pairs of pieces (representing requirements) that share a border and have conflicts (detected), the prototype draws interlocking shapes that do not fit well with each other.

A picture of the output of Jig3P, for the requirements Availability, Reliability, Real-time, and Accuracy of CMS, is presented in Figure D-1.



**Figure D-1 Picture of the output of Jig3P, for the requirements Availability, Reliability, Real-time, and Accuracy of CMS.**

The code of Jig3P is presented below.

```
=====
Jig3P.m
-----
```

```
% Program to Read the Requirements Text of a List of Concerns
```

```
% It requests the user to input the identification of the file that
```

```
% contains the concerns (in RDL format) and the vector with the names of
```

```
% the concerns for which we want to retrieve the requirements text
```

```
clear      % Clear all variables
```

```
clf        % Clear drawing window
```

```
% To do: ask the file name and concern names to the user
```

```
docId=xmlread('crisis_mgm_NFR.RDL'); % with no ; should print docID=[#document:
null]
```

```
v_offset=2.0;
```

```
text_rect_lines = 20;
```

```
text_rect_columns = 42;
```

```
MyFontSize=5;
```

```
MyFont='Arial';
```

```
concernNames=char('Real-time');
```

```
[ConcernName,ConcernText] =getConcernsText2string(docId, concernNames);
```

```
CN=char(ConcernName);
```

```
CT=char(ConcernText);
```

```
CT(find(double(CT)==10))=' ';
```

```
CT=removeSpaces(CT);
```

```

s=string2rectangle(CT,text_rect_lines,text_rect_columns);

center=[1 1]; % Square center; Matrix with 1*2

radius=3; % Square radius; Matrix with 1*1

sideTypes=[28 1 1 2]; %

showSquare(center, radius, sideTypes)

text(-
0.5,0+v_offset,s,'FontSize',MyFontSize,'FontName',MyFont,'VerticalAlignment','top')

text(-
0.5,0.5+v_offset,CN,'FontSize',MyFontSize+2,'FontName',MyFont,'FontWeight','bold',
'VerticalAlignment','top')

hold on

concernNames=char('Accuracy');

[ConcernName,ConcernText] =getConcernsText2string(docId, concernNames);

CN=char(ConcernName);

CT=char(ConcernText);

CT(find(double(CT)==10))=' ';

CT=removeSpaces(CT);

s=string2rectangle(CT,text_rect_lines,text_rect_columns);

center=[6.5 1]; % Square center; Matrix with 1*2

radius=3; % Square radius; Matrix with 1*1

sideTypes=[ 2 5 1 1 ]; %

showSquare(center, radius, sideTypes)

text(-
0.5+5.5,0+v_offset,s,'FontSize',MyFontSize,'FontName',MyFont,'VerticalAlignment','t
op')

text(-
0.5+5.5,0.5+v_offset,CN,'FontSize',MyFontSize+2,'FontName',MyFont,'FontWeight','
bold','VerticalAlignment','top')

hold on

```

```

concernNames=char('Availability');

[ConcernName,ConcernText] =getConcernsText2string(docId, concernNames);

CN=char(ConcernName);

CT=char(ConcernText);

CT(find(double(CT)==10))=' ';

CT=removeSpaces(CT);

s=string2rectangle(CT,text_rect_lines,text_rect_columns);

center=[1 6.5]; % Square center; Matrix with 1*2

radius=3; % Square radius; Matrix with 1*1

sideTypes=[ 1 1 22 17 ]; %

showSquare(center, radius, sideTypes)

text(-
0.5,0+5.5+v_offset,s,'FontSize',MyFontSize,'FontName',MyFont,'VerticalAlignment','t
op')

text(-
0.5,0.5+5.5+v_offset,CN,'FontSize',MyFontSize+2,'FontName',MyFont,'FontWeight',
bold','VerticalAlignment','top')

hold on

```

```

concernNames=char( 'Reliability');

[ConcernName,ConcernText] =getConcernsText2string(docId, concernNames);

CN=char(ConcernName);

CT=char(ConcernText);

CT(find(double(CT)==10))=' ';

CT=removeSpaces(CT);

s=string2rectangle(CT,text_rect_lines,text_rect_columns);

center=[6.5 6.5]; % Square center; Matrix with 1*2

radius=3; % Square radius; Matrix with 1*1

sideTypes=[ 1 16 5 1 ]; %

showSquare(center, radius, sideTypes)

```

```
text(-
0.5+5.5,0+5.5+v_offset,s,'FontSize',MyFontSize,'FontName',MyFont,'VerticalAlignme
nt','top')
```

```
text(-
0.5+5.5,0.5+5.5+v_offset,CN,'FontSize',MyFontSize+2,'FontName',MyFont,'FontWeig
ht','bold','VerticalAlignment','top')
```

```
axis([-1.2 8.7 -1.2 8.7])
```

```
axis off
```

```
=====
```

```
getConcernsText2string.m
```

```
-----
```

```
% Module getConcernsText
```

```
% Input: The document node (DOM) of the file that contains the concerns
```

```
% (in XML/RDL format) and
```

```
% a vector with the names of the concerns for which we want to
```

```
% retrieve the text
```

```
% Description: Displays in the screen the complete text of a set of
```

```
% concerns
```

```
function [ConcernName,ConcernText]=getConcernsText2string(docId,
concernNames)
```

```
allConcernElements = docId.getElementsByTagName('Concern');
```

```
for k = 0:allConcernElements.getLength-1
```

```
    thisConcernElement = allConcernElements.item(k);
```

```
    thisConcernAttributes = thisConcernElement.getAttributes();
```

```
    thisConcernNameValue =
```

```
thisConcernAttributes.getNamedItem('name').getNodeValue();
```

```
    if (belongsStringToVector(thisConcernNameValue, concernNames))
```



```

        s1=thisConcernNameValue;
        thisConcernText = thisConcernElement.getTextContent();
        s2=thisConcernText;
    end
end
ConcernName=s1;
ConcernText=s2;

```

```
=====
```

```
belongsStringToVector.m
```

```
-----
```

```

% Module belongsStringToVector
% Says (true, false) if a concernName belongs to a vector of concernNames
function found = belongsStringToVector(thisConcernName, concernNames)
k=1;
found = false;
[rows cols] = size(concernNames); % concernNames size of each dimension
while (found == false && k <= rows)
    if strmatch(thisConcernName, concernNames(k,:), 'exact')
        found = true;
    end
    k = k+1;
end
end

```

```
=====
removeSpaces.m
```

```
-----
function y=removeSpaces(s)
```

```
i=1;
```

```
while s(i)==' '
```

```
    i=i+1;
```

```
end
```

```
a=s(i);
```

```
i=i+1;
```

```
while i<=length(s)
```

```
    if ~(a(end)==' ' & s(i)==' ')
```

```
        a=[a,s(i)];
```

```
    end
```

```
    i=i+1;
```

```
end
```

```
y=a;
```

```
=====
string2rectangle.m
```

```
-----
function s_out=string2rectangle(s,M,N)
```

```
% s is the string to be written
```

```
% M is the number of lines
```

```
% N is the number of columns
```

```
s=s(:);
```

```
L=length(s);
```

```
if L>M*N
```

```
    s=s(1:M*N);
```

```

disp('Warning: text cropped in string2rectangle.m')
else
    for i=1:N*M-L
        s=[s; ' '];
    end
end
end
s=reshape(s,[N,M]);
s_out=s';

```

=====

showSquare.m

-----

```

function showSquare(Center, Radius, sideType)
% Center - 2 component vector
% Raio - scalar
% sideType - 4 component vector
Center=Center(:);
hold on
for k=1:4
    theta1=(k-1)*pi/2+pi/4;
    theta2=k*pi/2+pi/4;
    startPoint(1)=Center(1)+Radius*cos(theta1);
    startPoint(2)=Center(2)+Radius*sin(theta1);
    endPoint(1)=Center(1)+Radius*cos(theta2);
    endPoint(2)=Center(2)+Radius*sin(theta2);
    drawSide(sideType(k), startPoint, endPoint)
end
axis equal
axis off

```

hold off

=====

drawSide.m

-----

```
function drawSide(Type, startPoint, endPoint)
% Type - Graphic type of the side
% startPoint - Starting point of the side
% endPoint - End point of the side

side = endPoint - startPoint; % Vector
knots=1/6*[0:6];
knots=ones(7,1)*startPoint+[knots' knots'].*(ones(7,1)*side);
switch Type
    case 1    % Simple line
        drawLine(startPoint,endPoint)
    case 2    %
        drawLine(startPoint, knots(3,:) )
        drawSmallLeftTip( knots(3,:) , knots(5,:) )
        drawLine( knots(5,:) , endPoint)
    case 3    %
        drawLine(startPoint, knots(3,:) )
        drawSmallRightTip( knots(3,:) , knots(5,:) )
        drawLine( knots(5,:) , endPoint)
    case 4    %
        drawLine(startPoint, knots(3,:) )
        drawLargeLeftTip( knots(3,:) , knots(5,:) )
        drawLine( knots(5,:) , endPoint)
    case 5    %
```

```

drawLine(startPoint, knots(3,:) )
drawLargeRightTip( knots(3,:) , knots(5,:) )
drawLine( knots(5,:) , endPoint)

case 6    %
drawLine(startPoint, knots(2,:) )
drawSmallLeftTip( knots(2,:) , knots(4,:) )
drawLine( knots(4,:) , endPoint)

case 7
drawLine(startPoint, knots(2,:) )
drawSmallRightTip( knots(2,:) , knots(4,:) )
drawLine( knots(4,:) , endPoint)

case 8
drawLine(startPoint, knots(2,:) )
drawLargeLeftTip( knots(2,:) , knots(4,:) )
drawLine( knots(4,:) , endPoint)

case 9
drawLine(startPoint, knots(2,:) )
drawLargeRightTip( knots(2,:) , knots(4,:) )
drawLine( knots(4,:) , endPoint)

case 10   %
drawLine(startPoint, knots(4,:) )
drawSmallLeftTip( knots(4,:) , knots(6,:) )
drawLine( knots(6,:) , endPoint)

case 11
drawLine(startPoint, knots(4,:) )
drawSmallRightTip( knots(4,:) , knots(6,:) )
drawLine( knots(6,:) , endPoint)

case 12
drawLine(startPoint, knots(4,:) )

