

LINGUISTIC APPROACHES FOR EARLY MEASUREMENT
OF FUNCTIONAL SIZE
FROM SOFTWARE REQUIREMENTS

H M ISHRAR HUSSAIN

A DOCTORAL THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE &
SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

AUGUST 2014

© H M ISHRAR HUSSAIN, 2014

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: H M Ishrar Hussain

Entitled: Linguistic Approaches for Early Measurement of Functional Size from
Software Requirements

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards
with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair
Dr Rachida Dssouli

_____ Examiner
Dr Jean-Marc Desharnais

_____ Examiner
Dr Yong Zeng

_____ Examiner
Dr Constantinos Constantinides

_____ Examiner
Dr René Witte

_____ Supervisor
Dr Olga Ormandjieva

_____ Supervisor
Dr Leila Kosseim

Approved by _____
Chair of Department or Graduate Program Director

_____ 2014 _____
Dean of Faculty

Abstract

Linguistic Approaches for Early Measurement of Functional Size from Software Requirements

H M Ishrar Hussain, Ph.D.
Concordia University, 2014

The importance of early effort estimation, resource allocation and overall quality control in a software project has led the industry to formulate several functional size measurement (FSM) methods that are based on the knowledge gathered from software requirements documents. The main objective of this research is to develop a comprehensive methodology to facilitate and automate early measurement of a software's functional size from its requirements document written in unrestricted natural language. For the purpose of this research, we have chosen to use the FSM method developed by the Common Software Measurement International Consortium (COSMIC) and adopted as an international standard by the International Standardization Organization (ISO). This thesis presents a methodology to measure the COSMIC size objectively from various textual forms of functional requirements and also builds conceptual measurement models to establish traceability links between the output measurements and the input requirements. Our research investigates the feasibility of automating every major phase of this methodology with natural language processing and machine learning approaches. The thesis provides a step-by-step validation and demonstration of the implementation of this innovative methodology. It describes the details of empirical experiments conducted to validate the methodology with practical samples of textual requirements collected from both the industry and academia. Analysis of the results show that each phase of our methodology can successfully be automated and, in most cases, leads to an accurate measurement of functional size.

Keywords

Functional Size Measurement, Software Requirements Specification, Effort Estimation, Natural Language Processing, Text Mining.

Acknowledgements

I would like to express my most sincere gratitude to all the people who made this thesis possible. First and foremost, much of my appreciation goes to my dearest supervisors: Dr. Olga Ormandjieva and Dr. Leila Kosseim, for their continuous and invaluable guidance. They have been my mentors at every step of this extensive research, helped me tremendously in the write-up of this thesis, and no matter what the problems I faced, they always came across as my strongest support.

It was also a great honor for me to have many constructive feedback from Dr. Jean-Marc Desharnais, one of the co-founders of the original COSMIC measurement standard.

I would also like to sincerely thank Dr. René Witte for his very knowledgeable insights and time-to-time suggestions that were always extremely helpful. My deep appreciation also goes to my two other thesis committee members: Dr. Constantinides Constantinos and Dr. Yong Zeng for their insightful reviews and questions that guided this research.

My special thanks go to the current and previous members of the NLP-SE research group for providing their valuable efforts in the annotation and the measurement experiments; especially to Rolan Abdukalykov, for collecting the industrial project documents used in these experiments.

Finally, on a personal note, I would like to convey my deepest appreciation to my loving wife, my father and my parents-in-law, for their continuous inspirations and encouragements to complete this research.

I dedicate this thesis to my mother, whose heavenly blessings surround me at my every step.

Contents

List of Figures	xiii
List of Tables.....	xvii
1. Introduction.....	1
1.1 Introduction.....	1
1.2 Motivation.....	4
1.3 Objectives & Research Methodology	8
1.4 Overview of Contributions	13
1.4.1 Theoretical Contributions.....	13
1.4.2 Developed Tools.....	15
1.5 Overview of the Thesis	16
2. Background.....	17
2.1 Introduction.....	17
2.2 Software Requirements.....	17
2.3 Requirements Analysis using Textual Annotation	18
2.4 Problem of Estimating Effort from Requirements.....	20
2.5 Effort Estimation in Theory	21
2.6 Effort Estimation in Practice.....	22
2.7 Functional Size Measurement.....	23
2.7.1 ISO Standards for FSM	23

2.7.2	COSMIC.....	23
2.8	Size Measurement in Agile Development Processes.....	27
3.	Literature Survey.....	29
3.1	Introduction.....	29
3.2	Automated COSMIC Functional Size Measurement.....	29
3.2.1	µcROSE.....	30
3.2.2	Work of Candori-Fernández <i>et al.</i>	31
3.3	Approximation of Functional Size.....	31
3.4	Preparing Textual Requirements for FSM.....	32
3.4.1	On Detection of Requirements Defects.....	32
3.4.2	On Extracting Functional & Non-Functional Requirements.....	40
3.4.3	On Identifying Domain Concepts & Their Attributes.....	42
3.5	Tools for Requirements Annotation.....	43
3.5.1	GATE TeamWare.....	44
3.5.2	Amazon Mechanical Turk.....	44
3.5.3	Knowtator.....	45
3.5.4	Glozz.....	45
3.5.5	AnCoraPipe.....	45
3.6	Necessary Features of Annotation Tools.....	46
3.6.1	Document Acquisition.....	46
3.6.2	Document Pre-processing.....	47
3.6.3	Administration of Annotators' Recruitment.....	47
3.6.4	Customization of Annotation Interface.....	47

3.6.5	Relational Annotations	48
3.6.6	Hierarchical Dependency Among Annotation Labels.....	48
3.6.7	Aggregating Annotation Data of Multiple Annotators.....	48
3.6.8	Computation of Gold-standard Annotations.....	49
3.7	Effort Estimation Techniques	49
3.7.1	Using Parametric Models for Effort Estimation.....	50
3.7.2	Estimation by Analogy (EBA)	54
3.7.3	Delta Estimation	55
3.7.4	Calibration and Use of Machine Learning Algorithms	55
3.8	Conclusion	57
4.	Methodology	58
4.1	Introduction.....	58
4.2	Phases of Our Methodology	58
4.2.1	Inception	60
4.2.2	Phase I: FSM by Non-Experts	60
4.2.3	Phase II: Size Approximation.....	63
4.2.4	Phase III: Requirements Classification.....	65
4.2.5	Phase IV: FSM Model Extraction	66
4.2.6	Phase V: Evaluation of FSM Automation	71
4.3	Formalization of FSM Model and Quantification	73
4.4	Linguistic Introspection of Size Measurement Analysis (LISMA)	78
4.4.1	Analysis of Requirements Quality/Defects	80
4.4.2	Classification by Requirements Taxonomy.....	81

4.4.3	Measurement of Functional Size	82
4.5	Conclusion	87
5.	Functional Size Measurement By Non-Experts	88
5.1	Introduction.....	88
5.2	LASR: Live Annotation of Software Requirements	90
5.2.1	Annotation Interface for Functional Size Measurement.....	90
5.2.2	Additional Features of LASR.....	94
5.3	Experiments Overview	98
5.3.1	Participants: The Annotators	99
5.3.2	Experimental Materials.....	100
5.3.3	Experiment Execution	101
5.4	Experiment #1: Expert vs. Multiple Non-Experts	104
5.4.1	Hypotheses and Variables	104
5.4.2	Results and Analysis.....	106
5.5	Experiment #2a: Annotation Accuracy using LASR.....	109
5.5.1	Hypotheses and Variables	109
5.5.2	Results and Analysis.....	111
5.6	Experiment #2b: Size Measurement Accuracy with LASR	115
5.7	Experiment #3: Record of Annotation Time	118
5.8	Design & Implementation of LASR	119
5.9	Domain Model of LASR.....	121
5.9.1	Project.....	122
5.9.2	Annotation Work	122

5.9.3	Problem Domain.....	123
5.10	User Roles of LASR.....	124
5.11	Conclusion.....	125
6.	Automated Approximation of Functional Size.....	126
6.1	Introduction.....	126
6.2	Overview of Our Approach.....	126
6.2.1	CFP Measurement.....	127
6.2.2	Class Annotation of Functional Processes.....	129
6.2.3	Text Mining.....	130
6.3	Experiments.....	130
6.3.1	Corpus Annotation.....	132
6.3.2	Syntactic Feature Selection.....	134
6.3.3	Lexical Feature Selection.....	135
6.3.4	Feature Extraction and Classification.....	136
6.4	Results and Analysis.....	136
6.5	Conclusion.....	140
7.	Automated FSM Modeling.....	142
7.1	Introduction.....	142
7.2	Our Approach.....	143
7.3	Preprocessing.....	144
7.4	Extraction of Linguistic Features.....	145
7.4.1	Lexical Feature Set.....	146
7.4.2	Syntactic Feature Set.....	147

7.4.3	Combined Feature Set	148
7.5	Heuristic-based Classification	151
7.5.1	Data-Attribute Classification	152
7.5.2	Data-Movement Classification	153
7.6	Supervised Learning-based Classification	156
7.6.1	Feature Selection	156
7.6.2	Choice of Learning Algorithms	157
7.7	Data-Group Association	159
7.8	Extending COSMIC by CFP Range Measurement	161
7.9	Overview of the Classification Experiments	164
7.9.1	The Corpora	164
7.9.2	Testing Methods	166
7.9.3	Types of Classification Experiments	170
7.10	Heuristic-based Classification Results	172
7.11	Supervised Learning-based Classification Results	173
7.11.1	Results of Batch Testing for Incremental Learning	173
7.11.2	Results of 10-fold Cross-Validation	179
7.12	CFP Range Measurement Results	181
7.13	Conclusion	182
8.	Conclusions and Future Work	183
8.1	Major Contributions	183
8.1.1	Methodology for Practical Application of FSM	183
8.1.2	Formalization of an FSM Model for Traceability	184

8.1.3	Extension to FSM Quantification	184
8.1.4	Syntactic Features for Requirements Classification	185
8.1.5	Improving the Annotation Quality for Non-Experts	185
8.1.6	Automatic and Traceable Approximation of FSM	185
8.1.7	Linguistic Features of COSMIC FSM.....	186
8.1.8	Heuristics for COSMIC FSM	186
8.1.9	Annotated Corpora for Data-Movement Classification.....	187
8.2	Future Research Directions.....	187

Bibliography..... 189

Appendix A: Lexical Databases 211

A.1	Data-movement Verbs	211
A.1.1	Entry Verbs.....	211
A.1.2	Exit Verbs.....	212
A.1.3	Read Verbs	212
A.1.4	Write Verbs	212
A.1.5	Triggering-Entry Verbs	213
A.1.6	System-Message-Exit Verbs	213
A.2	Stative Verbs.....	213
A.3	Attribute Names	214
A.4	Data-group Names	214
A.5	Actor Names	215

Appendix B: Raw Test Results: Outputs from WEKA	216
B.1 10-fold Cross-Validation Results.....	216
B.1.1 For <i>Entry</i> Classifier	216
B.1.2 For <i>Exit</i> Classifier.....	217
B.1.3 For <i>Read</i> Classifier	218
B.1.4 For <i>Write</i> Classifier	220
B.2 Batch Test Results for Incremental Learning	221
B.2.1 For <i>Entry</i> Classifier	222
B.2.2 For <i>Exit</i> Classifier.....	223
B.2.3 For <i>Read</i> Classifier.....	224
B.2.4 For <i>Write</i> Classifier	225
 Appendix C: Significance of Linguistic Features	 226
 Appendix D: An Example of Automated FSM	 245

List of Figures

1. Cone of Uncertainty (McConnell, 2006)	2
2. Workflow of Early Estimation of Effort from Software Requirements	3
3. The Conceptual Framework of the Research Methodology.....	11
4. Simple Input-Output and the Intermediate Steps of LISMA Supporting Early Effort Estimation	12
5. Example of Sentence-level Annotation of Software Requirements.....	19
6. Generic Flow of Data-Groups in COSMIC	24
7. Example of Noun-Phrase-Level Annotation of Software Requirements (for COSMIC FSM).....	26
8. Steps of Iteration Planning in Agile [as presented in (Cohn, 2005)]	27
9. Requirements Specification Ambiguity Checker (ReqSAC).....	40
10. Cost Drivers in COCOMO II	51
11. Phases of Our Methodology.....	59
12. Ontology of COMIC FSM Model.....	75
13. Inputs, Outputs and the Intermediate Steps of LISMA Supporting Early Effort Estimation	79
14. Our Approach for Measuring the Functional Size from Functional Requirements	83
15. Our Approach for Building The Lexical Databases.....	85
16. An Overview Workflow Diagram of LASR	89
17. First Screenshot of LASR's Customized Annotation Interface for COSMIC Annotation.....	91

18. Second Screenshot of LASR's Customized Annotation Interface for COSMIC Annotation.....	94
19. Distribution of the True Gold-Standards (As Annotated by the Expert) in Our Corpus	104
20. Pair-wise Agreements between the Four Annotators of NE_{ft} and the Expert.....	107
21. Distribution of the Gold-standard Annotations (As Annotated by NE_{ft}).....	108
22. Distribution of the Gold-Standard Annotations (As Annotated by NE_{mt}).....	113
23. Quality of the Different Gold-Standard Annotations in Terms of Their Agreements (in Kappa) with the True Gold-Standard Annotations.....	114
24. Architecture of LASR	120
25. Domain Model of LASR.....	121
26. Inputs and Outputs of Our Approach for Automated Approximation of Functional Size.....	127
27. Building a Historical Database.....	128
28. Class Annotation by Box-Plot Analysis.....	129
29. Text Mining for Fast Approximation of COSMIC Functional Size	130
30. Distribution (with a box plot) of CFP Values in Our Historical Database	132
31. The Resultant C4.5 Decision Tree after Training with the Complete Dataset.....	137
32. Inputs and Outputs of Our Approach for Automated FSM Modelling	143
33. Major Steps of Our Approach for Automated FSM Modeling	144
34. Workflow of the Preprocessing Step.....	145
35. Workflow of the Feature Extraction Step	146
36. Example Decision Tree to Classify Entry Data-Movement (when the whole dataset is used to train the C4.5 algorithm)	158
37. Distribution of the Gold-Standard Class Labels for Data-Movement Classification	165

38. Number of Training Instances Used in Each Batch Test for the <i>Entry</i> Classifier.....	170
39. Types of Experiments to Validate the Different Data-Movement Classification Approaches	171
40. Results of Running Batch Tests for Incremental Learning Over the Entry Classifier	174
41. Results of Running Batch Tests for Incremental Learning Over the Exit Classifier	175
42. Results of Running Batch Tests for Incremental Learning Over the Read Classifier	176
43. Results of Running Batch Tests for Incremental Learning Over the Write Classifier	177
44. Example of the Feature: “Noun Phrase is a Direct Object”.....	227
45. Example of a Prepositional Object.....	228
46. Example of a Chain of Prepositional Objects.....	228
47. Example of the Feature: “Noun Phrase is in an Object-Like Position”.....	230
48. Example of the Feature: “Noun Phrase is a Subject”.....	231
49. Example of the Feature: “Noun Phrase is in an Object-Like Position of a Verb with an Actor Subject”.....	233
50. Example of the Feature: “Noun Phrase is in an Object-Like Position of a Verb with Non-Actor Subject”.....	235
51. Example of the feature: "Noun Phrase is in an Object-Like Position of a Data-Movement Verb".....	237
52. Example of the Feature: "Noun Phrase is a Mention of an Attribute".....	238
53. Example of the Feature: "Noun Phrase is a Mention of a Data-Group".....	239
54. Example of the Feature: "Noun Phrase is Related to an Attribute Name by a Chain of Prepositional Objects".....	240

55. Screenshot (1) of GATE, Loaded with the Textual Requirements of a Functional Process	245
56. Screenshot (2) of GATE, Creating a Corpus	246
57. Screenshot (3) of GATE, Loading a Pipeline for Preprocessing and Feature Extraction.....	247
58. Screenshot (4) of GATE, Showing the Output of Running the Preprocessing & Feature Extraction Pipeline.....	247
59. Screenshot (5) of GATE, Showing Feature Values Extracted from a Noun-Phrase.	248
60. Screenshot (6) of GATE, Loading a Pipeline for COSMIC FSM Model Extraction & Quantification	249
61. Screenshot (7) of GATE, Showing the Output Annotations of Noun-Phrases & Sentences Classified into Data-Movement Types	249
62. Screenshot (8) of GATE, Showing the Identified Data-Group That is Associated with an Output Annotation Noun-Phrase.....	250
63. Screenshot (9) of GATE, Showing the Output CFP and the List of Extracted Artifacts of COSMIC FSM Standard.....	250

List of Tables

1. Mapping of the Open Research Problems, Research Phases and Research Objectives	9
2. Meyer’s “The seven sins of the specifier” (Meyer, 1985)	33
3. Ambiguity in NL Requirements (Kamsties, Berry, & Paech, 2001).....	34
4. Different Quality Indicators of NL Requirements (Gnesi, Lami, & Trentanni, 2005)	37
5. Comparison of Features Provided by Current Annotation Tools.....	46
6. Mapping of Research Questions over the Reseach Phases in Relation to the Objectives	73
7. Documents in the Corpus, Used in the Experiments.....	100
8. Task of the Annotators	101
9. Confusion Matrix for Moving Data-Attribute Annotation by NE_{ft}	109
10. Confusion Matrix for Data-Attribute Annotation by NE_{mt}	113
11. Confusion Matrices for Data-Movement Annotation by NE_{mt}	114
12. Frequency of Data-Attributes & Data-groups.....	116
13. Aggregated Frequencies of Data-Movements.....	117
14. Total Measured CFP and MMRE Results.....	117
15. Activities Permitted to Different User Roles in LASR.....	124
16. A Hypothetical Example of CFP Calculation	128
17. Ranges of CFP Values to Define the Classes.....	129
18. Summary of the Case Studies	131

19. Data to be Associated with a Functional Process to Approximate Its Size.....	133
20. Ten Linguistic Features Most Highly Correlated with CFP	134
21. Some of the Keywords of POS Category: Noun, Verb and Adjective	136
22. Summary of the Results	137
23. Confusion Matrix When Using 10-fold Cross-Validation	138
24. Precision, Recall and F-Measure, When Using 10-fold Cross-Validation	139
25. 10-fold Cross-Validation Results of Using a 2-Class and 3-Class Classifier	140
26. Frequency of Data-movement Class Labels in the Corpora	165
27. Training and Testing Instances in Batch Test #1 for the <i>Entry</i> Classifier	167
28. Training and Testing Instances in Batch Test #2 for the <i>Entry</i> Classifier	167
29. Training and Testing Instances in Batch Test #3 for the <i>Entry</i> Classifier	168
30. Training and Testing Instances in Batch Test #4 for the <i>Entry</i> Classifier	169
31. Training and Testing Instances in Batch Test #5 for the <i>Entry</i> Classifier	169
32. Confusion Matrices for Heuristic-based Data-Movement Classification	172
33. Results of Heuristic-based Data-Movement Classification	172
34. Confusion Matrices for 10-fold Cross Validation on Supervised Learning-based Data-Movement Classification	180
35. Results of 10-fold Cross Validation on Supervised Learning-based Data-Movement Classification.....	180
36. CFP Range Measurements Results Based on the Heuristic-based Classification Labels	181

Chapter 1

Introduction

“Adding manpower to a late software project makes it later.”
— *Frederick P. Brooks, Jr.*

1.1 Introduction

Functional size is a fundamental characteristic of a software that indicates how big it is in terms of the amount of business level functionalities it provides. The size information can be used to perform various quality analyses about a software. However, functional size is primarily used to determine the development effort of the software, as the required effort to develop a software cannot be estimated without knowledge of its size. This thesis attempts to solve the problem of determining the functional size objectively from textual requirements.

Today’s software industry produces highly complex systems on competitive budget and schedule. Every commercial software project, therefore, begins with the step of determining the size of the software and its related effort, before starting the development process. However, a meta-study performed by (Molkken & Jørgensen, 2003) on 10 different surveys, shows that most (60-80%) of the industrial software projects face inevitable overruns in both the budget and/or the schedule because of inaccuracy in software effort estimation. The recent CHAOS manifesto, published by The Standish Group, shows an overall increase in the failure rate of software projects in the year 2012, with overruns in the budget and/or schedule still being a primary problem (The Standish Group, 2013). This shows that the task of effort estimation is very difficult to do accurately at the early stages of development. The reason is that the total effort depends on parameters, e.g. the software size, that cannot be objectively measured until later in the development process. This leads to the common phenomenon known as the “Cone of uncertainty”, shown in Figure 1, as presented in (McConnell, 2006),

that shows that the estimated effort is more likely to vary by a large extent at an early stage of development.

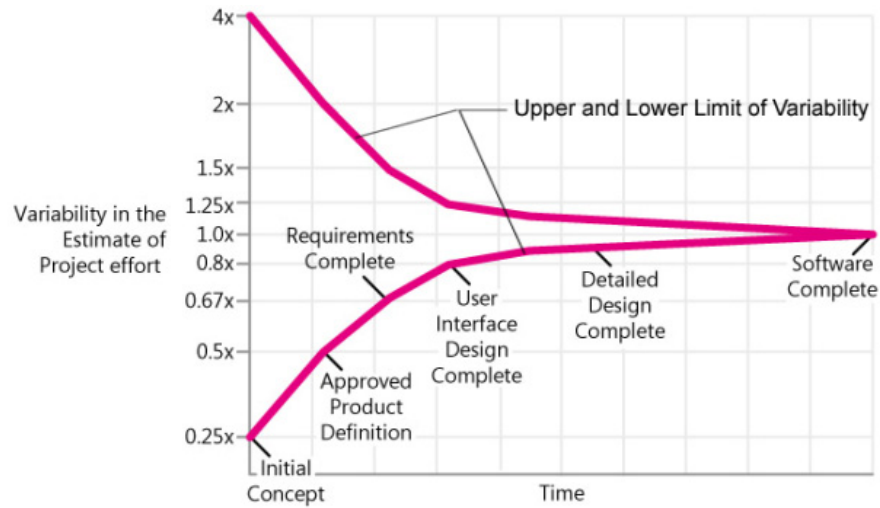


Figure 1: Cone of Uncertainty (McConnell, 2006)

To minimise costly errors in effort estimation, the industry needs for the size of software to be estimated effectively at an early development phase. Measuring the physical size of software, in terms of its Source Lines Of Code (SLOC), for example, can only be objectively possible after starting its development work. Functional size, on the other hand, indicates the logical size information about a software that can be measured before starting the development work by counting the units of functionalities that the software should provide (Albrecht & Gaffney, 1983). This functional size can then be used to estimate different aspects of the software project, including the effort required to develop the software. The software requirements document, one of the earliest deliverables produced in the software development life-cycle, usually holds enough details about the software for the experts to use in predicting the functional size of the software and its related development effort. A typical workflow of measuring functional size and estimating effort from textual software requirements is shown in Figure 2.

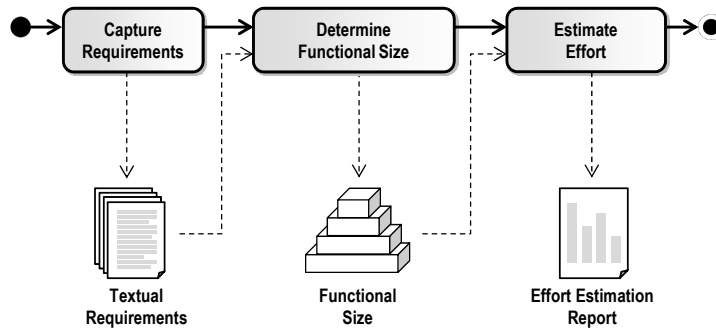


Figure 2: Workflow of Early Estimation of Effort from Software Requirements

Functional Size Measurement (FSM) standards, e.g. IFPUG (ISO/IEC 20926, 2003), Mark II (ISO/IEC 20968, 2002), NESMA (ISO/IEC 24570, 2005), FiSMA (ISO/IEC 29881, 2010) and COSMIC (ISO/IEC 19761, 2011), provide formal ISO-certified ways of measuring the functional size of the software. However, the available FSM methods do not provide specific guidelines for extracting the size information from the initial and informally written textual software requirements. The information about a software's functional size is encoded in its requirements document and an expert needs to read and decode it using his/her experience gained over time through trial-and-error-like manual processes. Hence, none of the FSM methods can be carried out early without waiting for the requirements to be highly formalized or without the costly, yet subjective, insights of expert measurers.

The FSM methods measure the functional size of a software by first identifying some of its conceptual artifacts that are referred to as the Base Functional Components (BFC) (ISO/IEC 14143-1, 1998). Each of these artifacts represents an elementary unit of a functional requirement, the types of which are to be aggregated and counted in a way defined by the FSM method in order to measure the size of the software. COSMIC¹ is an FSM method that claims to be objective in counting the types of BFCs (ISO/IEC 19761, 2011). COSMIC's method of measurement is also demonstrated to be compatible with the requirements that are strictly formalized and well-decomposed. For example, the work of Condori-Fernández, Abrahão, & Pastor (Condori-Fernández, Abrahão, & Pastor, 2007) showed that all modeling artifacts of COSMIC can be mapped to UML-based conceptual modeling artifacts that are extracted manually by human experts from software requirements documents.

¹ COSMIC is named after its developer organization COMmon Software Measurement International Consortium.

However, during the early phases of software development, software requirements are informally produced and documented in textual form in plain natural language without any formalization. Natural language textual requirements also inherently contain different requirements defects (Meyer, 1985; IEEE, 1998). Thus, the conceptual modeling artifacts that are required to be identified during functional size measurement are not clearly visible to the measurers when using textual requirements without formalization. This makes the task of functional size measurement, even following the COSMIC standard, rather subjective and highly dependent on the measurers' judgments and levels of experience. The COSMIC manual (ISO/IEC 19761, 2011) identifies these problems as defects of un-formalized software requirements that are left for the measurer to solve with his/her subjective assessment. Thus, like all other FSM standards, COSMIC also avoids providing specific guidelines for dealing with textual requirements in measuring the functional size of the software.

1.2 Motivation

Crucial management decisions made during software development depend on the initial estimations. With an early knowledge of well-estimated functional size and the related development effort, a project manager can confidently plan future courses of action. Thus, there has always been a demand from the industrial point of view for better estimations of size and effort, so that project managers can determine the competitive cost of a project, improve process monitoring, and negotiate contracts from a position of knowledge.

The motivation of our research originates from the industrial demand of the functional size and its related development effort to be estimated effectively at an early development phase, so that costly errors in planning and possibility of mismanagement of projects can be minimized. Functional Size Measurement (FSM) methods (Albrecht A. J., 1979; ISO/IEC 20968, 2002; ISO/IEC 19761, 2003; ISO/IEC 29881, 2010) were presented with the aim to reduce subjectivity in the estimation of software size and its related effort by measuring only the logical size of software represented by the amount of functionalities described in its requirements. However, FSM methods also rely on the subjective judgments of human experts, especially when extracting the size information from textual requirements produced during the early stages of software development life-cycle.

Current state-of-the-art FSM approaches do not investigate the details of performing functional size measurement using unrestricted textual requirements, which, however, is the form of requirements produced during the earliest phases of the software development life-cycle. Thus, current studies on FSM do not explore the linguistic aspects of textual requirements related to functional size measurement. This leads to the following open research problem:

Open Problem #1: *“To our knowledge, no research to date has attempted to discover how the linguistic elements of the textual requirements of a software influence its functional size.”*

Addressing this research problem can aid in emulating an expert's way of interpreting the functional size of a software from the linguistic elements of informally-written textual requirements. This linguistic knowledge can therefore be used to approximate functional size at a very early stage when textual requirements are not fully developed.

Furthermore, classifying software requirements onto the various dimensions of requirements taxonomy is one of the primary tasks performed in the requirements analysis processes (SWEBOK, 2004). Functional Size Measurement (FSM) involves a requirements analysis process where the functional size is measured by analyzing only those requirements that are functional. Therefore, before carrying the process of FSM, the functional requirements need to be identified first from within the requirements documents in case they appear interleaved together with non-functional requirements. Thus, the task of classifying textual requirements into functional and non-functional requirements is a pre-requisite for performing functional size measurement. This task is commonly performed by a human requirements analyst in practice, while some recent studies (Rashwan, Ormandjieva, & Witte, 2013; Hussain, Kosseim, & Ormandjieva, 2008; Cleland-Huang, Settini, Zou, & Solc, 2006) have addressed the use of discriminating keywords along with statistical learning-based text mining approaches for requirements classification. However, no other research, to our knowledge, has so far evaluated the accuracy of using richer linguistic knowledge, such as the syntactic information of the textual requirements, for these text mining approaches. Using discriminating syntactic information for aggregating the keywords, not only helps to reduce the feature space of the text mining approaches, but can also help to generalize the

classification model to classify textual requirements with unseen words with better accuracy. Thus, this leads to a second open research problem, presented as follows:

Open Problem #2: *“To our knowledge, no research to date has attempted to explore the use of syntactic features of textual requirements for distinguishing functional and non-functional requirements.”*

Addressing this research problem can help improve the current state-of-the-art by improving the accuracy of automatic requirements classification. This can in turn provide feasible means of automating the task of extracting functional requirements prior to conducting functional size measurement that is otherwise performed manually by requirements analysts.

Human measurers build their experiences in Functional Size Measurement (FSM) tasks by learning to extract FSM modeling artifacts through interpreting the functional requirements that are well-decomposed. However, their practice of FSM involves a trial-and-error-based process of interpreting these textual requirements. To our knowledge, the literature does not investigate how a human measurer perceives different parts of the textual requirements to extract the objects of interest in an FSM model. These FSM approaches, therefore, cannot eliminate the need of applying subjective judgments of human measurers in identifying these FSM modeling artifacts from textual requirements. This leads to our last open research problem:

Open Problem #3: *To our knowledge, no research has attempted to discover the relationship between the objects of interest in a Functional Size Measurement (FSM) model and the linguistic elements of textual requirements.*

Addressing this problem can explain the tasks of FSM objectively by building traceability links between linguistic elements of well-decomposed textual requirements and the outcomes of FSM and its related objects of interests. Identifying these relations can aid us to develop linguistic guidelines of conducting FSM from textual requirements that can be used to train non-experts or an automated system for an early automation of FSM tasks.

The three open research problems presented above entail several practical issues:

(1) Human bias, with varied levels of expertise: Subjective judgments during functional size measurement involve wide-ranging bias of human measurers that can produce inconsistent results. The correctness of measurement involving subjective judgments also depends on the levels of expertise of human measurers. Therefore, when measurers lack experience on a particular problem domain, they often produce erroneous measurements.

(2) Costly execution: Experts in FSM methods and certain problem domains attain their knowledge and experience over time of conducting FSM activities successfully over unrestricted textual requirements related to those problem domains. Having experienced human experts available for each software project belonging to many different problem domains can introduce additional costs for the respective projects. Manipulating large amounts of textual requirements can also be costly, affecting the overall productivity of a software project. Thus, functional size measurement processes are in practice, often avoided for Agile-based projects, where strong constraints are imposed on the time of completing an iteration.

(3) Lack of traceability: The subjective process of measurements entails that there exists no formal method of recording the justification for the outcome of a measurement task. Thus, subjective judgments during functional size measurement does not record the reasoning of how the size was measured and where this knowledge of size originated from, i.e. which parts of the textual software requirements provided the size information, which conceptual modeling artifacts were discovered by the measurers to deduce the size, and which parts of the textual requirements indicated the presence of these conceptual modeling artifacts. Without such traceability information, the final outcome of the functional size measurement cannot be justified, and any error in the outcome, therefore, cannot be traced back to its original source for an effective fix.

In the next section we present our research objectives along with an overview of our research methodology.

1.3 Objectives & Research Methodology

Based on the open problems described in Section 1.2, we set the aim of our research presented in this thesis as follows:

“To develop an objective procedure that does not depend on human expertise to effectively measure functional size from textual requirement.”

Our aim leads us to perform an applied research that explores a feasible solution to the following key question:

“What methodology that does not depend on human expertise can be applied objectively to measure functional size from textual requirements with minimal errors (i.e. within an acceptable rate of errors)?”

Our work attempts to solve the above key question by decomposing it into several supporting research objectives, each having a set of strictly defined research questions to be addressed:

Objective #1: *“To investigate if the process of functional size measurement (FSM) can be executed effectively by non-experts.”*

Objective #2: *“To improve the overall process of FSM-related requirements annotation by attaining accurate annotations with non-experts having minimal training.”*

Objective #3: *“To determine the most discriminating linguistic features of informally written textual requirements for approximating functional size.”* (This addresses the open problem #1, as presented in Section 1.2)

Objective #4: *“To explore the most discriminating syntactic features of textual requirements for classifying them into functional and non-functional requirements.”* (This directly addresses the open problem #2, as presented in Section 1.2)

Objective #5: *“To identify how experts deduce the relationship between the linguistic elements of unrestricted textual requirements and the objects of interest in a functional size measurement model.”* (This addresses the open problem #3, as presented in Section 1.2)

Objective #6: “To evaluate the feasibility of automating functional size measurement from textual requirements.”

Objectives #3, #4 and #5 directly address the open research problems described in Section 1.2; while our objectives #1, #2 and #6 extend the scope of our research to make its outcomes applicable in practice. In addition, each of the above objectives is associated with a set of strictly defined research questions, for which we developed measurable hypotheses. A detailed discussion on each of our objectives, along with the related research questions and hypotheses are presented in Chapter 4.

To accomplish the above objectives, we conducted several empirical studies in different phases of our research:

Phase I: FSM by Non-Experts

Phase II: Size Approximation

Phase III: Requirements Classification

Phase IV: FSM Model Extraction

Phase V: Evaluation of FSM Automation

The details of each of these phases are described in Chapter 3 along with the methodology used in this thesis.

	Open Research Problem #1	Open Research Problem #2	Open Research Problem #3	Extending Scope for Practical Application
Phase I: FSM by Non-Experts				Objective #1 & Objective #2
Phase II: Size Approximation	Objective #3			
Phase III: Requirements Classification		Objective #4		
Phase IV: FSM Model Extraction			Objective #5	
Phase V: Evaluation of FSM Automation				Objective #6

Table 1: Mapping of the Open Research Problems, Research Phases and Research Objectives

	Open Research Problem #1	Open Research Problem #2	Open Research Problem #3	Extending Scope for Practical Application
Phase I: FSM by Non-Experts				Objective #1 & Objective #2
Phase II: Size Approximation	Objective #3			
Phase III: Requirements Classification		Objective #4		
Phase IV: FSM Model Extraction			Objective #5	
Phase V: Evaluation of FSM Automation				Objective #6

Table 1 shows how each of these phases maps to address the open research problems, presented in Section 1.2 along with the six objectives of our research.

Each of the open problems of Section 1.2 can be viewed as a problem of text mining that targets specific requirements analysis activities related to functional size measurement. We therefore designed the conceptual framework of our research methodology as illustrated in Figure 3.