```

drawLargeLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 13

drawLine(startPoint, knots(4,:) )

drawLargeRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 14

drawLine(startPoint, knots(2,:) )

drawSmallLeftTip( knots(2,:) , knots(4,:) )

drawSmallLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 15

drawLine(startPoint, knots(2,:) )

drawSmallRightTip( knots(2,:) , knots(4,:) )

drawSmallRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 16

drawLine(startPoint, knots(2,:) )

drawSmallLeftTip( knots(2,:) , knots(4,:) )

drawLargeLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 17

drawLine(startPoint, knots(2,:) )

drawSmallRightTip( knots(2,:) , knots(4,:) )

drawLargeRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 18

drawLine(startPoint, knots(2,:) )

drawLargeLeftTip( knots(2,:) , knots(4,:) )

drawSmallLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 19

drawLine(startPoint, knots(2,:) )

drawLargeRightTip( knots(2,:) , knots(4,:) )

drawSmallRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 20

drawLine(startPoint, knots(2,:) )

drawLargeLeftTip( knots(2,:) , knots(4,:) )

drawLargeLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 21

drawLine(startPoint, knots(2,:) )

drawLargeRightTip( knots(2,:) , knots(4,:) )

drawLargeRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 22

drawLine(startPoint, knots(2,:) )

drawSmallLeftTip( knots(2,:) , knots(4,:) )

drawSmallRightTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 23

drawLine(startPoint, knots(2,:) )

drawSmallRightTip( knots(2,:) , knots(4,:) )

drawSmallLeftTip( knots(4,:) , knots(6,:) )

drawLine( knots(6,:) , endPoint)

case 24

drawLine(startPoint, knots(2,:) )

```
drawSmallLeftTip( knots(2,:) , knots(4,:) )  
drawLargeRightTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

case 25

```
drawLine(startPoint, knots(2,:) )  
drawSmallRightTip( knots(2,:) , knots(4,:) )  
drawLargeLeftTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

case 26

```
drawLine(startPoint, knots(2,:) )  
drawLargeLeftTip( knots(2,:) , knots(4,:) )  
drawSmallRightTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

case 27

```
drawLine(startPoint, knots(2,:) )  
drawLargeRightTip( knots(2,:) , knots(4,:) )  
drawSmallLeftTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

case 28

```
drawLine(startPoint, knots(2,:) )  
drawLargeLeftTip( knots(2,:) , knots(4,:) )  
drawLargeRightTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

case 29

```
drawLine(startPoint, knots(2,:) )  
drawLargeRightTip( knots(2,:) , knots(4,:) )  
drawLargeLeftTip( knots(4,:) , knots(6,:) )  
drawLine( knots(6,:) , endPoint)
```

end



```
=====
drawLine.m
```

```
-----
function drawLine (startPoint,endPoint)
plot( [startPoint(1) endPoint(1)], [startPoint(2) endPoint(2)],'-k')
```

```
=====
drawLargeLeftTip.m
```

```
-----
function drawLargeLeftTip(startPoint,endPoint)
side=endPoint-startPoint;
M=norm(side)/2;
R=M*5/10;
knot1=startPoint+(1/2-5/20)*side;
knot2=startPoint+(1/2+5/20)*side;
leftVector=(1/(2*M))*[-side(2) side(1)]; % leftVector is a normalized vector, orthogonal
to side
leftDisplacement=R/2*leftVector;

hold on
drawLine(startPoint,knot1)
drawLine(knot1,knot1+leftDisplacement)
semicircleLeft(knot1+leftDisplacement,knot2+leftDisplacement)
drawLine(knot2+leftDisplacement,knot2)
drawLine(knot2, endPoint)
```

```
=====
drawSmallLeftTip.m
-----
```

```
function drawSmallLeftTip(startPoint,endPoint)
side=endPoint-startPoint;
M=norm(side)/2;
R=M*3/10;
knot1=startPoint+(1/2-3/20)*side;
knot2=startPoint+(1/2+3/20)*side;
leftVector=(1/(2*M))*[-side(2) side(1)]; % leftVector is a normalized vector, orthogonal
to side
leftDisplacement=R/2*leftVector;

hold on
drawLine(startPoint,knot1)
drawLine(knot1,knot1+leftDisplacement)
semicircleLeft(knot1+leftDisplacement,knot2+leftDisplacement)
drawLine(knot2+leftDisplacement,knot2)
drawLine(knot2, endPoint)
```

```
=====
drawLargeRightTip.m
-----
```

```
function drawLargeRightTip(startPoint,endPoint)
drawLargeLeftTip(endPoint,startPoint)
return
```

```
=====
drawSmallRightTip.m
-----
```

```
function drawSmallRightTip(startPoint,endPoint)
drawSmallLeftTip(endPoint,startPoint)
return
```

```
=====
semicircleLeft.m
-----
```

```
function semicircleLeft(startPoint,endPoint)