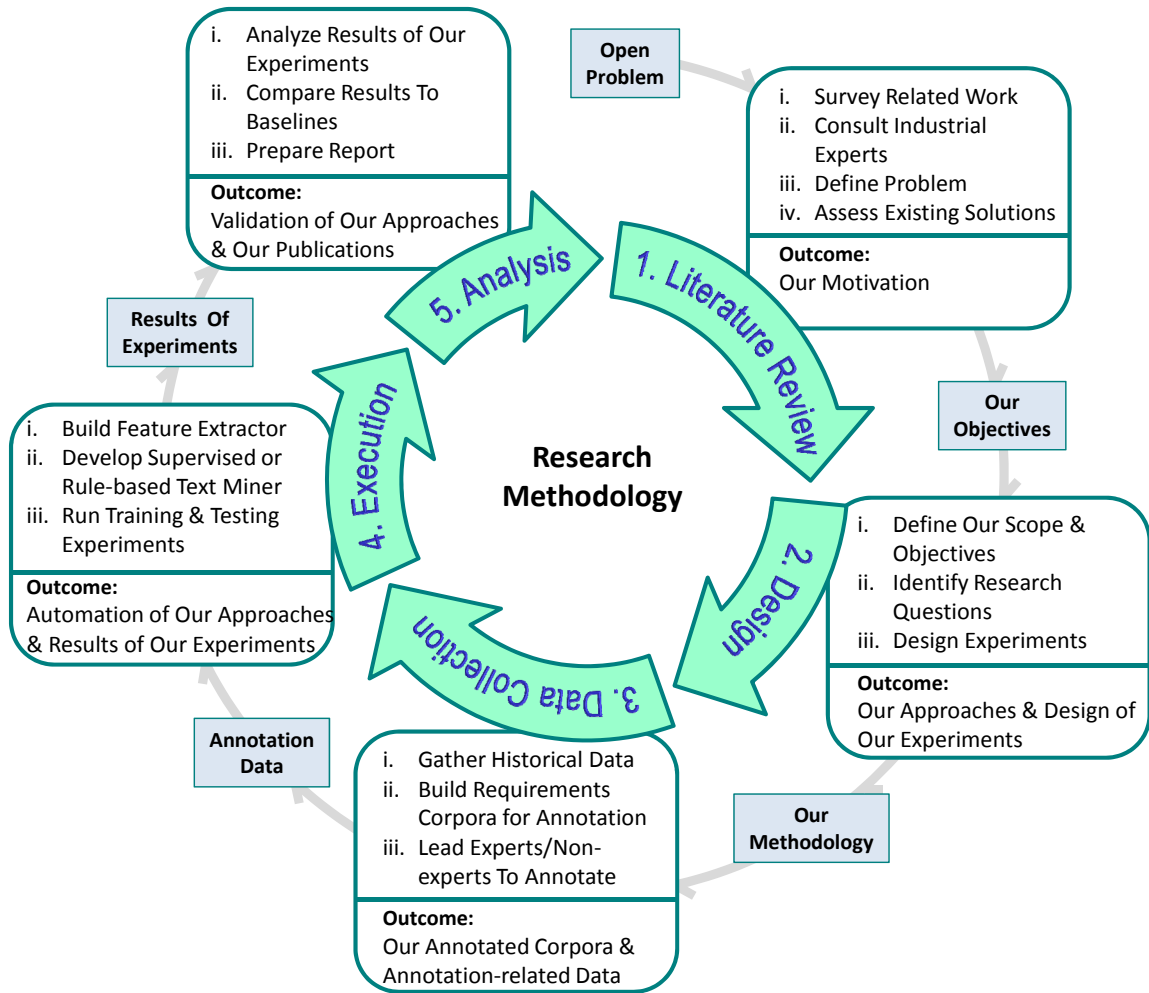


Figure 3: The Conceptual Framework of the Research Methodology

As shown in Figure 3, the phases of our research start with investigating the open research problems through a literature review. This allows us to set our motivation for the research, along with the related research questions, and design our methodology and experiments to validate the methodology. We then continue with the tasks of data-collection and building the necessary tools for executing the experiments. Finally, we analyze the results of our experiments and publish our findings.

Our overall aim with this research is to address the open problems described in Section 1.2 by developing a comprehensive methodology for measuring functional size from textual software requirements written in unrestricted natural language, and, thus, facilitate early estimation of software development effort. We name our methodology: Linguistic Introspection of Size Measurement Activities (LISMA), and present its details in this thesis.

Figure 4 briefly shows the input-output of our methodology, along with the intermediate tasks performed.

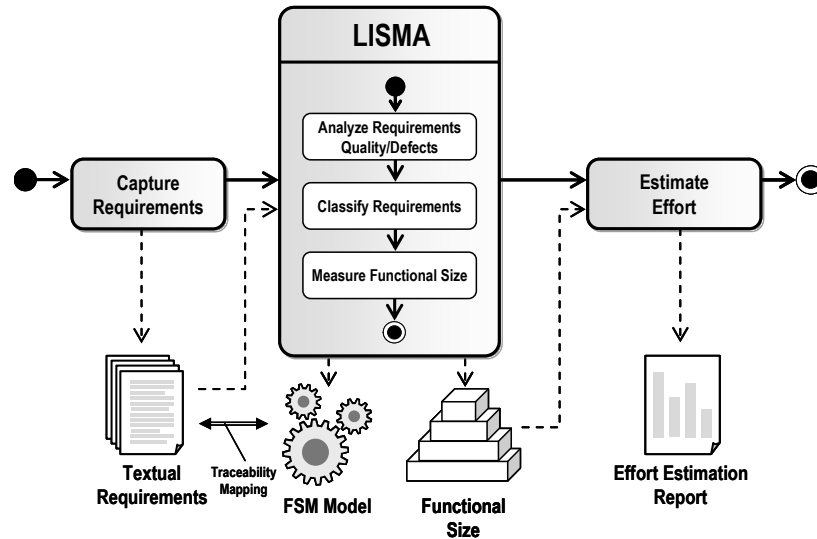


Figure 4: Simple Input-Output and the Intermediate Steps of LISMA Supporting Early Effort Estimation

As shown in Figure 4, LISMA can be incorporated within the existing workflow of measuring functional size that is presented earlier in Figure 2. LISMA starts by taking textual requirements as input and produces the functional size for effort estimation. It implements three intermediate steps that aim to achieve three practical goals:

- (1) To analyze the quality and/or defects of software requirements, using the approach presented in (Hussain, 2007).
- (2) To classify textual requirements into any prescribed standard of requirements taxonomy based on its linguistic features, and distinguish functional from non-functional requirements.
- (3) To measure the functional size using the textual form of the functional requirements, and output the size information along with a functional size measurement (FSM) model that includes a traceability mapping between the linguistic elements of textual requirements and the objects of interest in the FSM model.

Thus, we intend to build natural language processing applications, powered by different supervised machine learning techniques, to demonstrate the possible automation of our

methodology, LISMA, and emulate the COSMIC standard (ISO/IEC 19761, 2003) as our preferred FSM method for this research.

1.4 Overview of Contributions

The following sections list the contributions that this research makes to the fields of software measurement, software requirements engineering and natural language processing in general.

1.4.1 Theoretical Contributions

The theoretical contributions of this research, in relation to our research objectives, as listed in Section 1.3, are listed below:

- (1) Our work, through our research objective #1, presents and verifies the feasibility of an innovative approach of functional size measurement through manual annotation of textual requirements. The details are presented in Section 4.2.2, and then in Chapter 5.

Publication: The details of this work and the results of the experiments verifying the feasibility of our model is published in (Hussain, Ormandjieva, & Kosseim, 2012).

- (2) Our work, through our research objective #2, presents our dynamic annotation adjustment model that can help non-experts to perform requirements annotation tasks related to functional size measurement and achieve an acceptable level of agreement with the annotations of an expert. The details of this work are presented in Section 4.2.2, and in Chapter 5.

Publication: The description of this work, along with an empirical evaluation can be found in (Hussain, Ormandjieva, & Kosseim, 2012).

- (3) Our work, through our research objective #3, determines the most discriminating linguistic features of informally written textual requirements for approximating functional size. The details of this work are presented in Section 4.2.3 and Chapter 6.

Publication: Details of this work and the experimental results have been published in (Hussain, Ormandjieva, & Kosseim, 2010; Hussain, Kosseim, & Ormandjieva, 2013).

- (4) Our work, through our research objective #4, determines the most discriminating syntactic features of textual requirements for distinguishing the functional and non-functional requirements automatically. An overview of this work is presented in Section 4.4.2.

Publication: The complete details of this work was published in (Hussain, Kosseim, & Ormandjieva, 2008).

- (5) Through our research objective #5, our work determines a pool of linguistic features that can help identify which base noun-phrases² are related to specific artifacts of a functional size measurement model. The details of this work and the results of the experiments are presented in Section 4.2.5, and then, in Chapters 7.

- (6) Our work through our research objective #5, also presents heuristics to build traceable relationships between linguistic elements of textual requirements and the artifacts of functional size measurement model. The accuracy of using these heuristics was compared to our supervised learning-based solution to evaluate its feasibility. The details of this work and the results of our experiments are presented in Section 4.2.6, and then in Chapter 7.

- (7) Through our research objective #6, we developed a complete methodology for measuring functional size automatically from textual requirements. The details of this methodology are presented in Section 4.4.

² A base noun-phrase is a noun-phrase that does not contain any other noun-phrase within its scope, but itself (Sang, et al., 2000). For our work, however, we use the term “base noun phrase” to refer to a noun or a noun compound or a personal pronoun. Thus, in our case, it actually refers the smallest part of the base noun phrase that does not contain any part-of-speech class of words, other than the nouns or a personal pronoun. It therefore represents the smallest segment of textual requirement that can independently express the mention of an artifact of a functional size measurement model.

Publication: The details of this methodology are also published in (Hussain, Ormandjieva, & Kosseim, 2009).

- (8) In relation to our research objective #6, we evaluated the accuracy of automatically measuring functional size from textual requirements using our approach. Details of our experiments and their results are presented in Chapter 7.

To demonstrate the applicability of our research, we also presented an innovative approach of effort estimation using functional size measurement that takes into account the impacts of different types of Non-Functional Requirements and different types of problem domains. The details of our approach and the results of the experiments verifying the accuracy of our approach is published in (Abdukalykov, Hussain, Kassab, & Ormandjieva, 2011).

1.4.2 Developed Tools

We present below the list of tools which we have developed as part of our practical contributions of this research:

- (1) We developed a prototype for an online annotation tool, called Live Annotation of Software Requirements (LASR), that allows requirements engineers to remotely contribute software requirements documents and to collaboratively annotate textual requirements in a secured environment.
- (2) We also developed a text mining application in Java that can automatically classify textual requirements sentences as functional and non-functional requirements.
- (3) Moreover, we implemented our approach of approximating functional size by developing a text miner in Java that can automatically approximate the functional size of a system based on its informally-written unrestricted textual requirements.
- (4) Furthermore, we developed supervised learning-based text mining applications that altogether can automatically extract the modeling artifacts of a functional size measurement model from textual requirements.

- (5) Finally, we also implemented algorithms for measuring functional size by developing a heuristic-based text miner in GATE (Cunningham H., *et al.*, 2011) that can automatically extract the modeling artifacts of a functional size measurement model from textual requirements and calculate the functional size of a system.

1.5 Overview of the Thesis

This chapter introduced the open problems that we addressed with our research. It presented how the early application of any functional size measurement standard prescribes costly manipulation of human experts and how their process of measurement lacks objectivity and traceability. It then presented our motivations for this research, along with our research objectives and major contributions in the related fields.

The next chapter, that is Chapter 2, introduces the necessary background knowledge on the topics related to this research. Then, Chapter 3 presents a detailed survey of the current literatures, while Chapter 4 describes our research methodology in details. Chapter 4 also discusses our formalization of a conventional function size measurement process and a detailed overview on how the process can be automated. Then, Chapters 5, 6 and 7 present the implementation details of three different approaches for functional size measurement that are proposed in this thesis: (i) measuring functional size manually by non-experts, (ii) approximating functional size automatically based on linguistic traits, and (iii) measuring functional size by automatically extracting the conceptual artifacts of the measurement model, respectively. Each of these chapters describes the proposed approach and the experimental work to validate each of these solutions, along with an analysis of the experimental results. We then summarize the findings of our research and propose the future avenues of research in Chapter 8. The attached appendices include supporting details in relation to the topics discussed in the body of this thesis.

Chapter 2

Background

“Be the measure great or small...
let it be honest in every part.”
— *John Bright*

2.1 Introduction

Before discussing the details of the various approaches proposed in the current literature for addressing the research problems, described in Section 1.2, we present in this chapter the details of the background topics related to this research.

2.2 Software Requirements

Software requirements (SR) document is the medium used to communicate user's requirements to technical people responsible for developing the software. It is one of the earliest artifacts produced in a software development life cycle that not only provides the development team the knowledge about the required behavior and quality characteristics of the system to be developed, but also gives the estimators the notion of its size and the development effort that it is going to require. (Leffingwell & Widrig, 2003) defined a software requirement in their book as follows:

- *A software capability needed by the user to solve a problem to achieve an objective*
- *A software capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation.*

This indicates that the task of writing requirements as a two-way process, where both the users (and/or clients) and the technical people (analysts, developers, managers and others) are involved. Thus, the common practice is to write the requirements document without any formalization, i.e. in plain natural language (NL), so that it can be easily conveyable between the two parties.

Software requirements provide a high-level baseline from which project progress can be compared and scope can be controlled. Understanding all features specified in a requirements document is a key factor for improving software project planning and gaining agreement with the client on scope, cost, and schedule. A requirements document thus acts as a primary source of information that can aid the estimation of the software functional size and its development effort at an early phase in software development life cycle.

2.3 Requirements Analysis using Textual Annotation

The annotation of requirements documents is a common practice of extracting information from informally written requirements (Ko, Park, Seo, & Choi, 2007). Requirements analysts annotate different parts of the software requirements document to indicate what classes of software requirements they contain (such as functional or non-functional), or, which software engineering artifacts are present (e.g. domain entities, data attributes, etc.), or, any other class of information vital to identifying the base functional components that pertain to a chosen FSM method.

In the context of software project management, FSM needs to be performed at an early phase of the software development lifecycle (Meli, 1997), when the textual requirements are immature and are essentially captured in unrestricted natural language without any formalization, so that they can easily be conveyable between the clients (and/or the potential users) and the technical people (analysts, developers, managers and others) (Leffingwell & Widrig, 2003). Being written in unrestricted natural language, these textual requirements are often corrupted with ambiguity that an expert has to first manually identify and resolve (Meyer, 1985). The documents containing these textual requirements can also be either

unstructured or of varied levels of structure which demands additional effort from an expert to manually extract crucial knowledge about the software to be developed. For example, sentences describing non-functional requirements are often found embedded in paragraphs containing functional requirements that an expert often has to manually organize by identifying the non-functional requirements from the functional ones (Hussain, Kosseim, & Ormandjieva, 2008). For example, Figure 5 shows an extract from a requirements document, and how an expert has chosen to classify its requirement sentences with different annotation labels.

Extract from a requirements document	
<p><i>...The following use case describes approving a budget. First, the user navigates to the budget overview page. The system then displays the budget overview with editable budget attributes. System presets some of the budget attributes. User edits the budget attributes and sets the status as "Approved". All the mandatory attributes cannot be empty and the budget amounts cannot be negative. User finally saves the budget. ...</i></p>	
Requirement Sentence	Annotation Label
<i>The following use case describes approving a budget.</i>	Noise
<i>First, the user navigates to the budget overview page.</i>	Functional
<i>The system then displays the budget overview with editable budget attributes.</i>	Functional
<i>System presets some of the budget attributes.</i>	Functional
<i>User edits the budget attributes and sets the status as "Approved".</i>	Functional
<i>All the mandatory attributes cannot be empty and the budget amounts cannot be negative.</i>	Non-Functional
<i>User finally saves the budget.</i>	Functional

Figure 5: Example of Sentence-level Annotation of Software Requirements

The requirements sentences in this example (shown in Figure 5) are to be annotated into four classes:

- i. Functional Requirement:** A software requirement that expresses the required behavior of the system.
- ii. Non-Functional Requirement:** A software requirement that expresses the quality requirements and the constraints over the related behavior of the system.
- iii. Ambiguous Requirement:** A software requirement that can be interpreted in more than one way by the annotator (i.e. the requirements analyst).

- iv. Noise:** Any sentence that does not express any of the above types of software requirement.

In practice, software requirements can be further classified according to different standards of requirements taxonomy. A software development organization usually adapts to one such standard and chooses the classes of requirements that are to be annotated during the requirements analysis phase. Reuse of requirements documents also require annotating its parts following a standard of requirements taxonomy (Eriksson, Börstlerb, & Borga, 2009).

Software measurement experts can also annotate well-decomposed textual requirements to extract crucial information about the functional size of the software to be developed. This allows early measurement of functional size from software requirements that can be used to estimate the development of effort.

2.4 Problem of Estimating Effort from Requirements

Since software requirements documents are most commonly written in natural language, they are susceptible to many defects. Bertrand Mayer lists them as the “Seven Sins of Specifier” (Meyer, 1985) (details on these defects will be discussed in Section 3.4.1). These defects degrade the quality of an SR document by introducing misunderstanding of requirements, which can lead to many severe problems, including erroneous estimation of development effort, if not detected earlier in the software development lifecycle.

Also, the IEEE Standard 830-1998 (IEEE, 1998) describes the practices recommended by IEEE to write an SR document, and defines the quality characteristics of a “good” SR document: (1) Correct, (2) Unambiguous, (3) Complete, (4) Consistent, (5) Ranked for importance, (6) Verifiable, (7) Modifiable and (8) Traceable. An SR document needs to be carefully written with the goal of maintaining these good quality characteristics, so that the developers can clearly visualize the problem that the requirements state before starting the process of developing the actual system. As the SR document also acts as a contract between the potential users and the developers, it imposes additional emphasis on its text to be clear and accurate when describing user needs.

One of the common reasons of failing to provide accurate effort estimation is regarded to be ill-developed requirements documents (Grimstad & Jorgensen, 2007). An SR document often obstructs the estimator with its many defects to foresee the system clearly and to perform the tasks of accurate early effort estimation. Since our research intends to deal with requirements documents written in any level of quality, one of our sub-goals in this work is to explicitly define and improve the quality of SR documents as we proceed on to do the estimate of the development effort.

2.5 Effort Estimation in Theory

There exists a large body of research in the field of effort estimation. Magne Jørgensen, one of the leading researchers in the field, is currently maintaining a website called, *BESTweb* (Jørgensen & Shepperd, 2007), listing a large number of research papers, journal articles, editorials, and books — all related to software development effort estimation. Although the website does not contain an exhaustive list of publications in this area, it already holds 1,242 publications at time of writing this thesis.

Numerous parametric models of estimating software effort have been proposed, like COCOMO (Boehm B., 1984), COCOMO II (Boehm, *et al.*, 2000), SLIM (Putnam, 1981), ESTIMACS (Rubin, 1983), all of which depend on some FSM methods for early estimation of effort from the software functional size. It should be noted that none of these effort estimation models are inherently compatible with COSMIC, which objectively measures the functional size of software. This makes the inputs to these models biased by the subjective judgment of the functional size measurers to begin with. Again, models like COCOMO and SLIM depend on the number of SLOC (Source Lines of Code) to be estimated before starting the process of estimating effort, where it can degrade the quality of their input data even more to begin with, as the estimation process of SLOC includes parameters that depend on subjective judgment of the human estimators. This problem was evident in the study of (Kemerer, 1987) where SLOC-based estimation models performed poorly compared to non-SLOC-based estimation models. Others (Shepperd & Schofield, 1997; Idri, Abran, & Khoshgoftaar, 2002; Angelis & Stamelos, 2000) introduced the approach of effort estimation by analogy (EBA) that simply extends the idea of Case-Based Reasoning (Aamodt & Plaza,

1994), relying extensively on historical data. They claimed that EBA performed better than parametric methods. However, some studies also show contradictory or inconclusive results that were obtained when different models were applied (Myrtveit, Stensrud, & Shepperd, 2005; Menzies, Zhihao, Hihn, & Lum, 2006).