center= (startPoint+endPoint)/2;
radius= norm(endPoint-startPoint)/2;
step= endPoint-startPoint; % Vector
angleStepBack = atan2(step(2),step(1));
nSides=12;
% hold on
for k=1:nSides
    theta1=angleStepBack+(k-1)*pi/nSides;
    theta2=angleStepBack+k*pi/nSides;
    x1=center(1)+radius*cos(theta1);
    y1=center(2)+radius*sin(theta1);
    x2=center(1)+radius*cos(theta2);
    y2=center(2)+radius*sin(theta2);
    plot([x1 x2],[y1 y2],'-k');
end
% hold off
```

# **Appendix E Research protocol form**

**TITLE OF RESEARCH: Managing imperfection in requirements engineering**

## **INVESTIGATORS:**

Maria Pinto Albuquerque (PhD student)

Email: maria.albuquerque@iscte.pt

Address: ISCTE – University Institute of Lisbon, Av. Forças Armadas, 1649-026 Lisboa

Tel: +351-217903987

Professor Awais Rashid (Supervisor)

Email: marash@comp.lancs.ac.uk

Address: School of Computing and Communication, Infolab21, Lancaster University, Lancaster LA1 4WA

Tel: +44-1524-510316

Fax: +44-1524-510492

## **INTRODUCTION**

You will be taking part in a research study concerned with understanding how requirement engineers and stakeholders communicate and work together to detect, analyze and solve imperfections present in the requirements documentation for a system. For imperfection we mean things like incompleteness, misplacement, ambiguity, and conflict. We are looking only for conflict of meaning in requirements expression, not conflict due to different interests of stakeholders and/or software engineers.

The research study you are asked to take part in uses observational techniques to study the way you work, in a group, during a technical meeting aimed at performing validation of requirements documentation for a system. This means that we will

undertake observational studies of you and the other members of the group as you review requirements documentation for a system, in particular using a tool based on jigsaw puzzle metaphor. Among other collecting techniques we will use audio and video recording for documenting your experiences.

It is important that you read and understand several principles that apply to all who take part in our studies:

- a) taking part in the study is entirely voluntary;
- b) personal benefit may not result from taking part in the study, but knowledge may be gained that will benefit others;
- c) any significant findings will be discussed with you if you desire;
- d) you may withdraw from the study at any time.

The nature of the study, the risks, inconveniences, discomforts, and other pertinent information about the study are discussed below. You are urged to discuss any questions you have about this study with the investigators before you sign this consent.

In accord with all of our research protocols, privacy will be fully protected and confidentiality maintained at all times.

## **STUDY PROCEDURE**

You are being asked to participate in a study that will require your cooperation in the following. You, the other participants in the study and the investigator(s) will perform a technical meeting (emulating a real one) aimed at reviewing and validating a specific part of the requirements documentation for a system. The investigator(s) will perform the role of a facilitator (emulating the facilitator of a real technical meeting). Before the technical meeting, the investigator will provide information on the systems you will be working with. The technical meeting will consist on the presentation of one jigsaw puzzle piece set. These jigsaw puzzle sets represent a specific part of the requirements documentation for a system (system 1). The goal is to perform group work with the other participants to assemble the jigsaw puzzle and when you face problems in this assemblage activity, try to discover what the puzzle tells you about the requirements and possible imperfections on the requirements.

After working with the jigsaw puzzle set for system 1, you will be presented with a text describing part of the requirements for a second system, and some additional information on the imperfections it might contain (system 2). Now the goal is to read the text and discuss these possible imperfections on the requirements.

Any comments, questions, remarks that came to your mind are valuable, so you are free to speak about them. Remember that this study is not to evaluate you, it is to discover what kind of methods and tools could help you and others review and validate requirements documentation.

The investigator will observe the work. He will take notes of the activities and communication threats and the meeting will be both audio and video recorded. The audio and video recorder and the investigator notebook can be ‘turned off’ or data erased as you wish—your wishes are paramount during the fieldwork stage. The aim of this type of research is to be as non-intrusive as possible, so the investigator will attempt to get in the way as little as possible.

When writing the results from our observations into a project report or any other form of documentation, steps are taken to ensure anonymity for all those involved in the study. Confidentiality will be maintained at all times. Any tape recordings, that are made, are the property of the investigators will be kept in a secure environment and will be destroyed at the conclusion of the research.

## **RISKS OF PARTICIPATION IN THE STUDY**

The risks of participating in this study are minimal. It is the investigators’ intention that your identity in these studies will remain confidential. However, there is a small risk of inadvertent disclosure. In addition, your identity and study findings may be disclosed through legal action. However, disclosure is unlikely to have an adverse effect on you, on your family members, and on your family relationships. Nor is disclosure likely to result in discrimination in hiring, retention, or promotion.

## **BENEFITS OF PARTICIPATION IN THE STUDY**

There may be no personal benefit to you from participating in this project. The benefits of this research may include learning more about how requirements

documentation is reviewed, what kind of methods and tools may be used and how these might be improved.

## **COSTS AND COMPENSATION**

You will not be paid for participating in this study.

There is unlikely to be any cost —financial or other— to you for participation in the study.

While you may initially be a little concerned or embarrassed at being observed our experience is that people soon learn to ignore the investigator and get on with their work. If your participation in the study becomes unwanted or inconvenient at any time you can ask to be withdrawn from the study.

## **CONFIDENTIALITY**

All information collected in this study belongs to the investigator(s) and will be maintained in a confidential manner at Lancaster University. Nobody, other than the investigators above named, will have access to the data. Any tape (audio and video) recordings will be destroyed at the end of the project. Although rare, it is possible that disclosure may be required by law. Otherwise, the information will not be disclosed to third parties without your permission. If the study is published, your name and institution will be kept confidential.

## **PEOPLE TO CONTACT**

If you have further questions related to this research study, you may contact the Investigator

Professor Awais Rashid (Supervisor)

Email: [marash@comp.lancs.ac.uk](mailto:marash@comp.lancs.ac.uk)

Address: School of Computing and Communication, Infolab21, Lancaster University, Lancaster LA1 4WA

Tel: +44-1524-510316

Fax: +44-1524-510492

## **Appendix F Research protocol - consent form**

### **TITLE OF RESEARCH: Managing imperfection in requirements engineering**

I understand that I am free to refuse to participate in this research project or to withdraw my consent and discontinue participation in the project at any time without prejudice.

I understand that I will not be paid to participate in this study.

I have had the opportunity to fully discuss this investigation and the procedure(s) with a study investigator.

All my questions regarding this project have been answered.

I agree to participate in the project as described above.

#### **Subject's printed name**

A COPY OF THIS FORM HAS BEEN GIVEN TO ME \_\_\_\_\_

#### **Date signed**

If I am not satisfied with the manner in which this study is being conducted, I may report (anonymously if I so choose) any complaints to Yvonne Fox, Secretary to the Ethical Committee, Lancaster University by calling +44-1524-592068 , emailing y.fox@lancaster.ac.uk; or addressing a letter to Y.Fox, Ethical Committee, Lancaster University, LA1 4YR, UK.

I have discussed with the subject, (and, if required, the subject's guardian) the procedure(s) described above and the risks involved; I believe he/she understands



the contents of the consent form, and is competent to give a legally effective and informed consent.