The work of (Lewis, 2001) showed that the problem lies in having parameters or features on the estimation that are subjectively computed, which induces serious flaws in calibrating a model. The author suggests that many of the previous studies presented overly exaggerated results because of this bias involved in their work. Again, most of the results of all these studies in software effort estimation are presented with the statistically unreliable measure of MMRE and PRED (Conte, Dunsmore, & Shen, 1986), which many studies thoroughly criticize, like (Kitchenham, Pickard, MacDonell, & Shepperd, 2001; Foss, Stensrud, Kitchenham, & Myrtveit, 2003; Port & Korte, 2008).

After decades of repetitive research, Mike Ross, CEO of r2Estimating (*r2estimating.com*), stated in a panel discussion that there have not been a single study or research that can prove “superiority” over the other research; while Barry Boehm, founder of COCOMO, the most widely researched open model in the last three decades, stated at the panel that industries should build project plans in a way that recognizes the fact that “people and models will never be perfect” (Fraser, Boehm, Erdogmus, Jorgensen, Rifkin, & Ross, 2009).

Although all solutions proposed in the literature emphasize, directly or indirectly, on early effort estimation by using functional size as the primary independent variable, the existing literature, to the best of our knowledge, does not report on automation of any of the existing size measurement processes (discussed in Section 2.7) that receives textual requirements as input to begin the task of estimation at the requirements specification phase.

2.6 Effort Estimation in Practice

In practice, the most popular method of software effort estimation in the industry is to use *expert judgment* (Shepperd & Cartwright, 2001). The process of expert judgment is mostly conducted by individuals, seldom by groups, which comprises of often informal ways of intuitive judgment of estimation, based on the expertise of the estimators. Studies of (Lederer

& Prasad, 1992; Jørgensen M., 2004a) report that effort estimation done by expert judgment, on average, were at least as accurate as those done on estimation models. But, the results of expert judgment cannot be consistent, as the level of expertise varies from one person to the other; and, much of this fact is reflected by the increased number of project failure rates in the industry, where the principal reason is inaccurate estimation of software development effort (Molkken & Jørgensen, 2003).

2.7 Functional Size Measurement

Software development effort is directly proportional to its functional size. For better estimation of effort and deeper understanding on the functional size variable, the industry formulated several methods for *Functional Size Measurement (FSM)*. In 1979, Allan Albrecht first proposed FSM in his work on *Function Point Analysis (FPA)* (Albrecht A. J., 1979), where he named the unit of functional size as “*Function Point (FP)*”. His idea of effort estimation was then validated by many studies, like (Albrecht & Gaffney, 1983; Kitchenham & Taylor, 1984) and, thus, measuring the functional size of the software became an integral part of effort estimation.

2.7.1 ISO Standards for FSM

Over the years, many standards have been developed by different organizations on FSM methods, following the concepts presented in Albrecht's FPA method. Four of these standards have been accepted as ISO standards: they are IFPUG (ISO/IEC 20926, 2003), Mark II (ISO/IEC 20968, 2002), NESMA (ISO/IEC 24570, 2005), FiSMA (ISO/IEC 29881, 2010) and COSMIC (ISO/IEC 19761, 2011).

2.7.2 COSMIC

For the purpose of this research, we have chosen to use the COSMIC FSM method developed by the Common Software Measurement International Consortium (COSMIC) and now adopted as an international standard (ISO/IEC 19761, 2011). We chose this method in particular, because it conforms to all ISO requirements (ISO/IEC 14143-1, 1998) for FSM, focuses on the “*user view*” of functional requirements, and is applicable throughout the agile development life cycle. Its potential of being applied accurately in the requirements

specification phase compared to the other FSM methods is demonstrated by the study of (Gencel, Demirors, & Yuceer, 2005). Also, COSMIC does not rely on subjective decisions by the functional size measurer during the measurement process (ISO/IEC 19761, 2011). Thus, its measurements, taken from well-specified requirements, tend to be same among multiple measurers. This is particularly important for validating the performance of our automatic size measurements.

COSMIC measures functional size of a software in terms of the number of *Data-movements*, which accounts for the movement of one or more data-attributes belonging to a single *Data-group*. A data-group is an aggregated set of data-attributes. A *Functional Process*, in COSMIC, is an independently executable set of data-movements that is triggered by one or more *triggering events*. A triggering event is initiated by an *actor* (a functional user or an external component) that occurs outside the boundary of the software to be measured. Thus, a functional process holds the similar scope of a use case scenario, starting with the triggering event and ending with the completion of the scenario. Figure 6 illustrates the generic flow of data-groups from a functional perspective, presented in the COSMIC standard (ISO/IEC 19761, 2011).

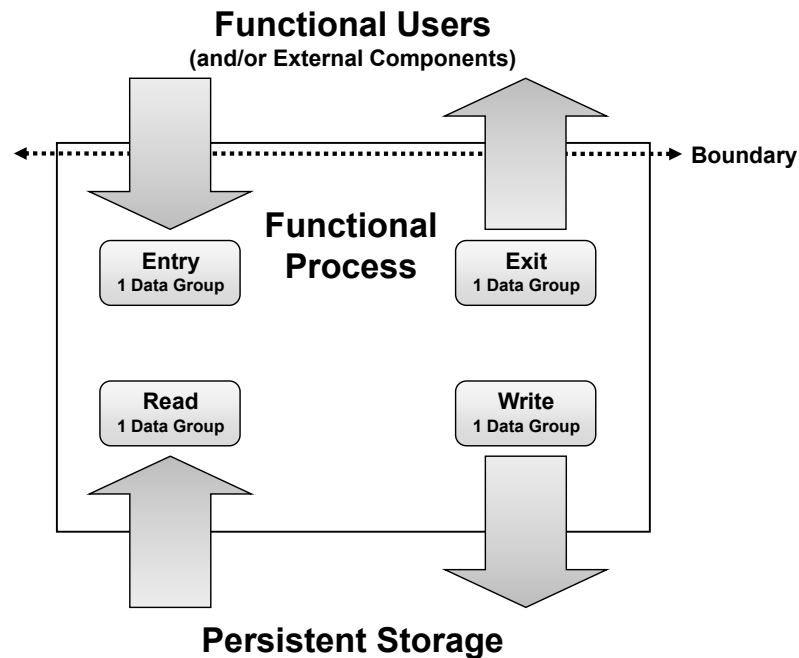


Figure 6: Generic Flow of Data-Groups in COSMIC

As shown in Figure 6, the data-movements can be of four types: Entry, Exit, Read and Write. An Entry moves a data-group from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process.

The types of these data-movements constitute the base functional component types (ISO/IEC 14143-1, 2007) of the COSMIC FSM method. Thus, COSMIC counts each of these data-movements as one CFP (COSMIC Function Point) of functional size, and measures the size of each of the functional processes separately. It then adds the sizes of all the functional processes to compute the total size of the system to be measured.

Thus, the requirements annotation tasks that can realize a functional size measurement method like COSMIC are relatively more complex than the straight forward annotation tasks presented earlier in Section 2.3. Here, we find that both the sentences (implicitly) and the base noun-phrases or base-NP³ (explicitly) of a requirements document represent information about the actors, the data-groups, the data attributes etc. of the system to be developed, and, thus, need to be annotated to measure the functional size in COSMIC.

COSMIC FSM requires the base-NP's in each functional requirement sentences to be annotated by an expert in the domain and measurement processes to specify if the base-NP's indicate the presence of different data-attributes. And, if they do, the expert must also indicate which data-group each of the data-attributes belongs to and which types of data-movements they participate in. Figure 7 shows a similar example, where a domain expert annotated an extract of the requirements document to identify the COSMIC modeling artifacts in order to measure the COSMIC size.

³ As mentioned in Chapter 1, we use the term “base noun phrase” or “base NP” to refer to a noun or a noun compound or a personal pronoun. Thus, in our case, it refers the smallest part of the base noun phrase (Sang, et al., 2000) that includes only nouns or a personal pronoun. It therefore represents the smallest segment of textual requirement that can independently express the mention of a COSMIC modeling artifact.

Extract from a requirements document					
<i>...The following use case describes creating a new budget. First, the user navigates to the budget creation page. He then enters the budget attributes. All the mandatory attributes cannot be empty and the budget amounts cannot be negative. User saves the budget. ...</i>					
Requirement Sentence	Annotation Label for Sentence	Base Noun-Phrases	Annotation Label for Data-attribute (Yes/No)	Annotation Label for Data-group	Annotation Label for Data-movement
<i>The following use case describes creating a new budget.</i>	Noise	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>
<i>First, the user navigates to the budget creation page.</i>	Functional	user	No	<i>n/a</i>	<i>n/a</i>
		budget creation page	No	<i>n/a</i>	<i>n/a</i>
		<i>(not mentioned)</i>	Yes	<i>(triggering event)</i>	Entry
<i>He then enters the budget attributes.</i>	Functional	He	No	<i>n/a</i>	<i>n/a</i>
		budget attributes	Yes	Budget Data	Entry
		<i>(not mentioned)</i>	No	<i>n/a</i>	<i>n/a</i>
<i>All the mandatory attributes cannot be empty and the budget amounts cannot be negative.</i>	Non-Functional	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>	<i>n/a</i>
<i>User saves the budget.</i>	Functional	User	No	<i>n/a</i>	<i>n/a</i>
		budget	Yes	Budget data	Write
		<i>(not mentioned)</i>	No	<i>n/a</i>	<i>n/a</i>

Figure 7: Example of Noun-Phrase-Level Annotation of Software Requirements (for COSMIC FSM).

Figure 7 shows that the expert first extracted the base noun-phrases from the sentences that he/she annotated as functional requirements. Then, the expert annotated each base noun-phrase, indicating if it was a data-attribute. In some cases, the data-attributes may not be mentioned explicitly as the base noun-phrases, and the expert took that into consideration as well. For each data-attribute that the expert identifies, he/she then annotates it with the data-group name that it belongs to and the type of data-movement that it participates in.

Here, we find that the annotation labels are hierarchically dependant on each other (i.e. the annotation labels of data-attributes, data-groups and data-movements are assigned to the base noun-phrases, if and only if the sentence containing those base noun-phrases is annotated with the label: “Functional”). Also, data-group annotation requires the availability of a list of data-group names as annotation labels that should dynamically build up over time with the accumulation of domain knowledge. Thus, such annotation tasks are more complex than straight-forward natural language annotation tasks. To our knowledge, the existing annotation

tools do not provide the necessary support to aid these complex annotation tasks that are required for a functional size measurement process, like COSMIC. In Section 3.5, we discussed some of the current annotation tools. It should be noted here that the above features are all supported by our annotation tool, LASR. The tool also includes some additional features to support the functional size measurement tasks that are not supported by any of the tools discussed above. Section 5.2 describes LASR and how it supports the tasks of functional size measurement.

Thus, by extracting the conceptual modeling artifacts from textual requirements and measuring the functional size by simply counting their frequencies, the COSMIC standard offers an objective method of FSM. It is designed to be applied in the traditional processes of software development, where documentation of requirements using formalisms and templates is required. However, over the years, the IT industry has recognized the traditional processes to cause many problems including delays and is now increasingly moving towards agile development processes, such as Scrum (Martin, 2003), an agile approach that does not impose documentation templates or formalisms on requirements.

2.8 Size Measurement in Agile Development Processes

Agile development processes are driven by the motto of delivering releases as quickly as possible (Larman, 2003). Planning an iteration in an agile project involves estimating the size of the required features as the first step. Figure 8 shows the steps of iteration planning in agile.

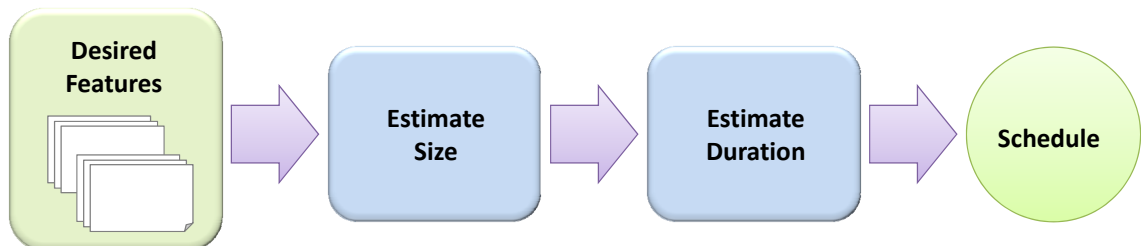


Figure 8: Steps of Iteration Planning in Agile [as presented in (Cohn, 2005)]

The size of every agile iteration is subjectively estimated by means of user requirements that are written less formally than use case descriptions. These textual requirements, which are mostly available in the form of *smart use cases* (Hoogendoorn, 2009) or *user-stories* (Martin, 2003), although, do not provide detailed description of the scenarios like those found in use

cases, they must hold “enough details” to perform the size estimation (Martin, 2003). Size measurement methods in agile development processes include story-points (Cohn, 2005) and smart estimation (Hoogendoorn, 2009), and depend on the subjective judgment of human experts, and, therefore, are prone to biases and errors (Cohn, 2005).

In an agile development process, the lack of formalism in requirements restricts FSM methods, like COSMIC, to be applied for measuring the functional size of an iteration. For example, from the discussion in Section 2.7.1, it can be understood that the number of data-groups, which is necessary to be known to carry out COSMIC FSM, cannot be identified by the measurer from a set of requirements statements alone unless he/she is supplied with a complete list of available data-groups that requires formalizing the requirements with conceptual model (e.g. a domain model).

Our work, on the other hand, presents an alternative solution to estimate the COSMIC functional size in agile that does not require the use of formalism in requirements; instead, it proposes an objective way of approximating the COSMIC functional size of a functional process (i.e. a use case) that is described by an informally written set of textual requirements, in forms likely to be used in agile size estimation.

In the next chapter, we discuss the current literature and state-of-the-art tools that are introduced in the different fields related to our research.

Chapter 3

Literature Survey

“The most reliable way to forecast the future is to try to understand the present.”

— *John Naisbitt*

3.1 Introduction

Our research encompasses many fields of studies to deal with the problem of early effort estimation. Since we intend to estimate the development effort from requirements documents (of any quality), we first need to use natural language processing (NLP) techniques to extract the functional size of the software. We will then devise a solution for estimating the effort using the functional size as the primary variable and different cost drivers as other variables in a machine learning environment to perform various regression analyses. Thus, we will present here the work of different fields that relate to the scope of our research.

3.2 Automated COSMIC Functional Size Measurement

As discussed in Section 2.7, many standards were proposed by different organizations on functional size measurements of software, after Allan Albrecht had first proposed his function point analysis (FPA) (Albrecht A. J., 1979): such as IFPUG (ISO/IEC 20926, 2003), Mark II (ISO/IEC 20968, 2002), NESMA (ISO/IEC 24570, 2005), FiSMA (ISO/IEC 29881, 2010) and COSMIC (ISO/IEC 19761, 2011; COSMIC, 2014). Although, like every FSM method, COSMIC has some disadvantages (e.g. until now, the standard does not take into account non-functional requirements to its size measurement), we have chosen to automate this standard for the following reasons:

- (1) COSMIC is currently the only ISO recognized FSM method that does not rely on subjective decisions by the functional size measurer during measurement process (ISO/IEC 19761, 2011). Thus, its measurements, taken from well-specified requirements, tend to be same among multiple measures. This is particularly important for validating the performance of the automatic size measurements that would be yielded by our solution.
- (2) Compared to other ISO recognized FSM Methods, COSMIC only demonstrates the prospect to be applied at the earliest phase of software development lifecycle, that is, in the requirements specification phase (Gencel, Demirors, & Yuceer, 2005).

There have been many different approaches proposed by in recent years (Sneed, 2001; Diab H., Koukane, Frappier, & St-Denis, 2005; Condori-Fernández, Abrahão, & Pastor, 2007) where researchers attempted to automate different functional size measurement processes. However, to our knowledge, no previous work has addressed the problem where textual requirements are taken as input to start the measurement process. Instead, they all relied on conceptual models to be manually built from the requirements, so that their automated approach can be adopted. In the following sections, we will only be focusing on notable approaches towards automating COSMIC FSM processes.

3.2.1 μ cROSE

One of the leading work done in the area of automating COSMIC FSM is by (Diab H., Koukane, Frappier, & St-Denis, 2005), where the authors developed a comprehensive system called, μ cROSE, which accepts state charts as inputs to measure the functional size of real-time systems only. Their work heavily depends on a set of hard-coded rules for mapping different objects of interest to different COSMIC components, and also require C++ code segments to be attached with the state transitions and supplied as inputs too, so that data-movements can be identified. They presented a very brief validation of their work by an expert, testing their system against only one case study, where it performed poorly in detecting the data groups, resulting in some erroneous measurement outputs.

3.2.2 Work of Candori-Fernández *et al.*

(Candori-Fernández, Abrahão, & Pastor, 2007), on the other hand, performed another study, where they presented step by step guidelines to first derive manually the UML modeling artifacts, e.g. the use case models and system sequence diagrams from the requirements, and then, apply their set of rules for measuring the COSMIC functional size of the system from the UML models. Their approach was validated on 33 different observations, showing reproducible results with 95% confidence.

3.3 Approximation of Functional Size

Most of the related work in this field attempted to perform a precise measurement of COSMIC functional size that rely on tedious manual processing to extract conceptual modeling artifacts, require formalization of the requirements, and, therefore, are not applicable to agile development processes. On the other hand, the work of (Meli, 1997) presents a fully manual approach of quick approximation of COSMIC size from textual requirements without extracting COSMIC modeling artifacts. It first classifies past projects into fuzzy size classes (e.g. Small, Medium, Large, Very Large,...), finds the common traits within the concepts used in software of belonging to the same size class, and, finally, allows a human measurer to discover similar traits in the new software component, so that the measurer can estimate its COSMIC size by drawing analogy to the past projects. We find a good potential of this work to be applied in the environment of agile processes that demand quicker estimation of software size.

The goal of our thesis is to develop a fully automated tool that would do quicker estimation of COSMIC size using informally written textual requirements of any quality as input, making it favorable for agile processes. We extend the idea of (Santillo, Conte, & Meli, 2005) by finding common traits, or 'features', among software projects of the same size classes, but looking for linguistic features within the textual requirements, and use supervised text mining methods to automate the process.

3.4 Preparing Textual Requirements for FSM

As our work of extracting functional size information starts from requirements documents written in any quality, our first concern is improving the quality of textual requirements so that the FSM process could be carried out. There have been numerous studies on automatically detecting defects in textual requirement, classifying requirements into various classes of requirements taxonomy, extracting conceptual models etc. — all of which provided vital guidance on the formulation of our research. Many of these studies are described in the following sections.