**Signature of Investigator**

**Date signed**

# Appendix G Survey about preferences for pieces background and more data from experiment 2

## G.1 Set up

Since there were several complaints during experiment 2 concerning the picture in the background of the jigsaw puzzle pieces, it was decided, between experiment 2 and 3, to perform a survey with the 11 participants of experiment 2. The goal was to know from four possible pictures for the background of the jigsaw puzzle what would be, in their opinion, the best and worst pictures. It was shown to each group of participants the jigsaw puzzle with which they had worked in four versions where the only difference was the picture in the background (in some cases the text was moved in order to be more readable). Figures G-1, G-2, G-3, and G-9 show the jigsaw puzzles with the four different backgrounds for the CMS example.

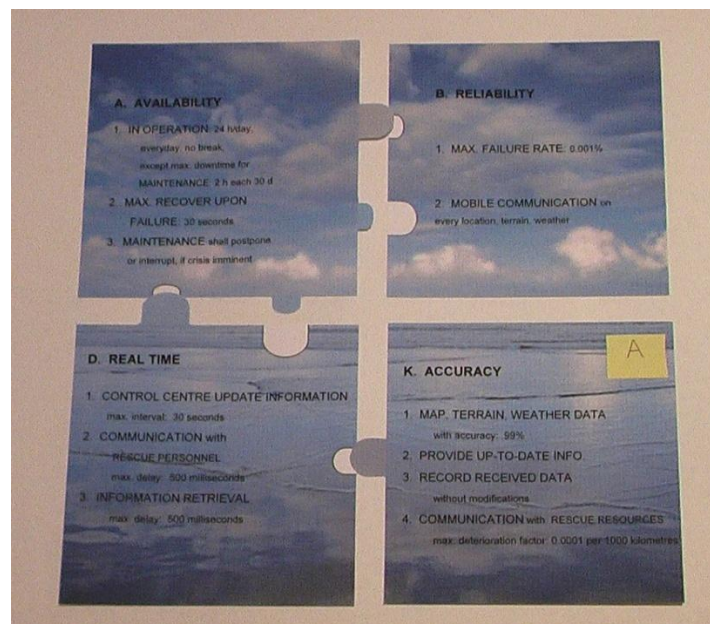


Figure G-1 The jigsaw puzzle for CMS example, with picture A (“clouds and water”).

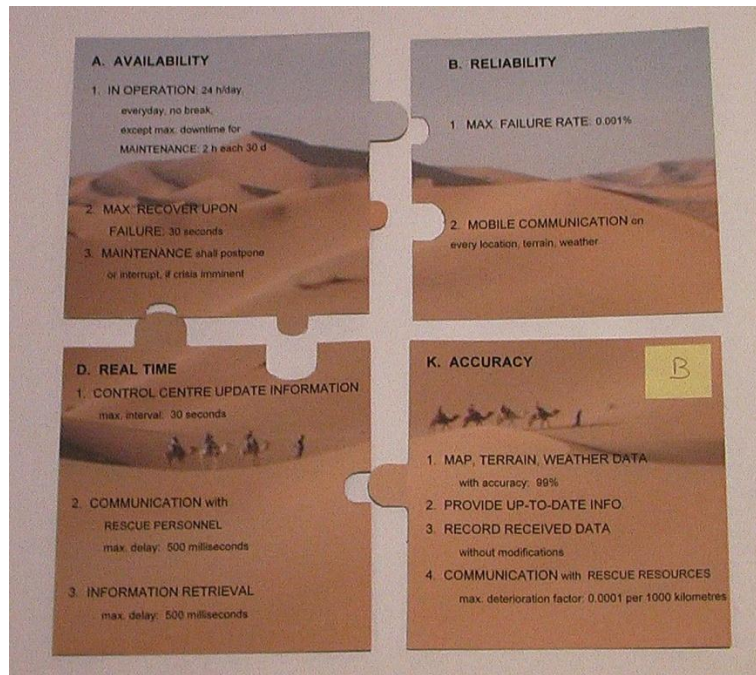


Figure G-2 The jigsaw puzzle for CMS example, with picture B (“desert”).

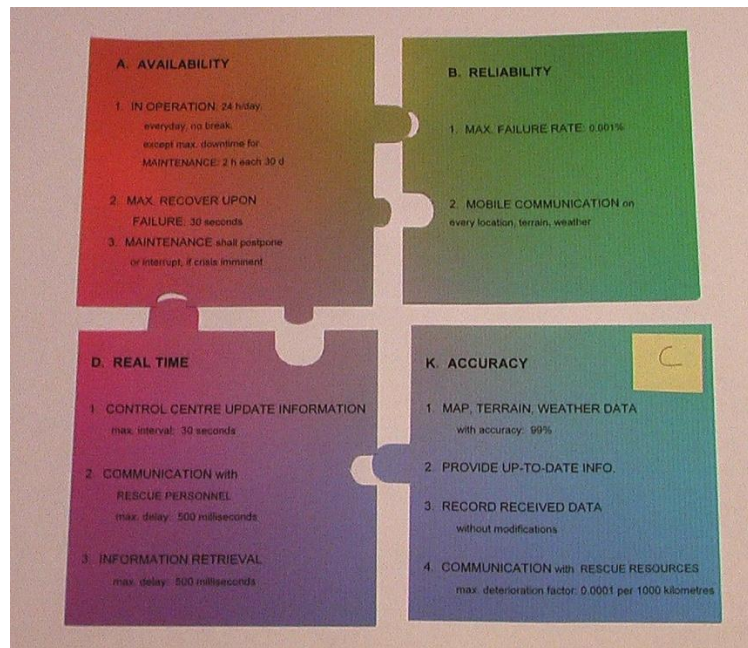


Figure G-3 The jigsaw puzzle for CMS example, with picture C (“red, green, blue gradient”).

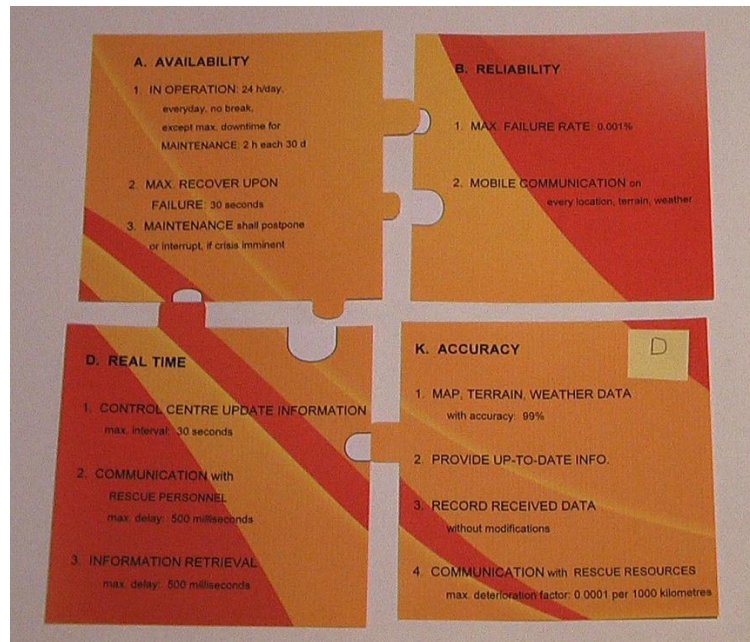


Figure G-4 The jigsaw puzzle for CMS example, with picture D (“orange, yellow abstract”).

## G.2 Analysis of results

It was asked participants to rank the four hypotheses for the background of the jigsaw puzzle from the best to the worst. Then it was attributed 4 points to the best option, 3 to the second best option, then 2, and finally 1 point to the worst option for each participant. The results are summarized in Table G-1 below.

The winning hypothesis was the hypothesis D with the “orange, yellow abstract” picture, closely followed by hypothesis A with the “clouds and water” picture.

It is interesting to note that, in general, half of the persons prefer the nature pictures and the other half the abstract ones.

Session1	A	B	C	D
P1	4	3	1	2
P2	2	1	3	4
P3	2	3	1	4
P4	1	3	2	4
Session2				
P1	3	1	2	4
P2	4	3	2	1
P3	4	3	2	1
P4	4	1	3	2
Session3				
P1	2	1	3	4
P2	3	4	2	1
P3	1	2	3	4
Total	30	25	24	31

**Table G-1 Table showing the number of points each picture obtained for participants preference.**

### **G.3 Threats to validity**

There are no significant threats to validity in this survey.

# **Appendix H Lists of ambiguities and conflicts to discuss in each experiment**

## **H.1 Experiment 1**

### **H.1.1 Imperfections with no visual cue but detected by participants**

Participants in the first experiment discovered **four imperfections** with no visual cues. These were:

1. Assuming CMS demands quite large distances (this is not explicitly written, it is an implicit assumption that is now made explicit), the second and third phrases of Real-time requirement (max. delay of 500ms) can be difficult to comply with. These amounts to two imperfections.
2. There is a conflict between the second phrase of Reliability requirement (“Mobile communication on every location, terrain, weather”) and the fourth phrase of Accuracy requirement (“Communication with rescue resources max deterioration factor: 0.0001 per 1000 kilometres”). Moreover, the second phrase of Reliability requirement is infeasible, and if we recognize that, then we can see the fourth phrase of Accuracy requirement as a specification of the limit to impose.

3. The possible conflicts amongst the phrases of Real-time requirement referred to as “Real-time description requiring times possibly incompatible among them” and **described in Section 4.4.1.**

## H.2 Experiment 2

### H.2.1 Conflicts with visual cue

The CMS jigsaw puzzle pieces were designed with visual cues only for the conflicts (with ID 1, 2 and 4) in the Table H-1.

ID	Name	Description
1	Real-time dependency on Availability incompatible	See Section 4.3.2
2	Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement	See Section 4.3.2
4	Downtime in Availability requirement and failure rate in Reliability requirement demanding incompatible values	Described below and in Section 4.4.1

**Table H-1 Conflicts with a visual cue in the jigsaw puzzle for the CMS system in experiment 2**

The HW example presented cues (either visual or textual) for the following four conflicts:

- Conflict between Availability (second phrase) and Performance requirement (second phrase),

- Conflict between Security requirement with the second phrase of Performance,
- Security requirement conflicts with Standards requirement, and
- The second phrase of Performance may conflict with Standards requirement, which was not represented with an interlocking shape between Performance and Standards, but instead “through” Availability. This is described in more detail below.

When designing the jigsaw puzzle set for the Health-Watcher, it contained a piece representing Performance with possible conflicts with the other three. But the layout with 2x2 cells all with the same size did not allow placing interlocking shapes between Performance and Standards. Thus, the conflict between Standards and Performance was not represented, in experiment 2, through an interlocking shape between the two pieces that represent these requirements. In this case it was placed an interlocking shape between Standards and Availability, with the expectation that “through” Availability the users will consider the possibility of a conflict between Performance and Standards. The lack of satisfaction with this poor design lead us to design some cells not squared but rectangular to allow the docking of three interlocking shapes from three pieces in a common one. This much better design for cases such as the one described, was tried for the CAS example, which is shown below in Figure 5-4. In experiment 3 this design was also applied to the Health-Watcher example.

The CAS example presented cues (either visual or textual) for the following four conflicts:

- Between Availability and Security requirements (they pointed that it can also be said that the relation is very demanding), and



- Between Security and Multi-channel access requirements.
- The conflict between Security (third and fourth phrases) and Multi-channel access requirement (phrase 1b).

## **H.2.2 Session 1 – Imperfections with no visual cues but detected by participants**

In this session the CMS was presented as a jigsaw puzzle (see Figure 5-2). Participants detected the following conflicts and ambiguities, for which there were no visual cues:

1. *“Nothing described concerning up-to-date in Accuracy requirement”.*
2. *“Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?”*
3. There is a conflict between the second phrase of Reliability (“Mobile communication on every location, terrain, weather”) and the first phrase of Accuracy (“The system shall have access to map, terrain and weather data with a 99% accuracy.”). In experiment 1 a conflict was detected between the same second phrase of Reliability but with the fourth phrase of Accuracy instead (“Communication with rescue resources max deterioration factor: 0.0001 per 1000 kilometres”). Moreover in the first experiment participants reported that the second phrase of Reliability requirement is in their opinion infeasible, in which case the fourth phrase of Accuracy requirement can be seen as a specification of the limit to impose.

4. There is ambiguity in the third and fourth phrases of Accuracy (“3. The system shall record data upon receipt without modifications.” and “4. The communication between the system and rescue resources shall have a maximum deterioration factor of 0.0001 per 1000 kilometres.”), that should be clarified.

The second system used was Health-Watcher, presented in textual form (see Appendix B). Participants pointed also the following conflicts/ambiguities (for which there were no visual cues in the corresponding HW jigsaw puzzle):

1. The two phrases of Availability appear to be conflicting.
2. Concerning Performance, they noted that both the words “users” and “response time” were ambiguous (what is meant by these words?); as well as ambiguity about what happens if there are more than 20 users: failure or deterioration of service?

### **H.2.3 Session 2 – Imperfections with no visual cue but detected by participants**

In the second session of experiment 2 first a sub-set of the Health-Watcher requirements was presented through a jigsaw puzzle, shown in Figure 5-3. Participants detected the following conflicts/ambiguities, for which there were no visual cues:

1. The two phrases of Availability appear to be conflicting (which was also detected in session 1 using the textual form).
2. The second phrase of Availability is ambiguous.
3. The two phrases of Security are ambiguous.

The second system used in session 2 was the CAS example, presented in textual form (see Appendix C). Participants detected also the following conflicts/ambiguities, (for which there were no visual cues in the corresponding CAS jigsaw puzzle):

1. The requirement Availability is unfeasible: very demanding, ambiguous.
2. The requirements Availability and Multi-channel access are conflicting.
3. The requirement 'Accurate and up-to-date information' is unfeasible/very demanding, especially anywhere and for off-line use.