3.4.1 On Detection of Requirements Defects

Several research projects have addressed the problem of detecting deficiencies in natural language requirements specification. These studies typically use a small number of approaches, which are, although often similar in the types of tools they use, radically different in the way they try to detect ambiguities in requirements documents. In the following sections, we will review a few noteworthy studies by categorizing them according to their approaches.

A. Manual Detection Process

Manual detection is the most popular approach to detect and resolve the ambiguities of NL requirements specification. One of the early leading studies in this field was conducted by Bertrand Meyer in (Meyer, 1985), showing the areas of a natural language requirements specification, where the specifier is more prone to make mistakes (see Table 2). Meyer stressed the point that natural language requirements specification are inherently ambiguous, and for resolving these ambiguities, use of formal specifications are absolutely necessary. However, for detecting such ambiguities, he explains the process of manually going through each word, phrase and sentence of the NL requirements specification text of his case study, and checking if they reflect any of the seven sins of the specifier.

Noise	<i>The presence in the text of an element that does not carry information relevant to any feature of the problem.</i>
Silence	<i>The existence of a feature of the problem that is not covered by any element of the text.</i>
Over-specification	<i>The presence in the text of an element that corresponds to a feature of the problem but to features of a possible solution.</i>
Contradiction	<i>The presence in the text of two or more elements that define a feature of the system in an incompatible way.</i>
Ambiguity	<i>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.</i>
Forward Reference	<i>The presence in the text of an element that uses features of the problem not defined until later in the text.</i>
Wishful Thinking	<i>The presence in the text of an element that defines a feature of a problem in such a way that a candidate solution cannot realistically be validated with respect to this feature.</i>

Table 2: Meyer's "The seven sins of the specifier" (Meyer, 1985)

Another study worth mentioning here is the one done by (Kamsties, Berry, & Paech, 2001), who introduced five classes of different ambiguity problems of NL requirements specifications — each well-defined with practical examples, and used as items of a checklist for validating a requirements document. They are: Lexical Ambiguity, Systematic Ambiguity, Referential Ambiguity, Discourse Ambiguity and Domain Ambiguity. Table 3 describes these items briefly.

Item	Description
Lexical Ambiguity, Polysemy	Does a word in a requirement have several possibly related meanings? Be aware that <i>lexical ambiguity</i> arises in particular from the actual usage of a word in an RE context.
Systematic Polysemy	A systematic polysemy applies to a class of words: (1) The <i>object–class ambiguity</i> arises when a word in a requirement can refer either to a class of objects or to just a particular object of the same class. (2) The <i>process–product ambiguity</i> arises when a word can refer either to a process or to a product of the process. (3) The <i>volatile–persistent ambiguity</i> arises when a word refers to either a volatile or a persistent property of an object.
Referential Ambiguity	Can a phrase in a requirement refer to more than one object in other requirements? Check pronouns (it), definite noun phrases (the roads), and ellipses (... If not, ...).
Discourse Ambiguity	Does a requirement have several interpretations in relation to other requirements? This ambiguity arises when (1) words such as <i>first</i> , <i>before</i> , <i>between</i> , <i>after</i> , and <i>last</i> are used and can refer to several elements and when (2) adjectives, verbs, or noun phrases refer to more than one condition described before.
Domain Ambiguity	Is the requirement ambiguous with respect to what is known about the application, system, or development domain?

Table 3: Ambiguity in NL Requirements (Kamsties, Berry, & Paech, 2001)

By describing the steps for ambiguity detection using this checklist, they argued in favor of manual inspection and stated that current NLP tools are not apt for proper disambiguation of NL requirements; rather, they are misleading. Their work also demonstrated dependence on formal specifications, e.g. UML models, especially for detecting domain ambiguities. Their suggested heuristics for detecting ambiguities involve attempting to develop UML models, and finding the points of contradiction and lack of information in the requirements specification. They recommended this process to be carried out by manual manipulation only. Their study concludes with the statement “one cannot expect to find all ambiguities in a requirements document with realistic resources” – even with such complete human involvement (Kamsties, Berry, & Paech, 2001).

Manual detection is typically the most accurate approach; however, it is also the most expensive. Again, use of formalization is not well-understood by non-technical users as well. We also find (Letier, Kramer, Magee, & Uchitel, 2005; Cyre, 1995) proposing the use of formal specifications to validate requirements.

B. Restricting Natural Language

Many other studies attempt to reduce the problems associated with unrestricted NL by limiting the scope of the language. Some use a new NL-like sublanguage, which severely limits the expressiveness of the requirements specifiers. Others propose to restrict the grammar to consider only a subset of NL while writing requirements document (Denger, Berry, & Kamsties, 2003; Fantechi, Gnesi, Ristori, Carenini, Vanocchi, & Moreschini, 1994; Rolland & Proix, 1992; Tjong, Hallam, & Hartley, 2006). Using a restricted language does simplify the task of maintaining the quality of textual requirements and keep them free of ambiguities, but imposes severe constraints on the requirements specifier's expression.

We will first look into the details of the study carried out by (Heinrich, Kemp, & Patrick, 1999), where they proposed the use of a restricted language, called "Flexible Structured Coding Language (FSCL)", thoroughly defined by a fixed set of grammatical rules. The advantage of FSCL is that it has an unrestricted vocabulary, and it claims to be unambiguous enough to be translated into programming code automatically. Though the paper never defined the process of translation, the grammar it used has the potential to be unambiguous because of its strictness.

(Fantechi, Gnesi, Ristori, Carenini, Vanocchi, & Moreschini, 1994) suggested the use of a set of grammatical rules for aiding the translation from NL requirements specification to the formulae of "action-based temporal logic", called ACTL. They also have a domain-specific dictionary that helps the translation process. The grammar they defined can only deal with the possible structures of those NL sentences, which describes an expression of ACTL. This makes their grammar very limited for parsing a real requirements specification document. The ambiguities they could detect in their case study using this process were due to lack of information in the time and the quantification of an expression only.

A study conducted by (Rolland & Proix, 1992) translated natural language requirements specification to a form of semantic net, allowing a broad logical representation in conceptual schema. This required the use of a dictionary grouping verbs in six major categories: Agentive, Instrumental, Dative, Factitive, Locative and Objective. Each such category led to define a fixed set of grammar rules for parsing NL requirements statements into case

notations. Their rules, thus, restrict the grammar of natural language used in specification. After translation, their system then follows a “paraphrasing process”, using Chomsky’s transformational grammar (Chomsky, 1965), to translate the conceptual schema of case notations back to NL-based statements. This allows the requirements elicitor to compare the natural language requirements specification given as input to the system, and with the one received as output from the system, and detect ambiguities. The work thoroughly relies on a fixed set of grammar rules and, although, claims to work with requirements written in NL, the case study they presented worked with an example requirements document that contained sentences of a very simple structure, targeted to be caught by their fixed set of rules.

C. Using NLP Tools on Unrestricted Language for Requirements

NLP techniques have advanced at tremendous speed during the past few years. For over a decade now, researchers in the fields of both NLP and software engineering, have been trying to merge NLP techniques with the tasks of requirements engineering. We know that requirements elicitation and validation is one of the key-phases of software’s lifecycle that often takes considerably long time to finish with manual manipulation of information. A real-life requirements document can be lengthy and contain numerous words, phrases and sentences, where each of them becomes a candidate for possible ambiguities of different kinds. All these reasons made way for NLP techniques to come into the picture for tackling this problem. Researchers have introduced NLP in many different ways to detect ambiguity in requirements specification. The next sections present a brief survey on some of the most important research work in this area.

QuARS

(Fabbrini, Fusani, Gnesi, & Lami, 2001) and (Gnesi, Lami, & Trentanni, 2005) addressed the issue by trying to measure the quality of a problem description, written in unrestricted NL. They initially made a survey on the contemporary studies revealing a number of defects that can exist in an NL requirements specification and listed those defects as “indicators” of poor-quality requirements specification. These are shown in Table 4 [extracted from (Gnesi, Lami, & Trentanni, 2005)].

Characteristic	Indicators
Vagueness	<i>The occurrence of Vagueness-revealing wordings (as for example: clear, easy, strong, good, bad, useful, significant, adequate, recent, ...) is considered a vagueness Indicator</i>
Subjectivity	<i>The occurrence of Subjectivity-revealing wordings (as for example: similar, similarly, having in mind, take into account, as [adjective] as possible, ...) is considered a subjectivity Indicator</i>
Optionality	<i>The occurrence of Optionality-revealing words (as for example: possibly, eventually, if case, if possible, if appropriate, if needed, ...) is considered a optionality Indicator</i>
Implicitly	<i>The occurrence of: - Subject or complements expressed by means of: Demonstrative adjective (this, these, that, those) or Pronouns (it, they...)or - Terms having the determiner expressed by a demonstrative adjective (this, these, that, those) or implicit adjective (as for example previous, next, following, last...) or preposition (as for example above, below...) Is considered an implicitly Indicator</i>
Weakness	<i>The occurrence of Weak verbs is considered a weakness Indicator</i>
Under-specification	<i>The occurrence of words needing to be instantiated (for example: flow instead of data flow, control flow, .. , access instead of write access, remote access, authorized access, ... , testing instead of functional testing, structural testing, unit testing, .., etc.) is considered an under-specification Indicator.</i>
Multiplicity	<i>The occurrence of sentences having multiple subject or verb is considered a multiplicity Indicator</i>

Table 4: Different Quality Indicators of NL Requirements (Gnesi, Lami, & Trentanni, 2005)

Their studies proposed the use of their tool, called “*QuARS: Quality Analyzer for Requirements Specification*”, for detecting sentences exhibiting different kinds of ambiguity in a problem description. Their tool first performs a lexical analysis over a problem description using a POS tagger. It also syntactically parses the sentences using the MINIPAR parser (Lin, 2003), and finally, it combines both results for detecting the indicators of poor-quality requirement specification. It also contains an interface, called “View”, for the requirements engineer to view the requirements statements by “clusters” having all the requirements regarding a specific function or property together. At every stage of processing, their tool requires the use of a different “modifiable” dictionary, which is specially created and modified for a particular stage of processing and for a specific problem domain by the requirements engineer. Their idea heavily depends on using a set of such special dictionaries,

whose relevance and practical usage is uncertain. They developed their tool as a prototype for their idea, and it is said to produce a quality metrics of NL requirements specification. Again, in our view, their quality metrics are not well-defined to classify a problem description as ambiguous.

ARM

An automated tool for measuring the quality statistics of NL requirements documents, called “ARM: Automated Requirements Measurement”, was developed by Software Assurance Technology Center (SATC) of NASA. Its developers, (Wilson, Rosenberg, & Hyatt, 1996), presented nine categories of quality indicators for requirements specification in detail. They are: **Imperatives, Continuances, Directives, Options, Weak Phrases, Size, Specification Depth, Readability** and **Text Structure**. The first five of these categories are based on frequencies of specific words occurring in ambiguity raising contexts. The remaining four are related to the organization of the entire requirements specification document. The results, derived from using the ARM tool, appeared to be more effective than others at detecting the level of ambiguity, but they ignored the use of more advanced NLP methods, e.g. morphological analysis and syntactic analysis, which could have pointed out more ambiguity issues.

Newspeak

(Osborne & MacNish, 1996) used an NLP parser to derive all possible parse trees of each sentences in a requirements specification document. Their system, called “Newspeak”, then tries to detect ambiguity, if multiple parse tree exists for a particular sentence. Thus, their work only focuses on detecting ambiguous syntactic structure of sentences only, and do not deal with semantics or even individual ambiguous keywords.

Circe

The work of (Ambriola & Gervasi, 1997) attempts to validate NL Specification with the aid of the user after deriving a conceptual model automatically from the requirements documents by their tool called Circe. Although their tool being funded by IBM is now available as a plug-in for Eclipse and is used in practical fields, it still does not consider the existence of

ambiguities, which can corrupt their conceptual model, making the errors tough for a user to detect from the model later on.

Our Work on Ambiguity Detection

Our own work (Hussain, Ormandjieva, & Kosseim, 2007) addressed the problem of detecting ambiguities in textual requirements documents. Acknowledging the fact that none of the previous work has tested the applicability or performance of using a text classification system to automate such detection process, this work demonstrated that the approach of using a text classifier is applicable in the practical fields for detecting ambiguous passages in requirements documents. The work also encompassed some related important topics, e.g. how difficult it is to detect ambiguity manually from requirements documents and how the automatic tools developed can compare to human performance.

To prove our concept, we developed a text classification system that can detect ambiguity in a requirements document by classifying its passages as ambiguous or unambiguous. The system yielded high accuracy in performance demonstrating impressive results with 86.67% accuracy using 10-fold-crossvalidation technique. Comparing its results of how it agrees with the decisions of an expert, it outperformed human annotators with *average* expertise in detecting ambiguities. It can also be affirmed that the system will perform better in practical fields with the inclusion of new training data. We also built a prototype of this system, called *Requirements Specification Ambiguity Checker (ReqSAC)*, to demonstrate its use. We strongly believe that this system, with the potential to clean up ambiguities will not only serve our current research, but also be useful as a standalone application working in conjunction with the requirements specification writing tools. The prototype of this system is, therefore, implemented to run both as a standalone application and from within Eclipse and/or Rational XDE environment.

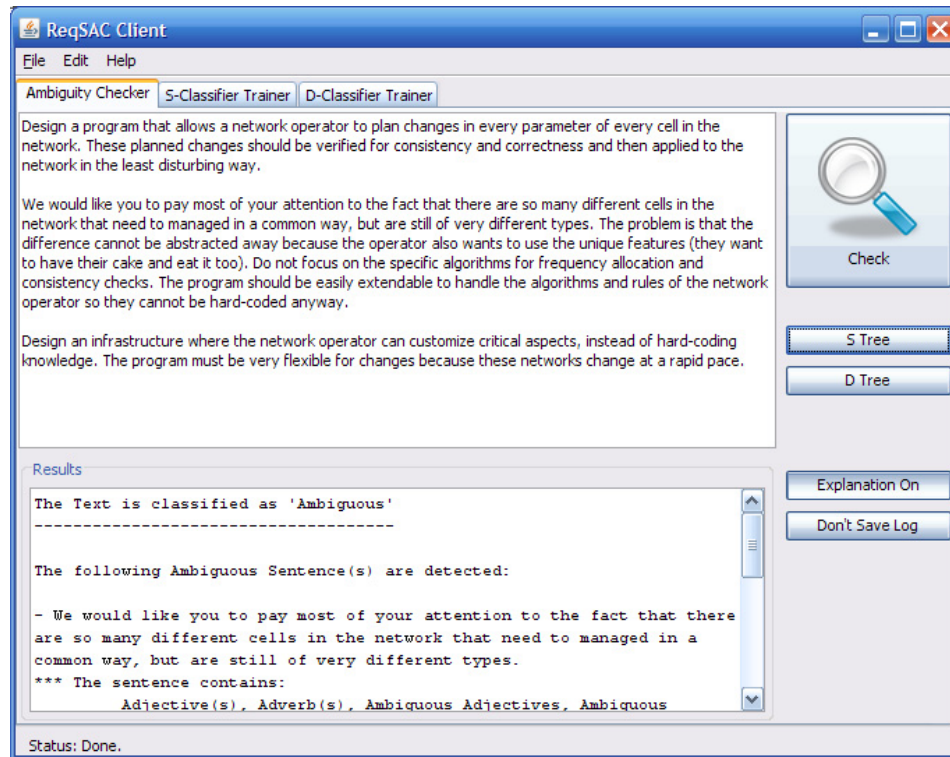


Figure 9: Requirements Specification Ambiguity Checker (ReqSAC)

Although our system established the idea of using a text miner successfully in detecting ambiguity from requirements documents, our future work should still focus on introducing more training data to improve its efficiency in dealing with unseen textual requirements.

3.4.2 On Extracting Functional & Non-Functional Requirements

Functional Size Measurement (FSM) approaches manipulate functional requirements only (ISO/IEC 14143-1, 2007), ignoring all the different classes of non-functional requirements completely in the initial phrases of measurement. However, in recent years, studies have tried to outline processes of quantifying non-functional requirements along with the functional size. (Kassab, Ormandjieva, Daneva, & Abran, 2008), for example, suggested to first use the NFR ontology (Kassab M., 2009) to realize different types of non-functional requirements into functional ones, and determine their weights of development complexity on effort estimation by regression over historical data. On the other hand, IFPUG recently proposed their SNAP framework (IFPUG, 2013) for assessing different types of non-functional requirements in terms of different SNAP categories of functionalities, which is then subjectively weighted by

human experts based on previous experiences of their complexities. Thus, to our knowledge, measuring the impact of non-functional size can still not be performed early from textual description of software requirements, we also suggest the impact NFR size to be considered along with effort estimation by means of performing regression analysis over historical data. Therefore, in relation to our work of automating the functional size measurement process in terms of the COSMIC standard (COSMIC, 2014), as mentioned in Section 1.3, we focus here on the related studies on automatic extraction of functional requirements (FR), to separate them from the non-functional requirements (NFR), from a collection textual requirements.

The current processes to extract non-functional requirements (NFR) from requirements documents mostly rely on manual inspection, where an analyst reads the texts to identify a sentence manually as FR or NFR following different approaches, e.g. (Chung & Sapakkul, 2006; Cysneiros & Leite, 2002; Hill, Wang, & Nahrstedt, 2004). Research in this field to automate the process of separating NFRs from requirements documents has been scarce.

A study by (Cleland-Huang, Settimi, Zou, & Solc, 2006) explored the use of text classification as an attempt to classify requirements statements into ten different classes, one class of FR and nine classes of NFR. As reported in their paper, their work attained a recall measure of 0.767 and a precision measure of 0.248 with their corpus, on average of the 10-class classification. The authors used a stemmer to stem the words of the documents, and then selected keywords based on their high probability of occurrences in NFR statements. Their system then classified a statement as NFR, if the density of those selected keywords in that statement exceeds a particular threshold, else, otherwise.

Some of the latest research work in classifying textual form of non-functional requirements into various classes were presented by (Rashwan, Ormandjieva, & Witte, 2013; Casamayor, Godoy, & Campo, 2009). (Rashwan, Ormandjieva, & Witte, 2013) presents a simple supervised learning-based text classification technique that uses word-level features and an SVN classifier to identify functional and different types of non-functional requirements. On the other hand, (Casamayor, Godoy, & Campo, 2009) uses a semi-supervised learning-based text classifier that uses a probabilistic classifier over word frequency to identify different types of non-functional requirements.