4. For the requirement 'Accurate and up-to-date information', the phrase that says: "The success of CRM is strongly dependent on the availability of accurate and up-to-date information about customers and easy access to helpful services" is particularly ambiguous and it even does not seem pertaining to a requirement text.
5. It is ambiguous who is responsible for handling the information, and the interaction. In particular, it should be clarified that there must be a GUI server responsible for handling the interaction between all the interaction modes and the back-end server.
6. It is ambiguous if encryption is necessary for all communications and why is it only defined for RIA GUI and the GUI server. This is a particular aspect that can be connected with the conflict between Security and Multi-channel access requirements, referred above.

## **H.2.4 Session 3 – Imperfections with no visual cues but detected by participants**

In the third session of experiment 2, first a sub-set of the CAS requirements was presented through a jigsaw puzzle, shown in Figure 5-4. The written report contains the following list of detected ambiguous requirements, for which there were no visual cues:

1. The word “highest” in Security requirement (first phrase) is ambiguous.
2. The expression “whenever possible” in Security requirement (third phrase) is ambiguous.
3. It is ambiguous what the full list of interaction modes has to be to support Multi-channel access requirement (phrase 1a).
4. Multi-channel access requirement (phrase 1b) is ambiguous, more information is needed on:
  - what is needed regarding speech control;
  - what data should be replicated for off-line use; and
  - how is implied synchronization of data replication done.
5. The expression “accurate and up-to-date information” in ‘Accurate and up-to-date information’ requirement (first phrase) is ambiguous.
6. The concepts “easy access” and “helpful services” in ‘Accurate and up-to-date information’ requirement (second phrase) are ambiguous.

The second system used in session 3 was the CMS example, presented in textual form (see Appendix A). Participants detected the following conflicts/ambiguities:

1. *“Nothing described concerning up-to-date in Accuracy requirement”.*

2. In in the first phrase of Reliability requirement, how the concept “failure rate” is measured is ambiguous. The participants queried: This is a percentage of what? This is not exactly the ambiguity case *“Is downtime for maintenance in Availability requirement to be accounted for failure rate referred in Reliability requirement?”* but is related.
3. The first phrase of Accuracy is ambiguous: it would be required to know what layers of geographical information are required.
4. The word expressions “without modifications” and “deterioration factor” in Accuracy requirement (phrases 3 and 4) are ambiguous.

## **H.3 Experiment 3**

### **H.3.1 Conflicts with visual cues**

The CMS examples presented cues (either visual or textual) for the conflicts in the Table H-2.

<b>ID</b>	<b>Name</b>	<b>Description</b>
1	Real-time dependency on Availability incompatible	See Section 4.3.2
2	Something required concerning up-to-date information in Real-time requirement and nothing in Accuracy requirement	See Section 4.3.2
4	Downtime in Availability requirement and failure rate in Reliability requirement demanding incompatible values	Described below and in Section 4.4.1
5	Reliability requiring mobile communication with characteristics incompatible with what is required for communication and access to map, terrain , and weather data in the Accuracy requirement	New badly fitting interlocking shape

**Table H-2 Conflicts with a cue in the CMS system in experiment 2**

The HW example presented cues (either visual or textual) for the following four conflicts:

- Conflict between Availability (second phrase) and Performance requirement (second phrase),
- Conflict between Security requirement with the second phrase of Performance,
- Security requirement conflicts with Standards requirement, and
- The second phrase of Performance may conflict with Standards requirement.

### **H.3.2 Session 1 - Imperfections with no cues but detected by participants**

Participants, when working with the CMS presented in jigsaw puzzle, pointed out the following other 8 imperfections:

1. Ambiguity: *“Nothing described concerning up-to-date in Accuracy requirement”*.
2. The second phrase of Real-time requirement might also be in conflict with requirements the second phrase of Reliability and the fourth of Accuracy. These amounts to two imperfections.
3. The first and third phrases requirements of Availability requirement conflict with the second phrase of Reliability requirement. A posteriori, and listening to the audio it was possible to check that second phrase of Reliability was being interpreted as requiring mobile communication every time, which is in fact not written. Thus the researcher should not have accepted the report of this conflict without querying participants on this fact. These amounts to two imperfections.
4. In the Availability requirement there is ambiguity concerning how much time is required for the system to recover, when it is interrupted for maintenance, and if this is the time referred in the second phrase of Availability?
5. It is missing a requirement specifying on how to signal the non-recovery of the system and what should be done in that case.
6. The fourth phrase of Accuracy requirement has ambiguity: what is 0.0001?
7. It is missing a requirement enabling to know if there is a backup system, that runs whenever the main system goes to maintenance, and thus permitting the accomplishment of the first phrase of Reliability requirement. In other words: does failure rate also accounts for maintenance periods?
8. There first and third phrases of Accuracy requirement are ambiguous.

The second system worked was a sub-set of the Health-Watcher presented in text. Participants pointed out the following imperfections, which had no cues:

1. It is missing a requirement specifying that the user should be informed when the system is off-line or is going to be off-line.
2. It is missing a requirement specifying the need for a recovery system when the system enters in fault.
3. The first phrase of Performance requirement conflicts with the first phrase of Availability requirement.

### **H.3.3 Session 2 - Imperfections with no cues but detected by participants**

The first system worked was the CMS presented in text. Participants detected the following imperfections, which had no cues:

1. Ambiguity *"Nothing described concerning up-to-date in Accuracy requirement"*.
2. The first phrase of Accuracy requirement is ambiguous and might not be realistic.
3. It is missing a requirement specifying how the system receives information about a crisis.
4. It is missing a requirement specifying how maintenance should be handled in the event that a crisis is very long,
5. It is missing a requirement specifying what happens if the system does not recover from failure.
6. The second phrase of Real-time requirement might not be accomplished if it has to be used satellite communication.



7. The word “system” in Availability requirement and the word “control center” in Real-time requirement present ambiguity (“system” only appears in the text version): are these words to be interpreted as meaning the same, or not?
8. The word expressions “mobile units” in Reliability requirement and “rescue resources” in Accuracy requirement raise another ambiguity case (“mobile units” only appear in the text version): are these words to be interpreted as meaning the same, or not?
9. If the answer to the ambiguity case 7, above is yes, then there might be a conflict between the second phrase of Reliability and the first and fourth phrases of Accuracy.
10. The fourth phrase of Accuracy requirement is ambiguous, once it is not described what a “deterioration factor of 0.0001” is.

The second system worked on was a sub-set of the Health-Watcher presented through a jigsaw puzzle, shown in Figure 5-6. Participants detected the following other conflicts/ambiguities, for which there were no cues:

1. The word expression “max response time” in the Performance requirement is ambiguous.
2. The first phrase of Availability requirement conflicts with the Standards requirement.
3. The two phrases of Availability requirement are in conflict with each other.
4. The Performance requirement is ambiguous: is the requirement referring to the maximum or minimum number of simultaneous users that should be handled?
5. Concerning Security requirement it is ambiguous what the process that supports access control is (second phrase).

## References

- Akşit01 Akşit, M., and Marcelloni, F.; “Deferring elimination of design alternatives in object-oriented methods”, *Concurrency and Computation: Practice and Experience*, 13: 1247-1279, 2001.
- Andrews97 Andrews, K., Wolte, J. and Pichler, M.; “Information pyramids: A new approach to visualising large hierarchies”, *IEEE Visualization 97*, 1997.
- Andrews98 Andrews, K. and Heidegger, H.; “Information slices: Visualising and exploring large hierarchies using cascading, semi-circular discs” (late breaking hot topic paper), *IEEE Symposium on Information Visualization (INFOVIS'98)*, pp. 9–12, 1998.