Our Work on FR-NFR Classification

Our work presented in (Hussain, Kosseim, & Ormandjieva, 2008) uses a text miner to classify textual requirements into functional requirements (FR) and non-functional requirements (NFR). It used the same corpus that was used by (Cleland-Huang, Settimi, Zou, & Solc, 2006), extracted linguistic features, e.g. the frequency of discriminating syntactic features (like cardinals, adjectives etc.) and the frequency of discriminating keywords belonging to different Parts-of-Speech (POS) categories in requirements sentences, and showed that using a decision tree-based text classifier trained and tested with linguistic features attain a high accuracy of 98.56% in classifying textual requirements into FR and NFR, and that is when 10-folds-cross-validation is performed over the data used by (Cleland-Huang, Settimi, Zou, & Solc, 2006).

3.4.3 On Identifying Domain Concepts & Their Attributes

COSMIC functional size measurement standard requires the knowledge of the domain concepts or conceptual entities as “data-groups” for a particular problem domain, and a human measurer is required to identify their attributes from textual requirements and associate them with the data-groups that they belong to. Many work (Yue, Briand, & Labiche, 2011) in the fields of databases and requirements engineering have addressed the tasks on automating the process of identifying the domain concepts and their attributes from unrestricted textual requirements by means of different natural language processing techniques. The work of (Harmain & Gaizauskas, 2000; Samarasinghe & S., 2005) apply rule-based approaches where predefined grammars are used over unrestricted textual requirements to identify the domain concepts and their attributes. On the other hand, the work of (Gelhausen & Tichy, 2007; Körner & Landhäußer, 2010; Landhäußer, Körner, & Tichy, 2014) extends the rule-based approaches by first extracting the thematic relationships between the agents and the patients of actions to distinguish between domain entities and their attributes. All of these studies mentioned here go further by extracting the relationships among these domain entities to derive static models, which however is not relevant for our work of identifying COSMIC’s conceptual artifacts only, i.e. the data-groups and data-attributes.

3.5 Tools for Requirements Annotation

Software requirements annotation is often performed manually on paper without any tool support. However, managing and executing any annotation work with multiple experts working as annotators on large sets of software requirements is a tedious process that involves several activities, including:

1. Pre-processing the requirements documents to extract structured instances ready for annotation.
2. Training and/or recruitment of human experts as annotators based on their levels of skill.
3. Running and administering the annotation tasks in a collaborative environment, where annotators can share domain knowledge.
4. Aggregating annotation data (preferably, in digital form) for computer-aided analysis.
5. Evaluating the gold standard annotation for each instance.
6. Analyzing the annotation data to study the performance of the annotation work.

The above steps would require enormous effort for any type of annotation work, if done manually without any tool support. Thus, many annotation tools have been released [e.g. (Bontcheva *et al.*, 2013; Amazon.com Inc., 2012; Ogren, 2006; Widlöcher & Mathet, 2009; Bertran, Borrega, Recasens, & Soriano, 2008)] to automate/semi-automate some of the steps mentioned above. Using an annotation tool helps reduce time for the collection of annotation data and the conversion of the annotation data to digital format. Also, web-based annotation tools provide a collaborative environment where annotation data collected from multiple experts can efficiently be synchronized to ensure the robustness of the data.

However, to our knowledge, none of the current textual annotation tools are tailored specifically to aid the requirements annotation tasks. Several natural language annotation tools have been proposed over the years, e.g. (Bontcheva *et al.*, 2013; Amazon.com Inc., 2012), some are open-sourced, while others are not, some targeted to be used for specific fields, while others are intended for general linguistic annotation purposes. We briefly discuss a notable few in the following sections.

3.5.1 GATE TeamWare

Among all the annotation systems that we analyzed, we consider GATE TeamWare (Bontcheva *et al.*, 2013) to be one of the most powerful and versatile annotation tools available that can be used for any linguistic annotation projects. The advantages of GATE TeamWare over most other annotation systems are:

- i.* GATE TeamWare allows seamless integration with powerful pipelines of the GATE platform, for pre- and post-processing of the natural language documents.
- ii.* GATE TeamWare supports both open-ended and closed-ended forms of annotation. It allows customization of the annotation schema, so that annotators can either choose from a predetermined static set of annotations, or define a new annotation type while annotating an instance.
- iii.* GATE TeamWare supports customization of multiple projects with different sets of pre- and post-processing pipelines and annotation schema.
- iv.* GATE TeamWare allows reporting of different status information of the annotation projects to the project curators for monitoring of the annotation work.
- v.* Integration of GATE TeamWare with GATE also helps in analyzing the annotations and measuring the degree of annotators' agreement over the GATE platform.

The above advantages make TeamWare, in our opinion, to be one of the best annotation systems available for linguistic annotation tasks. However, the annotation schema of TeamWare cannot represent hierarchical dependencies among annotation types (e.g. when $TypeA \rightarrow TypeB$, i.e. an annotation type $TypeA$ is functionally dependant on another annotation type $TypeB$). Also, TeamWare does not provide options for computing gold-standard annotations automatically.

3.5.2 Amazon Mechanical Turk

Amazon Mechanical Turk (AMT) (Amazon.com Inc., 2012) is a web application that is designed by Amazon to support any kind of Human Intelligence Task, and, therefore, can

also be used for linguistic annotation work. However, annotation work on AMT is inherently open to public access, where any annotator with or without necessary skills can contribute his/her annotation. AMT also provides a highly customizable interface through its API. The API allows integrating AMT's functionality to any custom-built web application. It also includes an option for publishing the annotation work privately that restricts annotators without required qualification from participating.

3.5.3 Knowtator

Knowtator (Ogren, 2006) is an annotation tool that integrates with Protégé (Stanford Center for Biomedical Informatics Research, 2014) to store complex relational annotation data over well-organized ontologies. The tool supports complex annotation tasks by recording hierarchical dependency among annotation types. However, it does not support any form of pre-processing over textual documents and the annotators always have to manually select the span of an annotation instance before annotating it. The tool works offline, providing minimal support for multiple annotators working concurrently.

3.5.4 Glozz

Glozz (Widlöcher & Mathet, 2009) is another annotation tool that allows an annotator to set the span of each annotation and annotate it according to a set model. Its strength is its WYSIWYG graphical presentation of relational annotation, where one annotated instance is related to another annotated instance (e.g. for co-reference annotation). It also implements a query language to search through the graphs of relational annotations. However, Glozz runs locally on one annotator's machine offline; and there is no support for managing the work of multiple annotators, nor for reviewing their annotations.

3.5.5 AnCoraPipe

AnCoraPipe (Bertran, Borrega, Recasens, & Soriano, 2008) is a simple annotation tool that provides support for linguistic annotation at multiple levels. Although the tool is locally installed restricting administration of multiple annotators' recruitment, it can compare annotation data collected from multiple annotators via remote repositories. However, it cannot pre-process documents for corpus creation.

3.6 Necessary Features of Annotation Tools

In this section, we discuss the features of an annotation tool that are necessary to adequately support the complex requirements annotation processes in practical scenarios. Practical usage of an annotation tool for complex requirements annotation tasks, like functional size measurement, as discussed in Section 2.7.2, demands some important features to be available with an annotation tool. Table 5 summarizes a comparison of these features that the annotation tools, described in Section 3.5, provide.

Features (Support for ...)	GATE TeamWare	Amazon Mechanical Turk	Knowtator	Glozz	AnCoraPipe
(1) Document Acquisition	Yes	Yes	No	No	Yes
(2) Document Pre-processing (e.g. automatic segmentation)	Yes	Limited (Yes, via API & external tools only)	No	No	No
(3) Administration on Annotators' Recruitment	Yes	Limited (Yes, via API only)	No	No	No
(4) Customization of Annotation Interface	No	Limited (Yes, via API only)	No	No	No
(5) Relational Annotation	Yes	Limited (Yes, via API only)	Yes	Yes	Yes
(6) Hierarchical Dependency Among Annotation Labels	No	No	Yes	No	No
(7) Aggregating Annotation Data of Multiple Annotators	Yes	Yes	Yes	No	Yes
(8) Computation of Gold-standard Annotations	No	Limited (Yes, via API only)	No	No	No

Table 5: Comparison of Features Provided by Current Annotation Tools

We discuss below the features listed in Table 5 in relation to their necessities in functional size measurement activities.

3.6.1 Document Acquisition

The document acquisition feature represents the existence of a functionality in the annotation tools that facilitates secured interfaces to collect the requirements documents and maintain a document repository over a distributed work environment. The feature is not mandatory in supporting the annotation tasks related to functional size measurement. However, it helps to support large-scale annotation tasks over distributed environments, which are now

increasingly common for large software development projects in the industries. As shown in Table 5, most of the annotation tools that we tested support this feature.

3.6.2 Document Pre-processing

Here, document pre-processing refers to the features of an annotation tool that prepare the requirements instances for annotation. In case of functional size measurement, we need the annotation tool to extract the sentences and the noun-phrases from the requirements documents automatically before commencing the annotation tasks. As shown in Table 5, only GATE TeamWare and Amazon Mechanical Turk amongst the other annotation tools that we tested, support automatic segmentation of documents allowing extraction of the sentences and the noun-phrases.

3.6.3 Administration of Annotators' Recruitment

The feature of administering annotators' recruitment allows an annotation tool to restrict unauthorized access to the annotation tasks and allows the curator to recruit of a person as an annotator based on his/her background. In case of functional size measurement, the annotation tasks usually require persons with certain backgrounds (e.g. software engineers, requirements analysts, measurement experts etc.) to be recruited for the tasks. Also, most of the industrial software projects in practice are private projects that demands secured access to its requirements documents limited to only authorized users. As shown in Table 5, only GATE TeamWare and Amazon Mechanical Turk amongst the other annotation tools that we tested support administration of annotators' recruitment.

3.6.4 Customization of Annotation Interface

This feature of customizing annotation interface allows its annotation interface to be customized by the curator of the annotation tasks. Now, functional size measurement requires a specialized annotation interface that can guide the annotators to annotate the sentences first and then annotate its corresponding noun-phrases. The required interface should also be allowed to be customizable for different annotation tasks to facilitate different methods of functional size measurement. Moreover, the annotation interface needs to provide specialized feedbacks to the annotators about the computed functional size after the completion of the

annotation tasks. As shown in Table 5, only Amazon Mechanical Turk amongst the other annotation tools that we tested support this level of customization of its annotation interface via its API.

3.6.5 Relational Annotations

The feature of an annotation tool for assigning relational annotations allows an annotator to relate one annotation type with another while performing an annotation task. In case of functional size measurement, if an annotator annotates a base noun-phrase as a data-attribute, then the annotator needs to relate that annotation to the sentence that he/she annotated as a functional requirement. As shown in Table 5, all of the annotation tools that we tested support this feature.

3.6.6 Hierarchical Dependency Among Annotation Labels

Ensuring hierarchical dependency among annotation labels by an annotation tool dynamically limits the choices of annotation labels for an annotator based on an annotation label that he/she has chosen earlier for another instance. In case of functional size measurement, this feature allows an annotator to annotate the base noun-phrases with the data-attribute annotation labels, only when he/she has already annotated their root sentence as a functional requirement. As shown in Table 5, only Knowtator amongst the other annotation tools that we tested support this feature.

3.6.7 Aggregating Annotation Data of Multiple Annotators

Aggregating annotation data of multiple annotators allows an annotation tool to organize annotation data in such a way that multiple annotators can provide their annotations for the same requirement instance. The annotation tool equipped with this feature can aggregate the annotation data of multiple annotators on demand for the curator to analyze the data and generate performance evaluations. This feature is not mandatory for the annotation tasks related to functional size measurement, as the measurement work can be carried out by one expert only. However, it is recommended size measurement should be performed in groups (i.e. involving more than one annotators), when the experts are not available (Aiello, Alessi, Cossentino, Urso, & Vella, 2007), or when the requirements are informally written (Cohn,

2005). In addition, when the requirements annotation data is used for research, multiple annotators need to be involved in the annotation tasks to control bias. Also, large scale annotation tasks require multiple annotators to reduce the workload per annotator. Thus, for all these cases, an annotation tool needs to be capable of organizing and aggregating annotation data of multiple annotators. As shown in Table 5, most of the annotation tools that we tested support this feature.

3.6.8 Computation of Gold-standard Annotations

When multiple annotators annotate the same requirement instances, the computation of gold-standard annotations allows an annotation tool to use different statistical measures to automatically compute the gold-standard annotations for each instance. In case of functional size measurement with multiple annotators, as described in the previous paragraph, an annotation tool is required to have this feature, so that it can automatically determine the gold-standard annotations first, and then compute the functional size based on the gold-standards. As shown in Table 5, only one of the annotation tools that we tested, i.e. the Amazon Mechanical Turk, provides minimal support for this feature; that is only when this feature is programmed via its API. Most of the annotation tools we tested supports manual computation of the gold-standard annotations.

It should be noted here that the above study of the required features of a requirements annotation tool, guided us to develop a unique annotation tool, called LASR, which supports all of the features listed in this section. The tool also includes some additional features to specifically aid the functional size measurement tasks that are not supported by any of the tools discussed above. Section 5.2 describes the additional features of LASR and how they support the tasks of functional size measurement.

3.7 Effort Estimation Techniques

As discussed in Section 2.5, there have been an overwhelming number of studies performed, [for example, *BESTweb* (Jørgensen & Shepperd, 2007) lists of 1,242 related studies at the time of writing this thesis] that are often repetitive (Fraser, Boehm, Erdogmus, Jorgensen,

Rifkin, & Ross, 2009), all in attempts of introducing and validating different approaches of estimating software development effort. Although, all the solutions proposed in the literature acknowledge software size to be the primary factor driving effort and depend, directly or indirectly, on some FSM techniques for early effort estimation, the existing literature, to the best of our knowledge, does not report on automation of any of the existing FSM processes that receives textual requirements as input to begin the task of estimation at the requirements specification phase. We continue the discussion from what is discussed in Sections 2.5 and 2.6, and present some of the most crucial work in this field in following Sections.

3.7.1 Using Parametric Models for Effort Estimation

Many parametric models of estimating software development effort have been proposed in the literature. We will be discussing some of the popular parametric models used in context of estimating software development effort in this Section.

(1) COCOMO & COCOMO II

The Constructive Cost Model, widely referred to as COCOMO, is the most popular and extensively studied open parametric model for software effort estimation. It was first put fourth by Barry Boehm in (Boehm B., 1984), and, later, as COCOMO II in (Boehm, *et al.*, 2000).

The estimated effort in COCOMO is counted in person-months or person-hours, and the equation of its model (COCOMO II) is as follows:

$$Effort = A \times (Size)^B \times \prod_{i=1}^{17} EffortMultiplier_i$$

Here, A = A Multiplicative Constant

$$B = 1.01 + 0.01 \times \sum_{j=1}^5 ScalingFactor_j$$

The input variables of this model are software size, 17 effort multipliers and 5 exponential scaling factors that are presented in the following section.

Input Variables (or Cost Drivers) in COCOMO II

The primary variable in COCOMO models is software size that is measured in KSLOC (thousand source lines of code). The other variables that are considered as scaling factors and effort multipliers in COCOMO II are shown in Figure 10.

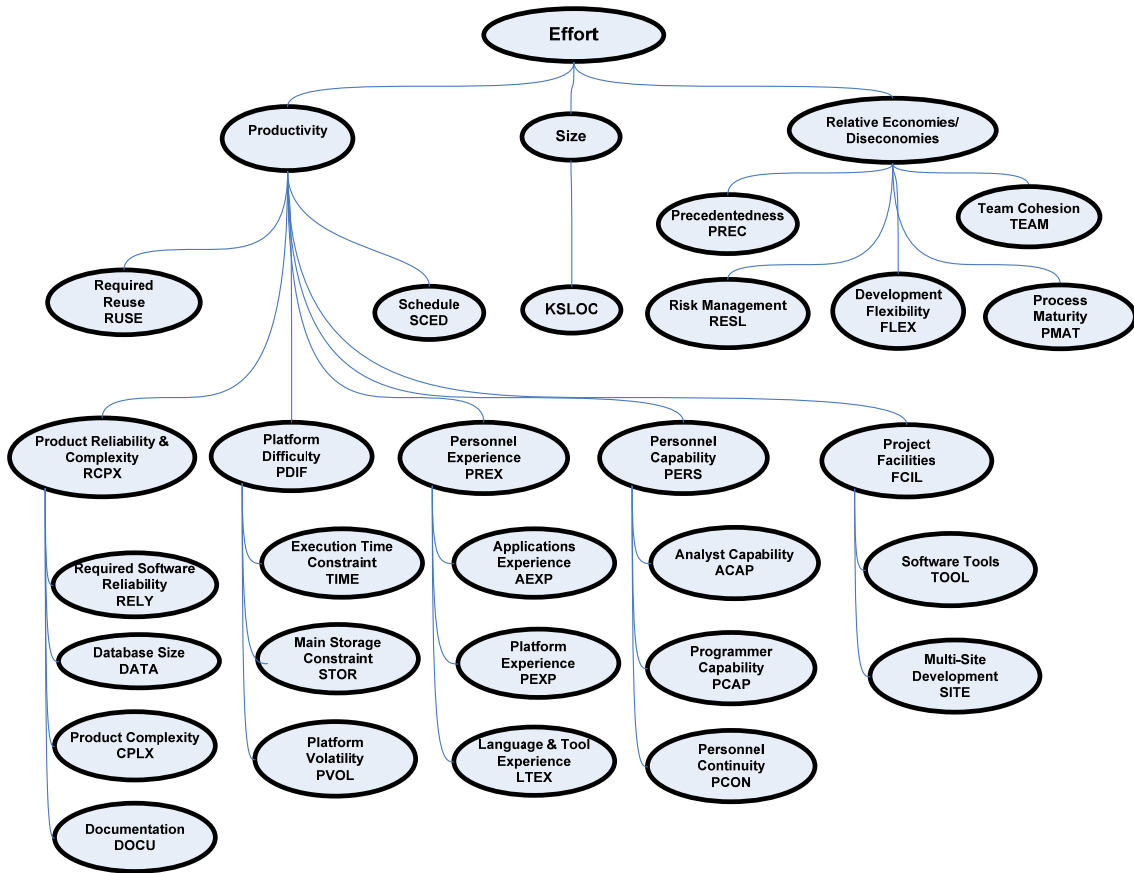


Figure 10: Cost Drivers in COCOMO II

In addition to the primary variable of size, Figure 10 shows the five exponential scaling factors to software size, under the group “Related Economies/Diseconomies”. They are: Precedentedness (PREC), Development Flexibility (FLEX), Risk Resolution (RESL), Team Cohesion (TEAM) and Process Maturity (PMAT). All these factors, except PMAT, can be set to the nominal values: *Very Low*, *Low*, *Nominal*, *High*, *Very High* and *Extra High*, while PMAT takes the weighted average of “Yes” answers to a questionnaire on Capability

Maturity Model (CMM) level of a an institute. The rest seventeen factors are effort multipliers that also take the nominal values, similar to the exponential scaling factors. The values are — *Very Low, Low, Nominal, High, Very High* and *Extra High* — which are manually set by human estimators.