- Ayed09 Ayed, D., Genssler, T.; *Dynamic Variability in complex, Adaptive systems*, Deliverable D6.1 of DiVA EC project, 2009.
- Balzer91 Balzer, R.; “Tolerating Inconsistency”, *13<sup>th</sup> International Conference on Software Engineering*, pp. 158–163, 1991.
- Balzer04 Balzer, M., Noack, A., Deussen, O., and Lewerentz, C.; “Software landscapes: Visualizing the Structure of Large Software Systems”, *VisSym 2004, Symposium on Visualization*, pp. 261–266, Eurographics Association, 2004.
- Bass03 Bass, L., Clements, P., and Kazman, R.; *Software Architecture in Practice*, 2nd edition, Addison-Wesley, 2003.
- Bederson02 Bederson, B.B., Shneiderman, B., and Wattenberg, M.; “Ordered and Quantum Treemaps: Making Effective Use of 2D

- Space to Display Hierarchies”, *ACM Transactions on Graphics (TOG)*, 21(4): 833-854, 2002.
- Berg05 Berg, K. van den, Conejero, J. M., and Chitchyan, R., AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation, AOSD-Europe, 2005.
- Berry03 Berry, D. M., Kamsties, E., and Krieger, M. M.; “From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity”, <https://cs.uwaterloo.ca/~dberry/handbook/ambiguity/Handbook.pdf>, November, 2003.
- Berry08 Berry, D. M.; “Ambiguity in Natural Language Requirements Documents”, *Innovations for Requirement Analysis. From Stakeholders’ Needs to Formal Designs, LNCS 5320*, pp 1-7, 2008.
- Boccuzzo07 Boccuzzo, S., and Gall, H.C; “Cocoviz: Towards cognitive software visualization”, *IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2007.
- Boehm88 Boehm, B. W.; “A Spiral Model of Software Development and Enhancement”, *IEEE Computer*, 21(5): 61-72, 1988.
- Boehm89 Boehm, B. W., and Ross, R.; “Theory-W Software Project Management: Principles and Examples”, *IEEE Transactions on Software Engineering*, 15(7): 902-916, 1989.
- Bresciani08 Bresciani S., and Eppler M.; “The Risks of Visualization. A Classification of Disadvantages Associated with Graphic Representations of Information”, *ICA Working Paper #1/2008*, University of Lugano (USI), 2008.

- Burge08 Burge, J.E., Carroll, J.M., McCall R. and Mistrík, I.; *Rationale-Based Software Engineering*, Springer-Verlag, 2008.
- Buxton07 Buxton, B.; *Sketching User Experiences: Getting the Design Right and the Right Design*, Morgan Kaufmann Publishers, 2007.
- Chantree06 Chantree, F., Nuseibeh, B., de Roeck, A., and Willis, A.; "Identifying Nocuous Ambiguities in Natural Language Requirements", *14th IEEE International Requirements Engineering Conference*, 2006.
- Charters02 Charters, S. M., Knight, C., Thomas, N., and Munro, M.; "Visualisation for informed decision making; from code to components", *International Conference on Software Engineering and Knowledge Engineering (SEKE '02)*, pp. 765–772, ACM Press, 2002.
- Checkland81 Checkland, P; *Systems Thinking, Systems Practice*, John Wiley & Sons, [rev 1999 ed], 1981.
- Chitchyan07 Chitchyan, R., Rashid, A., Rayson, P., and Waters R.; "Semantics-based Composition for Aspect-Oriented Requirements Engineering", *6th International Conference on Aspect-Oriented Software Development*, ACM, 2007.
- Christophe11 Christophe, F., Wang M., Coatanéa, E., Zeng, Y., and Bernard, A.; "Grammatical and Semantic Disambiguation of Requirements at Elicitation and Representation Stages", *23 rd International Conference on Design Theory and Methodology*, pp. 17-31, ASME, 2011.

- Christophe12      Christophe, F.; *Semantics and Knowledge Engineering for Requirements and Synthesis in Conceptual Design*, Doctoral Dissertations 90/2012, Aalto University, Finland, 2012.
- Chung00            Chung, L., Nixon, B. A., Yu, E., and Mylopoulos, J.; *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.
- Conklin88         Conklin, J. and Begeman, M.; "gIBIS: A hypertext tool for exploratory policy discussion", *ACM Transactions on Office Information Systems*, 6(4):303-331, 1988.
- Conklin89         Conklin, J., and Begeman, M.; "gIBIS: A tool for all reasons", *Journal of the American Society for Information Science*, 40(3): 200-213, 1989.
- Creswell02        Creswell, J.W.; *Research design: Qualitative, quantitative and mixed methods approaches*, 2nd Edition, Sage Publications, 2002.
- Cross98            Cross, J.H., Hendrix, T.D. and Maghsoodloo, S.; "The control structure diagram: An overview and initial evaluation", *Empirical Software Engineering*, 3:131–158, 1998.
- Cysneiros03       Cysneiros, L.M., Yu, E., and Leite, J.C.S.P.; "Cataloguing Non Functional Requirements as Softgoals Networks", *Workshop on Requirements Engineering for Adaptable Architectures*, 11<sup>th</sup> IEEE International Conference on Requirements Engineering, pp. 13-20, 2003.

- Davis89 Davis, J. S.; "Identification of errors in software requirements through use of automated requirements tools", *Information and Software Technology*, 31(9): 472-476, 1989.
- Davis93 Davis, A.; *Software Requirements: Objects, Functions, and States*, Prentice Hall, 1993.
- Diehl07 Diehl, S.; *Software Visualization*, Springer, 2007.
- Dix93 Dix, A., Finlay, J., Abowd, G. and Beale, R.; *Human-Computer Interaction*, Prentice Hall, 1993.
- DOORS IBM Rational® DOORS®, <http://www.telelogic.com/products/doors/index.cfm> (last accessed on Sept, 25, 2009).
- Easterbrook91 Easterbrook, S.; *Elicitation of Requirements from Multiple Perspectives*, PhD Thesis, Imperial College of Science Technology and Medicine, University of London, June 1991.
- Easterbrook91a Easterbrook, S. M.; "Handling Conflict Between Domain Descriptions With Computer-Supported Negotiation", *Knowledge Acquisition: An International Journal*, Vol 3, pp.255-289, 1991.
- Easterbrook93 Easterbrook, S.; "Domain Modelling with Hierarchies of Alternative Viewpoints", *First IEEE International Symposium on Requirements Engineering (RE'93)*, 1993.
- Easterbrook94 Easterbrook, S., Finkelstein, A., Kramer, J., and Nuseibeh, B.; "Coordinating Distributed ViewPoints: The anatomy of a consistency check", *Journal of Concurrent Engineering:*

*Research and Applications*, Vol 2, No 3 (Special Issue on Conflict Management), 1994.

- Easterbrook96 Easterbrook, S., and Nuseibeh, B.; "Using ViewPoints for Inconsistency Management", *Software Engineering Journal*, 11: 31-43, BCS/IEE Press, 1996.
- Easterbrook07 Easterbrook, S. M., Singer, J., Storey, M., and Damian, D.; "Selecting Empirical Methods for Software Engineering Research", in Shull, F. and Singer, J. (eds) *Guide to Advanced Empirical Software Engineering*, Springer, 2007.
- Eick92 Eick, S. G., Steffen, J. L. and Summner Jr., E. E.; "Seesoft-a tool for visualizing line oriented software statistics", *IEEE Transactions on Software Engineering*, 18(11): 957-968, 1992.
- Eick00 Eick, S. G.; "Visual Discovery and Analysis", *IEEE Transactions on Visualization and Computer Graphics*, 6(1): 44-58, 2000.
- Eppler04 Eppler, M. J.; "Facilitating Knowledge Communication through Joint Interactive Visualization", *Journal of Universal Computer Science*, 10(6): 683-690, 2004.
- Fagan76 Fagan, M. E.; "Design and code inspections and process control in the development of programs", *IBM Systems Journal*, 15(3): 182-211, 1976.
- Feather06 Feather, M.S., Cornford, S.L., Kiper, J.D. and Menzies, T.; "Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation",

*First International Workshop on Requirements Engineering Visualization, 2006.*

- Fickas85 Fickas, S.; "A knowledge-based approach to specification acquisition and construction", Technical Report CIS-TR-85-13, University of Oregon, Eugene, 1985.
- Fickas88 Fickas, S., Nagarajan, P.; "Being Suspicious: Critiquing Problem Specifications", 7<sup>th</sup> National Conference on Artificial Intelligence, pp. 19-24, 1988.
- Fickas92 Fickas, S., and Helm, R.; "Knowledge Representation and Reasoning in the Design of Composite Systems", *IEEE Transactions on Software Engineering*, 18(6): 470-482, 1992.
- Finkelstein92 Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., and Goedicke, M.; "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development", *International Journal of Software Engineering and Knowledge Engineering*, 2(1): 31-58, 1992.
- Finkelstein94 Finkelstein, A., Gabbay, D., Hunter, A., Kramer J., and Nuseibeh, B.; "Inconsistency handling in multiperspective specifications", *IEEE Transactions on Software Engineering*, 20(8): 569-578, 1994.
- Fuchs99 Fuchs, N., Schwertel, U., Schwitter, R.; "Attempto controlled english (ace) language manual version 3.0", Technical Report No. 99.03, Institut für Informatik der Universität Zürich, Switzerland, 1999.



- Gershon98 Gershon, N.; "Visualization of an Imperfect World", *IEEE Computer Graphics and Applications*, July/August, pp. 43-45, 1998.
- Gleich10 Gleich, B., Creighton, O., and Kof, L.; "Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources", *16th International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 218-232, 2010.
- Glinz00 Glinz, M.; "Problems and Deficiencies of UML as a Requirements Specification Language", *10th International Workshop on Software Specification and Design*, pp. 11–22, 2000.