Critical Discussion

The COCOMO and the COCOMO II models consider software size as its primary variable, where size is to be estimated in number of SLOC (Source Lines of Code) before starting the estimation process. The use of SLOC in estimation is thoroughly criticized by (Jones, 1997), pointing out that the use of SLOC estimate as software size degrades the quality of the input data, as the estimation process of SLOC includes parameters that depend on the subjective judgment by the estimators. This problem was evident in the study of (Kemerer, 1987) where SLOC-based estimation models performed poorly compared to non-SLOC-based estimation models.

Also, each of the other input variables (scaling factors and effort multipliers) to the COCOMO II model are manually set to one of the six different subjective nominal values (*Very Low, Low, Nominal, High, Very High* and *Extra High*) by the estimators, which can often become inconsistent being affected by the estimators' biases and different stakeholders influences on the estimators. For example, a scenario is depicted in (McConnell, 2006), naming it as *Estimation Politics*, where an estimator gets forced by the executive to unjustifiably tweak the values of the COCOMO II variables of “Programmer Capability (PCAP)” and “Analyst Capability (ACAP)” from below nominal to nominal category. Author notes that this irrational change results in a 39% reduction to the estimated effort, when using the COCOMO II model. Similar problems are mentioned in (Jørgensen & Molokken-Ostvold, 2004; Lederer & Prasad, 1991).

(2) SLIM

SLIM (Software Lifecycle Management) is another popular model developed using the theories presented in (Putnam, 1981; Putnam & Meyers, 1992). It is now incorporated in the commercial tool, called *SLIM-Estimate*, produced by Quantitative Software Management, Inc. (QSM).

The equation of the SLIM model to estimate the effort is as follows:

$$Effort = \left[\frac{Size}{ProcessProductivity \times (Time)^{\frac{4}{3}}} \right]^3 \times B$$

Here, effort is usually estimated in person-months or person-years.

Input Variables in SLIM

Similar to COCOMO, the primary variable in SLIM is software size that is measured in the number of SLOC (source lines of code), and needs to be estimated beforehand as an input to estimate the effort.

Process Productivity is a non-linear (often, exponential) variable that denotes the capability of an organization to produce a certain amount of source code with a given amount of effort and within an available time-frame. Its value is recommended to be chosen by calibrating the following equation with historical data:

$$ProcessProductivity = \frac{Size}{\left(\frac{Effort}{B} \right)^{\frac{1}{3}} \times (Time)^{\frac{4}{3}}}$$

B is a multiplicative constant, which is usually selected from values in a lookup table that increases with the software size and usually accounts for the effort in system integration testing.

When historical data is not available, the tool, *SLIM-Estimate*, developed by QSM, uses an expert system-based approach to select the value of Process Productivity considering many factors, such as the type of the system, the environmental factors (e.g. the programming language, the tools, methods, practices, database usage, use of standards etc.), the experience of the personnel, the management constraints (e.g. maximum allowable time, maximum cost, maximum and minimum staff size, required reliability), the economic factors (e.g. labor rates) etc. Most of the values of these variables can be objectively counted. When historical data is available within an organization, the non-linear Process Productivity value is converted to

linear Productivity Index (ranging from 1 to 40) to rate the productivity of different teams and scenarios within the organization.

Critical Discussion

Like the COCOMO models, the effort estimation model of SLIM also considers software size as its primary variable, where size is to be estimated in number of SLOC (Source Lines of Code) before starting the estimation process. Like the SLOC-based size measures, the SLIM model also suffers from the same problems as COCOMO, which was shown in the study of (Kemerer, 1987) where both COCOMO and SLIM models performed poorly compared to non-SLOC-based estimation models.

Also, when the historical data is absent, the model relies on the expert system-based approach that collects the estimator's answers to a series of 22 different questions to recommend a value of Process Productivity variable that was shown by (Kemerer, 1987) to failing to capture the essence of productivity in their study.

(3) Other Models

Other notable, but commercial, parametric estimation models include: ESTIMACS (Rubin, 1983) and SEER-SEM (Galorath, 2008). Both of these models use software size as their primary variables, but ESTIMACS uses Function Points (discussed in Section 2.7) as its only measure for software size, while SEER-SEM requires an SLOC-like measure, called *Effective Size* for its model (it includes formulas to convert function points and SLOC measures of software size to effective size). Since these models are commercially available, details on them have not been published.

3.7.2 Estimation by Analogy (EBA)

Authors of studies like (Shepperd & Schofield, 1997; Idri, Abran, & Khoshgoftaar, 2002; Angelis & Stamelos, 2000) presented the approach of effort estimation by analogy (EBA) that extends the idea of Case-Based Reasoning (Aamodt & Plaza, 1994). The approach relies extensively on the historical database of the past completed projects.

The idea is — first, to extract enough possible information on different attributes of the software project, the effort of which is to be estimated; then, to select projects from the

historical database similar to the target project; then, to compare the attributes of the target project to those of the similar project to find the multiplicative factors associated with each of the attributes; and, finally, to estimate the size and the corresponding effort for each of the attributes by multiplying the multiplicative factor to the size and the required effort of the attributes of the similar projects respectively.

Studies, advocating EBA claimed that EBA performed better than parametric methods. Moreover, several studies show that contradictory or inconclusive results were obtained when different parametric models and EBA method were applied on the same datasets (Myrtveit, Stensrud, & Shepperd, 2005; Menzies, Zhihao, Hihn, & Lum, 2006). Also, EBA cannot be performed very early in the requirements specification phase, as it is usually dependant on many of the design attributes of the target software to be known beforehand.

3.7.3 Delta Estimation

(Boehm B., 2000) introduced the concept of delta estimation, which attempts to estimate effort of a new project, by taking into account only the small changes (*delta*) to the cost drivers of the previous project. The author describes it as a “safe” method to be used in conjunction with the COCOMO I and II models.

However, (Menzies, Chen, Port, & Hihn, 2005) experimented with the COCOMO’81 dataset (Boehm B., 1981) and COCOMO NASA datasets of 60 software projects at NASA, both of which are available at the PROMISE repository (Sayyad Shirabad & Menzies, 2005), and showed that considering the changes to all the COCOMO cost drivers of the model can result in erroneous estimates.

3.7.4 Calibration and Use of Machine Learning Algorithms

All the parametric estimation models, described in Section 3.7.1, require to be calibrated by data from previous projects. For example, the COCOMO II equation for effort estimation requires its multiplicative constant, A , to be determined by calibration of the model with past completed projects. Similarly, it is recommended that the value of the Process Productivity variable in the SLIM model be determined by calibration of the model. It is assumed that a

large number of past projects used for calibration can yield better accuracy of the estimation models.

There have been uses of many different machine learning algorithms in recent studies for calibrating the effort estimation models. Also, these algorithms have been used to determine the multiplicative factors in support of estimation by analogy. The algorithms that were used in some of the important studies in software effort estimation are listed below:

- (1) **Regression Algorithms:** Used in (Heiat, 2002; Jørgensen M., 2004b; Levine & Hunter, 1983; Miyazaki, Terakado, Ozaki, & Nozaki, 1994)
- (2) **Bayesian Algorithms:** Used in (Chulani, Boehm, & Steece, 1999; Mendes & Mosley, 2008; Pendharkar, Subramanian, & Rodger, 2005; Stamelos, Angelis, Dimou, & Sakellaris, 2003)
- (3) **Neural Network Algorithms:** Used in (Dawson, 1996; Flitman, 2000; Hakkarainen, Laamanen, & Rask, 1993; Idri, Khoshgoftaar, & Abran, Can neural networks be easily interpreted in software cost estimation?, 2002; Park & Baek, 2008; Zhang, Patuwo, & Hu, 1998)
- (4) **Genetic Algorithms:** Used in (Shukla, 2000; Burgess & Lefley, 2001; Chang, Christensen, & Tao, 2001; Huang & Chiu, 2006; Shan, McKay, Lokan, & Essam, 2002; Braga, Oliveira, & Meira, 2008)

Although, in our view, the publicly available datasets often used in these studies, are inconsistent because of the differing quality of data corrupted with estimators' biases and the differing sources of data originating in differing environmental constraints from one company to the other. This inconsistency of datasets led these studies to attain average or below average results. Thus, many studies, for example (Mendes, Martino, Ferrucci, & Gravino, 2007; Kemerer, 1987), have shown that local calibration within a single company performs better than global cross-company calibration. Thus, it is recommended by most researchers to calibrate the parametric models of effort estimation using historical data of local projects (Fraser, Boehm, Erdogmus, Jorgensen, Rifkin, & Ross, 2009).

3.8 Conclusion

Despite so many studies carried out and so many methods introduced in the last three decades on software effort estimation, expert judgment remains the most popular and most used effort estimation method in the industry (Shepperd & Cartwright, 2001). Again, despite so many sophisticated algorithms used to attain better results in using effort estimation models, the studies of (Lederer & Prasad, 1992; Jørgensen M., 2004a) report that effort estimations done by expert judgment, on average, were at least as accurate as those done with estimation models. The work of (Lewis, 2001) showed that the main source of the problem is with the parameters or features of the estimation models that are subjectively computed inducing inconsistencies in the calibration data of the model. The author suggests that many of the previous studies presented overly exaggerated results because of the estimation biases. Also, most of the results of all these studies in software effort estimation are presented with the statistically unreliable measure of PRED (Conte, Dunsmore, & Shen, 1986), which many studies have criticized, like (Port & Korte, 2008; Foss, Stensrud, Kitchenham, & Myrtveit, 2003; Kitchenham, Pickard, MacDonell, & Shepperd, 2001).

Although all solutions proposed in the literature emphasize, directly or indirectly, on early effort estimation by the use of FSM, the existing literature, to the best of our knowledge, does not report on the automation of any of the existing FSM processes that receives textual requirements as input to begin the task of estimation at the requirements specification phase.

All these indicate that the research area of early effort estimation is still open, with much prospects in automating an FSM method like COSMIC that can objectively measure the size of the software.

Chapter 4

Methodology

“All things are difficult before they are easy.”
— *Thomas Fuller*

4.1 Introduction

The objectives of this research were introduced in Section 1.3. In this chapter, we will elaborate on these objectives setting a strictly defined set of research questions. We will then explain our related hypotheses and our methodology in details. The chapter will then present a brief overview of the metrics and tools used in our research.

4.2 Phases of Our Methodology

We conducted several empirical studies in this research in five different phases, as mentioned in Section 1.3. The phases were:

- Phase I:** FSM by Non-Experts
- Phase II:** Size Approximation
- Phase III:** Requirements Classification
- Phase IV:** FSM Model Extraction
- Phase V:** Evaluation of FSM Automation

The sequence of these phases, their inputs and outputs and their outcomes are illustrated in Figure 11.

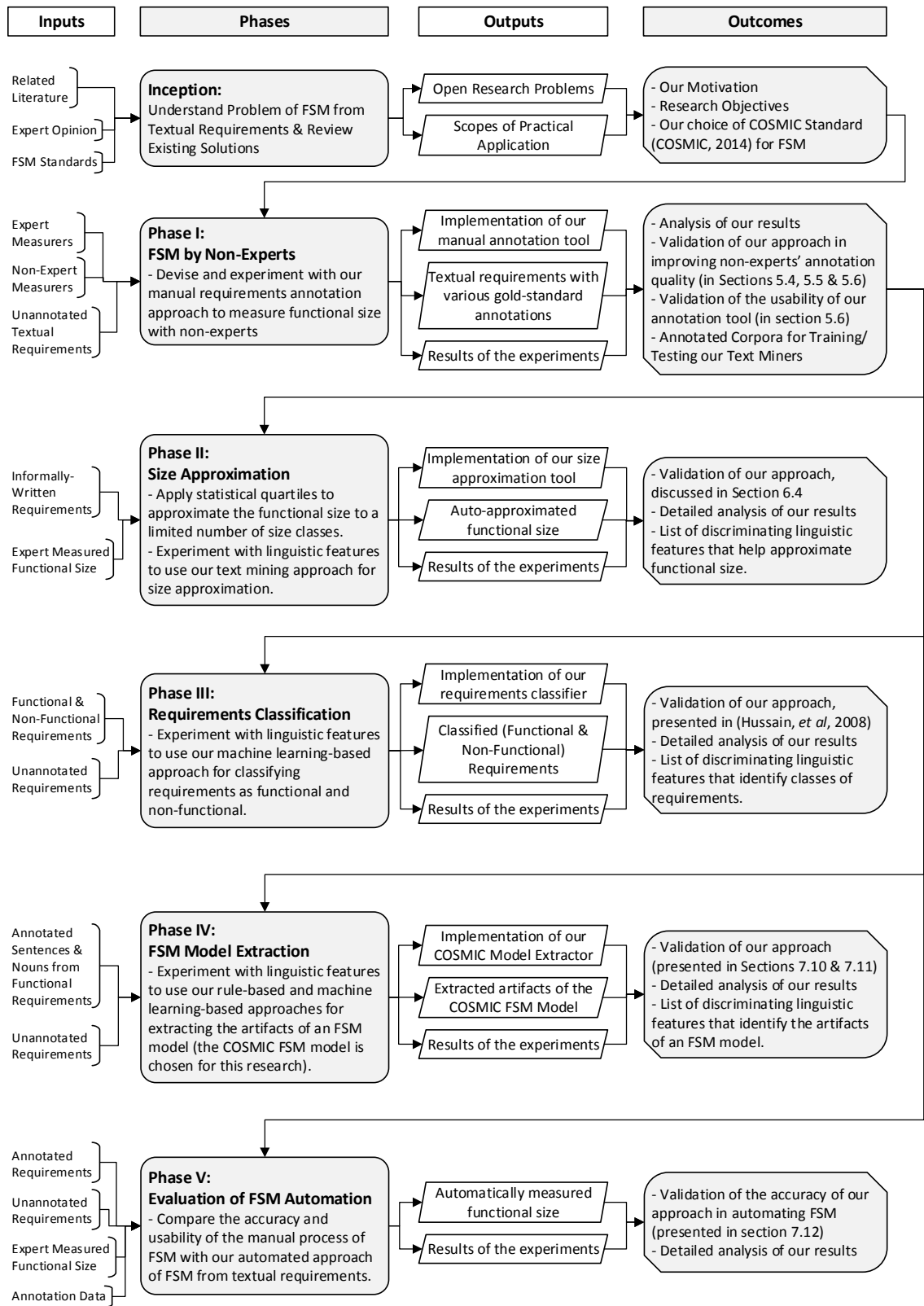


Figure 11: Phases of Our Methodology

The phases of our methodology, along with the initial inception phase, are described in details in the following sections.

4.2.1 Inception

We start our methodology with the aim to understand the problem of measuring functional size of a software from its requirements documents. Our target in this inception phase was to choose some of the leading work related to functional size measurement and to critically review their approaches to identify their strengths and weaknesses. This allowed us to select COSMIC as the FSM method that we would be experimenting with in this research, for its qualities of being objectively computed and applied early at the requirements specification phase. Thus, our study, in this phase, helped us formulate the objectives and the scope of our research, and also pointed out the lack of evaluative research to address the problem of measuring functional size early in the requirements specification phase.

4.2.2 Phase I: FSM by Non-Experts

The high costs induced by an expert in Functional Size Measurement (FSM), as discussed in Section 1.2, led us to choose the following objective for our research:

Objective #1: *To investigate if the process of functional size measurement can be executed effectively with non-expert.*

We first addressed this research objective in this phase. In this research, we intend to investigate if the task of functional size measurement can be done without engaging experts. The work of experts are costly and time-consuming. They have varying levels of expertise and use biased judgments that can introduce inconsistency in the outcome of the FSM tasks. Therefore, we extend our research with this objective by exploring if the task of FSM can be performed by non-experts. Fulfilling this objective can also help us in understanding the feasibility of achieving our objective #5, presented in Section 1.3.

Our Approach: Following this research objective, we first designed the FSM activities as annotation tasks to be performed on textual requirements. The details about our annotation task design are presented in Section 5.3 of Chapter 5.

We then ran experiments where an expert and a group of well-trained non-experts performed the FSM activities by annotating the same set of textual requirements.

Metrics: In the above experiments, we measured the accuracy of non-experts' annotation by their level of agreement with the expert's annotation in terms of Cohen's Kappa (Cohen, 1960) and set the baseline as moderate level of agreement following the evaluation scale of (Landis & Koch, 1977).

The research question that we targeted by our above experiments in relation to our objective is presented below, along our related null and alternative hypotheses:

Research Question, Q_I : "Can well-trained non-experts attain at least a moderate level of agreement with the expert for annotation tasks related to FSM?"

Null Hypothesis, $H_{I,0}$: Well-trained non-experts can never attain at least a moderate level of agreement with the expert for annotation tasks related to FSM.

Alternative Hypothesis, $H_{I,a}$: Well-trained non-experts can attain at least a moderate level of agreement with the expert for annotation tasks related to FSM.

The formalization of the above hypotheses, along with a detailed discussion on the related variables and metrics, are presented in Section 5.3, where we also discuss the details of our experiments and analyze our results.

During the above experiments, the annotation work of the non-experts required a two-week-long training and costly adjudication session afterwards to resolve the points of disagreements amongst the non-experts. Thus, in this phase, we extended our scope with an additional research objective as follows:

Objective #2: *To improve the overall process of FSM-related requirements annotation by attaining accurate annotations with non-experts having minimal training.*

Our Approach: Following this research objective, we developed a computer-aided manual annotation approach, along with a dynamic annotation adjustment model, that can help a

group of non-experts to perform the FSM related annotation tasks and achieve an acceptable level of agreement with an expert. We then implemented an online annotation tool, called Live Annotation of Software Requirements (LASR) to test out our annotation approaches. The implementation of LASR is briefly discussed in Section 5.8.

We then ran controlled annotation experiments using LASR to compare the accuracy of both well-trained and minimally-trained groups of non-experts for these FSM related tasks.

Metrics: In the above experiments, we measured the accuracy of non-experts' annotation by their degrees of agreement with the expert's annotation in terms of Cohen's Kappa (Cohen, 1960).