- Goguen92 Goguen, J.A.; "The Dry and the Wet: IS Concepts", *IFIP Working Group 8.1 Conference*, pp. 1–17, 1992.
- Goguen94 Goguen, J.A.; "Requirements Engineering as the Reconciliation of Technical and Social Issues", *Requirements Engineering: Social and Technical Issues*, Goguen, J.A. and Jirotko, M. (eds.), Academic Press, pp. 165–199, 1994.
- Gotel07 Gotel, O.C.Z., Marchese, F.T. and Morris, S.J.; "On Requirements Visualization", *2<sup>nd</sup> International Workshop on Requirements Engineering Visualization*, 2007.
- Gotel08 Gotel, O.C.Z., Marchese, F.T., and Morris, S.J.; "The Potential Synergy between Information Visualization and Software Engineering Visualization", *12<sup>th</sup> International Conference on Information Visualization*, 2008.

- IEEE-CS04 IEEE Computer Society; *Guide to the Software Engineering Body of Knowledge (SWEBOK®)*, 2004 Version, Abran, A. Moore, J. W., Bourque, P. Dupuis, R. (eds.), The Institute of Electrical and Electronics Engineers, 2004.
- IEEE-SA98 IEEE-SA Standards Board; *IEEE Recommended Practice for Software Requirements Specifications, IEEE Std. 830-1998*, The Institute of Electrical and Electronics Engineers, 1998.
- ISO11 International Standard Organization (ISO); *ISO/IEC 25010:2011: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) - - System and software quality models*, International Standard Organization, 2011.
- Jackson75 Jackson, M.; *Principles of Program Design*, Academic Press, 1975.
- Johnson91 Johnson, B. and Shneiderman, B.; "Tree-maps: A space-filling approach to the visualization of hierarchical information structures", *IEEE Visualization Conference*, pp. 284–291, 1991.
- Kaindl93 Kaindl, H.; "The missing link in requirements engineering", *SIGSOFT Softw. Eng. Notes*, 18(2): 30-39, 1993.
- Kamsties01 Kamsties, E., Berry, D., and Paech, B.; "Detecting ambiguities in requirements documents using inspections", *1<sup>st</sup> Workshop on Inspection in Software engineering*, pp. 68-80, 2001.
- Kamsties05 Kamsties, E.; "Understanding Ambiguity in Requirements Engineering"; In Aurum, A., and Wohlin, C. (eds.), *Engineering and Managing Software Requirements*, Springer-Verlag, 2005.

- Kienzle09 Kienzle, J., Guelfi, N., and Mustafiz, S.; “Crisis Management Systems: A Case Study for Aspect-Oriented Modeling”, <http://www.cs.mcgill.ca/~joerg/taosd/TAOSD/TAOSD.html>, May, 2009.
- Kiyavitskaya08 Kiyavitskaya, N., Zeni, N., Mich, L., and Berry, D.M.; “Requirements for tools for ambiguity identification and measurement in natural language requirements specifications”, *Requirements Engineering*, 13(3): 207-239, 2008.
- Knight00 Knight, C., and Munro, M.C.; “Virtual but visible software”, *International Conference on Information Visualisation*, pp. 198–205, 2000.
- Ko08 Ko, A.J. and Myers, B.A.; “Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior”, *Best Paper on International Conference on Software Engineering 2008*, 2008.
- Kunz70 Kunz, W. and Rittel, H.W.J., “Issues as elements of information systems”, *Working Paper 131, Center for Urban and Regional Development, University of California, Berkeley*, 1970.
- Lakoff80 Lakoff, G., and Johnson, M.; *Metaphors We Live By*, The University of Chicago Press, 1980.
- Lamsweerde91 Lamsweerde, A. van, Dardenne, A., Delcourt, B., and Dubisy, F.; “The KAOS Project: Knowledge Acquisition in Automated Specification of Software”, *AAAI Spring Symposium Series, Stanford University, American Association for Artificial Intelligence*, pp. 59-62, 1991.

- Lamsweerde00 Lamsweerde, A. van; "Requirements engineering in the year 00: a research perspective", *International Conference on Software Engineering 2000*, pp. 5-19, 2000.
- Lee03 Lee, J.; Kuo, J.; Hsueh, N., and Fanjiang, Y.; "Trade-off requirement engineering", Lee, J. (ed.) *Software Engineering with Computational Intelligence*, Springer, pp. 51-72, 2003.
- Leveson95 Leveson, N.; "Medical Devices: The Therac-25", *Safeware: System Safety and Computers*, Addison-Wesley, 1995.
- Liu96 Liu, X. F., Yen, J.; "An Analytic Framework for Specifying and Analyzing Imprecise Requirements", *International Conference on Software Engineering 1996*, pp. 60-69, 1996.
- Maiden96 Maiden, N., Rugg, G.; "ACRE: selecting methods for requirements acquisition", *Software Engineering Journal*, 11(3): 183-192, 1996.
- Maiden04 Maiden, N., Gizikis, A., and Robertson, S.; "Provoking Creativity: Imagine What Your Requirements Could Be Like", *IEEE Software*, 21(5): 68-75, 2004.
- Maiden05 Maiden, N., and Robertson, S.; "Integrating Creativity into Requirements Processes: Experiences with an Air Traffic Management System", *13th IEEE International Requirements Engineering Conference*, 2005.
- Maiden07 Maiden, N., Ncube, C., and Robertson, S.; "Can Requirements Be Creative? Experiences with an Enhanced Air Space Management System", *29th International Conference on Software Engineering*, 2007.

- Maiden08 Maiden, N.; *Creativity and Invention in Requirements Engineering*, Lecture slides, City University of London, February, 2008.
- Mangano08 Mangano, N., Baker, A., Dempsey, M., Navarro, E.O. and van der Hoek, A.; "Calico: A Tool for Early Software Design Sketching", *VL/HCC Workshop: Sketch tools for diagramming*, 2008.
- Marcelloni99 Marcelloni, F., and Akşit, M.; "Reducing Quantization Error and Contextual Bias problems in Software Development Processes by Applying Fuzzy Logic", *18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS'99)*, pp. 268-272, 1999.
- Marcelloni00 Marcelloni, F. and Akşit, M.; "Improving Object-Oriented Methods by using Fuzzy Logic", *ACM Applied Computing Review*, 8 (2): 14-23, 2000.
- Marcelloni01 Marcelloni, F., and Akşit, M.; "Leaving Inconsistency using Fuzzy Logic", *Information and Software Technology*, 43: 725-741, Elsevier, 2001.
- Marcelloni04 Marcelloni, F., and Akşit, M.; "Fuzzy logic-based object-oriented methods to reduce quantization error and contextual bias problems in software development", *Fuzzy Sets and Systems*, 145(1): 57-80, 2004.
- Marcelloni04a Marcelloni, F., and Akşit, M.; "Automating Software Development Using Fuzzy Logic", *Soft Computing in Software*

*Engineering Series: Studies in Fuzziness and Soft Computing*, pp. 97-124, Springer, 2004.

- Marhlo09 Marhold, C., Rohleder, C., Salinesi, C., and Doerr, J.; "Clarifying Non-functional Requirements to Improve User Acceptance – Experience at Siemens", *15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'09)*, pp. 139-146, 2009.
- Marshall96 Marshall, M.N.; "Sampling for qualitative research", *Family Practice*, 13: 522-525, 1996.
- MATLAB MATLAB®, <http://www.mathworks.com/products/matlab/>
- Merriam-Webster12 *Merriam-Webster Dictionary*, on-line version, Merriam-Webster, <http://www.merriam-webster.com/dictionary/>
- Meyer85 Meyer, B.; "On formalism in specifications", *IEEE Software*, 2(1): 6-26, 1985.
- Mich04 Mich, L., Franch, M., and Inverardi, P.N.; "Market research for requirements analysis using linguistic tools", *Requirements Engineering*, 9(1): 40-56, 2004.
- Mich04a Mich, L., Franch, M., and Inverardi, P.N.; "Erratum: Market research for requirements analysis using linguistic tools", *Requirements Engineering*, 9(1): 151, 2004.
- Mich04b Mich, L., Anesi, C., and Berry, D.M.; "Requirements Engineering and Creativity: An Innovative Approach Based on a Model of Pragmatics Communication", *10th Anniversary*

*International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'04)*, 2004.

- Millard98 Millard, N., Lynch, P., and Tracey, K.; "Child's Play: Using Techniques Developed to Elicit Requirements from Children with Adults", *Third International Conference on Requirements Engineering*, pp. 66-73, 1998.
- Mitamura99 Mitamura, T.; "Controlled Language for Multilingual Machine Translation" (invited paper), *Machine Translation Summit*, 1999
- Mitamura01 Mitamura, T., and Nyberg, E.; "Automatic Rewriting for Controlled Language Translation", *NLPRS 2001 Workshop on Automatic Paraphrasing: Theory and Application*, 2001.
- Mylopoulos90 Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M.; "Telos: Representing Knowledge about Information Systems", *ACM Transactions on Information Systems*, 8(4): 325-362, 1990.
- Mylopoulos92 Mylopoulos, J., Chung, L., and Nixon, B. A.; "Representing and Using Nonfunctional Requirements: A Process-Oriented Approach", *IEEE Transactions on Software Engineering*, 18(6): 483-497, 1992.
- Nassi73 Nassi, I. and Shneiderman, B.; "Flowchart techniques for structured Programming", *SIGPLAN Notices*, 8(8):12-26, 1973.
- Noppen07 Noppen, J.; *Imperfect Information in Software Design Processes*, Ph.D.Thesis, Enschede, The Netherlands, 2007.

- Noppen07a Noppen, J, van den Broek, P., and Aksit, M.; "Imperfect Requirments in Software Development", *Proceedings of the 13th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'07)*, pp. 247-261, 2007.
- Noppen08 Noppen, J, van den Broek, P., and Aksit, M.; "Software development with imperfect information", *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(1):3-28, 2008.
- Nuseibeh92 Nuseibeh, B., and Finkelstein, A.; "ViewPoints: A Vehicle for Method and Tool Integration", In Forte, G., Madhavji, N. H., and Muller, H. A. (eds.), *5th International Workshop on Computer-Aided Software Engineering (CASE '92)*, pp. 50-60, IEEE Computer Society Press, 1992.
- Nuseibeh94 Nuseibeh, B.; *A Multiple Perspective-Framework for Method Integration*, PhD Thesis, Department of Computing, Imperial College , London, UK, October,1994.
- Nuseibeh94a Nuseibeh, B., Kramer, J., and Finkelstein, A.; "A framework for expressing the relationships between multiple views in requirements specifications", *IEEE Transactions on Software Engineering*, 20(10): 760-773, 1994.
- Nuseibeh00 Nuseibeh, B., and Easterbrook, S.; "Requirements Engineering: A Roadmap", *The Future of Software Engineering, Companion volume of the 22nd International Conference on Software Engineering*, 2000.



- Nuseibeh01 Nuseibeh, B., Easterbrook, S., and Russo, A.; "Making Inconsistency Respectable in Software Development", *The Journal of Systems and Software*, 58: 171-180, 2001.
- OMG\_SysML Object Management Group; "OMG Systems Modeling Language (OMG SysML), <http://www.omg.org/spec/SysML/1.1/>.
- OMG\_UML Object Management Group; "Unified Modeling Language (UML), <http://www.uml.org/>.
- Panas03 Panas, T., Berrigan, R., and Grundy, J.; "A 3d metaphor for software production visualization", *International Conference on Information Visualization*, p. 314, 2003.
- Preece94 Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T.; *Human-Computer Interaction*, Addison-Wesley, 1994.
- Pool04 Pool, J.; *Ambiguity Control in Human-Machine Multiagent Systems: State of the Art*, <http://panlex.org/pubs/etc/ambigmas.html>, Utilika Foundation, 2004.
- Robinson93 Robinson, W.N.; "Automated negotiated design integration: Formal representation and algorithms for collaborative design", Technical Report, University of Oregon, Eugene, 1993.
- Robinson94 Robinson, W.N.; "Interactive Decision Support for Requirements Negotiation", *Concurrent Engineering: Research & Applications*, Special Issue on Conflict Management in Concurrent Engineering, The Institute of Concurrent Engineering, (2): 237-252, 1994.

- Robinson96            Robinson, W.N., and Volkov, S.; “Conflict oriented requirements restructuring”, Technical Report, Georgia State University, Atlanta, 1993.
- Robinson97            Robinson, W.N., and Volkov, S.; “A Meta-Model for Restructuring Stakeholder Requirements”, *19th International Conference on Software Engineering*, pp. 140-49, IEEE Computer Society Press, 1997.
- Robinson03            Robinson, W.N., Pawlowski, S.D., and Volkov, V.; “Requirements interaction management”, *ACM Computing Surveys (CSUR)*, 35 (2), 132-190.
- Robertson02            Robertson, J.; “Eureka! Why Analysts Should Invent Requirements”, *IEEE Software*, 19(4): 20-22, 2002.
- Russel96                Russell, S.; “Machine Learning”; In Boden, M. A. (eds.), *Artificial Intelligence*, Academic Press, 1996.
- Ryan93                 Ryan, K.; “The role of natural language in requirements engineering”, *IEEE International Symposium on Requirements Engineering (ISRE 1993)*, pp. 240–242. IEEE Computer Society Press, Los Alamitos, 1993.
- Sampaio07             Sampaio, A., Rashid, A., Chitchyan, R., and Rayson, P.; “EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering”, *Transactions on Aspect-Oriented Software Engineering III, LNCS 4620*, pp. 4-39, 2007.
- Sardinha09            Sardinha, A., Chitchyan, R., Weston, N., Greenwood, P., and Rashid, A.; “EA-Analyzer: Automating Conflict Detection in

- Aspect-Oriented Requirements”, *24<sup>th</sup> IEEE/ACM International Conference on Automated Software Engineering*, 2009.
- Sardinha10      Sardinha, A., Araújo, J., Moreira, A., and Rashid, A.; “Conflict Management in Aspect-Oriented Requirements Engineering”, *Information Sciences and Technologies Bulletin of the ACM Slovakia, Special Section on Early Aspects*, R. Chitchyan, S. Zschaler (eds.), 2(1): 56-59, 2010.
- Schneider92      Schneider, G.M., Martin, J., and Tsai, W.T.; “An Experimental Study of Fault Detection in User Requirements Documents”, *ACM Transactions on Software Engineering and Methodology*, 1(2): 188-204, 1992.
- Shaw95      Shaw, M., DeLine, R., Klein, D. V., Ross, T. L., Young, D. M., and Zelesnik, G.; “Abstractions for Software Architecture and Tools to Support Them”, *IEEE Transactions on Software Engineering*, 21(4): 314-335, 1995.
- Shaw96      Shaw, M.; “Truth vs Knowledge: The Difference Between What a Component Does and What We Know It Does”, *8th International Workshop on Software Specification and Design*, 1996.
- Sim08      Sim, S. E., Alspaugh, T. A., and Al-Ani, B.; “Marginal Notes on Amethodical Requirements Engineering: What experts learned from experience”, *16<sup>th</sup> IEEE International Requirements Engineering Conference*, 2008.
- Skeels08      Skeels, M., Lee, B., Smith, G., and Robertson, G.; “Revealing Uncertainty for Information Visualization”, *AVI 2008 the*

- International Working Conference on Advanced Visual Interfaces, 2008.
- Soares06            Soares, S., Borba, P., and Laureano, E.; "Distribution and Persistence as Aspects", *Software: Practice and Experience*, 36 (7):711–759, 2006.
- Sommerville97        Sommerville, I., and Sawyer, P.; *Requirements Engineering: A Good Practice Guide*, Wiley, 1997.
- Sommerville97a      Sommerville, I., and Sawyer, P.; "Viewpoints: principles, problems and a practical approach to requirements engineering", *Annals of Software Engineering*, 3, pp. 101-130, 1997.
- Stephenson99        Stephenson, A. G., LaPiana, L. S., Mulville, D. R., Rutledge, Peter J., Bauer, F. H., Folta, D., Dukeman, G. A., Sackheim, R., et al., *Mars Climate Orbiter Mishap Investigation Board Phase I Report*, NASA, 1999.
- Stolte02             Stolte, C., Tang, D. and Hanrahan, P.; "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases", *IEEE Transactions on Visualization and Computer Graphics*, 8(1): 52-65, January-March 2002.
- Storey08             Storey, M.-A., Ryall, J., Bull I., Myers, D. and Singer, J.; "TODO or To Bug: Exploring How Task Annotations Play a Role in the Work Practices of Software Developers," *International Conference on Software Engineering*, 2008.
- Tekinerdoğan03      Tekinerdoğan, B., and Akşit, M.; "Fuzzy Evaluation of Domain Knowledge", *Software Engineering with Computational*

*Intelligence: Studies in Fuzziness and Soft Computing*, 121, pp. 183-202, Springer, 2003.

- Tjong07 Tjong, S.F., Hartley, M., and Berry, D.M.; "Extended Disambiguation Rules for Requirements Specifications", *WER07 - Workshop em Engenharia de Requisitos*, pp. 97-106, 2007.
- Tjong08 Tjong, S.F.; *Avoiding Ambiguity in Requirements Specifications*, PhD thesis, Faculty of Engineering & Computer Science, University of Nottingham, Malaysia Campus, Semenyih, Selangor Darul Ehsan, Malaysia, 2008.
- Tversky02 Tversky, B., Morrison, J.B. and Betrancourt, M.; "Animation: Can it facilitate?", *International Journal of Human Computer Systems*, 57: 247-262, 2002.
- Weston09 Weston, N., Chitchyan, R., and Rashid, A.; "Formal semantic conflict detection in aspect-oriented requirements", *Requirements Engineering*, 14: 247-268, 2009.
- Wettel07a Wettel, R., and Lanza, M.; "Program Comprehension Through Software Habitability", *15th International Conference on Program Comprehension*, pp. 231 - 240, IEEE Computer Society, 2007.
- Wettel07b Wettel, R., and Lanza, M.; "Visualizing Software Systems as Cities", *4th IEEE International Workshop on Visualizing Software For Understanding and Analysis*, pp. 92 - 99, IEEE Computer Society, 2007.

- Yang11                      Yang, H., de Roeck, A., Gervasi, V., Willis, A., and Nuseibeh, B.; "Analysing anaphoric ambiguity in natural language requirements", *Requirements Engineering*, 16(3): 163-189, 2011.
- Yen93                        Yen, J., and Lee, J., "Fuzzy Logic as a Basis for Specifying Imprecise Requirements", *2nd IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'93)*, IEEE Computer Society, pp. 745-749, 1993.
- Yen93a                      Yen, J., and Lee, J., "A task-based methodology for specifying expert systems", *IEEE Expert*, 8(1): 8-15, Feb. 1993.
- Yin02                        Yin, R. K.; *Case Study Research: Design and Methods*, Sage Publications, 2002.
- Yu97                         Yu, E.; "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", *3rd IEEE International Symposium on Requirements Engineering (RE 1997)*, pp. 226-235, 1997.
- Zadeh86                     Zadeh, L. A.; "Test-score semantics as a basis for a computational approach to the representation of meaning", *Literacy Linguistic Computing*, 1: 24-35, 1986.
- Zadeh99                     Zadeh, L. A.; "From Computing with Numbers to Computing with Words – From Manipulation of Measurements to Manipulation of Perceptions", *IEEE Transactions on Circuits and Systems-I: Fundamental, Theory and Applications*, 45(1): 105-119, 1999.

- Zeng08                      Zeng, Y.; "Recursive Object Model (ROM) - Modeling of Linguistic Information in Engineering Design", *Computers in Industry*, 59(6), pp. 612–625, 2008.
- Zimmermann91            Zimmermann, H.-J.; *Fuzzy Set Theory and Its Applications*, Kluwer Academic, Boston, AIA, 1991.