The first research question, which we targeted by our above experiments in relation to our objective #2, is presented below, along our related null and alternative hypotheses:

Research Question, Q_2 : "Which type of FSM-related manual annotation tasks performed by non-experts attains a higher accuracy: the manual annotation task performed by well-trained non-experts, or the LASR-aided annotation task performed by minimally trained non-experts?"

Null Hypothesis, $H_{2,0}$: The FSM-related manual annotation tasks performed without LASR, but by well-trained non-experts and with disagreements resolved through the adjudication session, always attain a higher accuracy than the LASR-aided manual annotation tasks performed by minimally trained non-experts and with no adjudication process.

Alternative Hypothesis, $H_{2,a}$: The LASR-aided manual annotation tasks related to FSM that are performed by minimally-trained non-experts with no adjudication process, can attain an equal or higher accuracy than the manual annotation tasks performed by well-trained non-experts without LASR, but with disagreements resolved through the adjudication session.

The next research question, which we targeted by our above experiments in relation to our objective #2, is presented below, along our related null and alternative hypotheses:

Research Question, Q_3 : “Which type of FSM-related manual annotation tasks performed by non-experts finishes quicker: the manual annotation task performed by well-trained non-experts, or the LASR-aided annotation task performed by minimally trained non-experts?”

Null Hypothesis, $H_{3,0}$: The FSM-related manual annotation tasks performed without LASR, but by well-trained non-experts, always finish quicker than the LASR-aided manual annotation tasks performed by minimally trained non-experts and with no adjudication process.

Alternative Hypothesis, $H_{3,a}$: The LASR-aided manual annotation tasks related to FSM that are performed by minimally-trained non-experts, can finish at the same time as or quicker than the manual annotation tasks performed by well-trained non-experts without LASR.

The formalization of the above hypotheses, along with a detailed discussion on the related variables and metrics, are presented in Sections 5.3 to 5.7, where we also discuss the details of our experiments and analyze our results.

4.2.3 Phase II: Size Approximation

In phase II, we addressed the open research problem #1, as presented in Section 1.2, by formulating the following research objective:

Objective #3: *To determine the most discriminating linguistic features of informally written textual requirements for approximating functional size.*

Our objective aims to use informally specified software requirements for Functional Size Measurement so that the size can be measured at the earliest phase of software development lifecycle. Formalization of software requirements gradually improves requirements from its initial informal narration in textual form. The process is costly, requires expert intervention, delays the development and is often mistakenly avoided by the industry in practice to reduce cost and meet tight schedule. Although our intention is not against requirements formalization, we, however, want to investigate if functional size can be approximated without requirements formalization. Approximating FSM on informally written textual

requirements can allow effort estimation to be performed very early in the life cycle of a software project.

Our Approach: Following this objective, we first used statistical quartiles to label our sets of textual requirements by a discrete number of nominal classes based on their functional size measured by experts. We use these labels as gold-standards representing the approximated functional sizes for the respective sets of textual requirements. We then devised a text mining approach utilizing natural language processing tools, e.g. a part-of-speech tagger and a syntactic parser, to extract the linguistic features from our sets of textual requirements. Our approach then applies machine learning techniques to select the linguistic features that discriminate our sets of textual requirements the most, based on the nominal classes representing their approximated functional size. The details about our approach for the approximation of functional size are presented in Section 6.2.

We then ran our experiments to check if our text mining approach can predict the nominal classes representing the approximated functional size of unseen textual requirements and moderately agree with their gold-standard classifications.

Metrics: In the above experiments, we measured the level of agreement of the predicted class labels with the gold-standard labels in terms of Cohen's Kappa (Cohen, 1960) and set the baseline as moderate level of agreement following the evaluation scale of (Landis & Koch, 1977).

The research question, which we targeted by our above experiments in relation to our objective #3, is presented below, along with the related null and alternative hypotheses:

Research Question, Q_4 : “Can our text mining approach utilizing an automatically chosen set of discriminating linguistic features predict the nominal classes representing the approximated functional size of unseen textual requirements and at least moderately agree with their gold-standard classifications?”

Null Hypothesis, $H_{4,0}$: Our text mining approach utilizing the discriminating linguistic features cannot predict the approximated functional size classes of unseen textual requirements that at least moderately agrees with their gold-standard classifications.

Alternative Hypothesis, $H_{4,a}$: Our text mining approach utilizing the discriminating linguistic features can predict the approximated functional size classes of unseen textual requirements that at least moderately agrees with their gold-standard classifications.

A detailed discussion on our experiments to validate the above hypotheses, are presented in Section 6.3, where we also analyze our results.

4.2.4 Phase III: Requirements Classification

In phase III, we addressed the open research problem #2, as presented in Section 1.2, by formulating the following research objective:

Objective #4: *To explore the most discriminating syntactic features of textual requirements for classifying them into functional and non-functional requirements.*

Our objective here is to attain higher accuracy of classifying functional and non-functional requirements than the contemporary approaches by using the most discriminating syntactic features of textual requirements. We will be discussing the contemporary approaches in details in Section 3.4.2.

Our Approach: Following this objective, we first collected textual samples of functional requirements (FR) and non-functional requirements (NFR) that the contemporary studies used to report on the accuracies of their approaches. We assumed the labels (FR and NFR) for these already classified requirements as the gold-standards for our study. We then devised a text mining approach utilizing natural language processing tools, e.g. a parts-of-speech tagger and a syntactic parser, to extract the linguistic features from our textual samples of FR and NFR. Our approach then applies machine learning techniques to select the linguistic features that discriminate the most between our training sets of FR and NFR. The details about our approach for classifying textual requirements into functional and non-functional

requirements were presented in (Hussain, Kosseim, & Ormandjieva, 2008). We also include details about this approach in Section 4.4.2.

We then ran our experiments to check if our text mining approach can identify FR and NFR from unseen textual requirements and moderately agree with their gold-standard labels.

Metrics: In the above experiments, we again measured the level of agreement of the predicted class labels with the gold-standard labels in terms of Cohen’s Kappa (Cohen, 1960) and set the baseline as moderate level of agreement following the evaluation scale of (Landis & Koch, 1977).

The research question, which we targeted by the above experiments in relation to our objective #4, is presented below, together with the related null and alternative hypotheses:

Research Question, Q_5 : “Can our text mining approach utilizing an automatically chosen set of discriminating linguistic features identify functional and non-functional requirements from unseen textual requirements and at least moderately agree with their gold-standard classifications?”

Null Hypothesis, $H_{5,0}$: Our text mining approach utilizing the discriminating linguistic features cannot identify functional and non-functional requirements from unseen textual requirements and at least moderately agree with their gold-standard classifications.

Alternative Hypothesis, $H_{5,a}$: Our text mining approach utilizing the discriminating linguistic features can identify functional and non-functional requirements from unseen textual requirements and at least moderately agree with their gold-standard classifications.

A detailed discussion on our experiments to validate the above hypothesis, were presented in (Hussain, Kosseim, & Ormandjieva, 2008), where we also present our analysis of the results. The summary of the outcomes of these experiments are also included in Section 4.4.2.

4.2.5 Phase IV: FSM Model Extraction

In phase IV, we intended to build traceability of a functional size measurement (FSM) model by relating the elements of textual requirements with the artifacts of the model. We, thus,

addressed the open research problem #3, as presented in Section 1.2, by choosing the following research objective:

Objective #5: *To identify how the experts deduce the relationship between the linguistic elements of unrestricted textual requirements and the objects of interest in a functional size measurement model.*

This relates to our primary objective for this research that aims to learn the process of at least one functional size measurement (FSM) method in a way so that it can be applied over unrestricted textual software requirements. We intend to understand how a human expert measures functional size when he reads software requirements document. The conventional methods of FSM only records the numbers that leads to count the functional size of the software. Our goal is to make the process of measuring functional size transparent enough to provide traceable reasoning for all the decisions taken by an expert during functional size measurement. Capturing this knowledge and making FSM traceable do not only help during investigating the causes of any incorrect measurement, but also reduces the chances of introducing subjective judgments, as each of the judgments of the measurers would then be supported by detailed reasoning.

We found in our literature survey that the measurement task in COSMIC (ISO/IEC 19761, 2003; COSMIC, 2014), unlike the other FSM standards, can be performed without requiring the subjective judgment of human experts, but only when the software requirements are formalized and well-decomposed. Thus, we chose to emulate the COSMIC standard for our research, as it brings the manipulation of software requirements closer to be mapped on to the process of functional size measurement. Its manual lists the objects of interest of its FSM model that are to be extracted by an expert measurer through analyzing functional requirements. These objects are comprised of data-attributes belonging to data-groups and four different types of movements of the data-attributes. Thus, it is still dependent on the expertise of human measurers to analyze the functional requirements, prepare the COSMIC model and determine the COSMIC functional size.

Our approach: To learn which linguistic elements indicate the presence of which FSM modeling artifacts to the experts, we first devised a process of requirements annotation that

assigned a human expert to derive FSM modeling artifacts by carefully annotating different linguistic elements of the textual requirements. For example, in case of our studies using the COSMIC standard, the expert had to annotate the base noun-phrases in the functional requirement sentences as mentions of moving data-attributes belonging to certain *data-groups*, which are artifacts of the COSMIC FSM model. In their annotation, they also had to indicate what the types of movements these data-attributes are participating in. This associated the annotated noun-phrases with another kinds of COSMIC modeling artifacts, called the *data-movement types*. A detailed discussion on these COSMIC modeling artifacts are presented in Section 2.7.2 of Chapter 2.

We hold these expert annotations as the gold-standards representing the correct traceability links between the FSM modeling artifacts and the textual requirements. We then devised two different text mining approaches, both of which utilize natural language processing tools, e.g. a parts-of-speech tagger, a syntactic parser, gazetteers, syntactic rules etc., to first extract a pool of linguistic features from our textual samples of base noun-phrases belonging to the functional requirement sentences. Our first approach then applies a rule-based text mining technique that follows a series of our custom-developed heuristics to identify base noun-phrases that are linked to the FSM modeling artifacts using our pool of linguistic features. And, our second approach applies a supervised learning based text mining technique to dynamically select the linguistic features and rules that discriminate the most between the collection of base noun-phrases in our training dataset that can be linked to the FSM modeling artifacts, and the collection of base noun-phrases that cannot be linked. The details about both of our text mining approaches for classifying base noun-phrases as linked to different FSM modeling artifacts are presented in Chapter 7.

We then ran our experiments to check if our text mining approaches can better identify the base noun-phrases that are linked to the FSM modeling artifacts by agreeing more with the gold-standards.

Metrics: In the above experiments, we compared the accuracy of our two approaches by measuring their precision, recall, f-measure and their level of agreement with the gold-standard in terms of Cohen's Kappa (Cohen, 1960). We also set the baseline for both of our

approaches as moderate level of agreement following the Kappa-based evaluation scale of (Landis & Koch, 1977).

The first research question, which we targeted by the above experiments in relation to our objective #5, is presented below, along our related null and alternative hypotheses:

Research Question, Q_6 : “Can our supervised learning-based text mining approach utilizing our pool of linguistic features identify from unseen textual requirements the base noun-phrases⁴ as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards?”

Null Hypothesis, $H_{6,0}$: Our supervised learning-based text mining approach utilizing our pool of linguistic features cannot identify from unseen textual requirements the base noun-phrases as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards.

Alternative Hypothesis, $H_{6,a}$: Our supervised learning-based text mining approach utilizing our pool of linguistic features can identify from unseen textual requirements the base noun-phrases as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards.

The next research question, which we targeted by our aforementioned experiments in relation to our objective #2, is presented below, along our related null and alternative hypotheses:

Research Question, Q_7 : “Can our heuristics-based text mining approach utilizing our pool of linguistic features identify the base noun-phrases as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards?”

⁴ As mentioned earlier in Chapters 1 and 2, we use the term “base noun phrase” to refer to a noun or a noun compound or a personal pronoun. Thus, in our case, it actually refers the smallest part of the base noun phrase (Samarasinghe & S., 2005) that do not contain any part-of-speech class of word, other than nouns or a personal pronoun. It therefore represents the smallest segment of textual requirement that can independently express the mention of an artifact of a functional size measurement model.

Null Hypothesis, $H_{7,0}$: Our heuristics-based text mining approach utilizing our pool of linguistic features cannot identify the base noun-phrases as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards.

Alternative Hypothesis, $H_{7,a}$: Our heuristics-based text mining approach utilizing our pool of linguistic features can identify the base noun-phrases as being related to specific artifacts of a functional size measurement model and at least moderately agree with the gold-standards.

The final research question, which we targeted by our aforementioned experiments in relation to our objective #2, is presented below, along our related null and alternative hypotheses:

Research Question, Q_8 : “Which text mining approach utilizing our pool of linguistic features can identify with higher accuracy the base noun-phrases compounds as being related to specific artifacts of a functional size measurement model: our supervised leaning-based approach, or the heuristics-based approach?”

Null Hypothesis, $H_{8,0}$: There is no difference in accuracy between our supervised leaning-based text mining approach and our heuristics-based text mining approach when utilizing our pool of linguistic features to identify with higher accuracy the base noun-phrases as being related to specific artifacts of a functional size measurement model.

Alternative Hypothesis, $H_{8,a1}$: Our supervised leaning-based text mining approach attains a higher accuracy than our heuristics-based text mining approach when utilizing our pool of linguistic features to identify with higher accuracy the base noun-phrases as being related to specific artifacts of a functional size measurement model.

Alternative Hypothesis, $H_{8,a2}$: Our heuristics-based text mining approach attains a higher accuracy than our supervised leaning-based text mining approach when utilizing our pool of linguistic features to identify with higher accuracy the base noun-phrases as being related to specific artifacts of a functional size measurement model.

A detailed discussion on our experiments to validate the above hypotheses, is presented in Section 7.9 of Chapter 7, where we also analyze our results.

4.2.6 Phase V: Evaluation of FSM Automation

Finally, in phase V, we used our knowledge gathered from our research to implement the automation of our approaches of performing functional size measurement (FSM) from textual requirements. We intended to evaluate the feasibility of automating FSM by comparing its accuracy and its time-related efficiency to the manual process of FSM. We, thus, chose the following research objective:

Objective #6: *To evaluate the feasibility of automating functional size measurement from textual requirements.*

Fulfilling this objective can verify the rationale of implementing our approach of measuring functional size from textual requirements in practice.

Our approach: To address this objective, we first assigned a human expert to measure the functional sizes of different systems from their textual requirements. We identified these measurements as our gold-standard. We then implemented the automation of our text mining approaches, presented in Chapter 7, and extracted the FSM modeling artifacts automatically. Our implementation then calculated the functional size automatically based on the extracted FSM model. The brief details about our implementation are presented in Section 7.7.

We then ran our experiments to check if the automation of our text mining approaches measure the functional size within an acceptable margin of error and quicker than manually performed tasks of FSM over the same textual requirements.

Metrics: In the above experiments, we calculated the error in measurement through Mean Magnitude of Relative Error (MMRE). We also preset the acceptable margin of error in terms of MMRE.

The first research question, which we targeted by our above experiments in relation to our objective #6, is presented below, along our related null and alternative hypotheses:

Research Question, Q_9 : “Can the automation of our approaches measure functional size within the acceptable margin of error, when compared to the measurements of an expert?”

Null Hypothesis, $H_{9,0}$: The automation of our approaches measures functional size with errors higher than the acceptable margin of error, when compared to the measurements of an expert.

Alternative Hypothesis, $H_{9,a}$: The automation of our approaches measures functional size with errors equal to or lower than the acceptable margin of error, when compared to the measurements of an expert.

The next research question, which we targeted by our aforementioned experiments in relation to our objective #6, is presented below, along our related null and alternative hypotheses:

Research Question, Q_{10} : “Can the automation of our approaches measure functional size quicker than the time of an expert to measure functional size manually?”

Null Hypothesis, $H_{10,0}$: The automation of our approaches measures functional size slower than the time of an expert to measure functional size manually.

Alternative Hypothesis, $H_{10,a}$: The automation of our approaches measures functional size at the same time as or quicker than the time of an expert to measure functional size manually.

A detailed discussion on our experiments to validate the above hypotheses is presented in Section 7.8, where we also analyze our results.

Table 6 shows how all of our above research questions map to our research phases to address our research objectives.

	Objective #1	Objective #2	Objective #3	Objective #4	Objective #5	Objective #6
Phase I: FSM by Non-Experts	Research Question, Q_1	Research Questions, Q_2 & Q_3				
Phase II: Size Approximation			Research Question, Q_4			
Phase III: Requirements Classification				Research Question, Q_5		
Phase IV: FSM Model Extraction					Research Questions, Q_6 , Q_7 & Q_8	
Phase V: Evaluation of FSM Automation						Research Questions, Q_9 & Q_{10}

Table 6: Mapping of Research Questions over the Reseach Phases in Relation to the Objectives

Thus, to sum up our six research objectives that were targeted during the span of our five research phases, our overall aim with this research is to address the open problems mentioned in Section 1.2 by developing a comprehensive methodology for measuring functional size from textual software requirements written in unrestricted natural language, and, thus, facilitate early estimation of software development effort.

In the next section, we will be discussing our formalization of functional size measurement process and its approach of quantifying the size.

4.3 Formalization of FSM Model and Quantification

In this section, we describe how we formalized the conventional quantification process of functional size measurement (FSM). We related the conceptual artifacts of FSM to specific textual segments of software requirement by modeling an ontology. We then present the formulas, which not only can be applied over the instances of this model to calculate the numerical value of the functional size, but also fully complies with the standard process of FSM, described in (ISO/IEC 14143-1, 2007). Thus, this formalization can be implemented algorithmically to automate the computation of numerical value of the functional size of a software.

Extraction of functional size from software requirements documents require human experts to thoroughly understand the textual requirements, then follow a complex evaluation process and use experienced judgment to extract the conceptual artifacts that pertain to the functional size measurement standard. Various FSM standards prescribe different evaluation processes that often depend on subjective judgment of the experts in extracting the conceptual artifacts.

We therefore chose the latest iteration of the COSMIC FSM standard for our experiments that promises an objective measurement approach when well-decomposed and formalized functional requirements are used (COSMIC, 2014). However, the standard depends on experts' contribution in extracting its conceptual modeling artifacts, as discussed in Section 2.7.2. They are the COSMIC Data-movements and the COSMIC data-groups. The COSMIC data-movements are of four types: Entry, Exit, Read and Write.

Thus, as discussed in Section 2.7.2, the measurement process of COSMIC functional size requires identifying the presence of any or some of the four types of data-movements in one segment of functional requirement, and then mapping one or more data-groups to each of the data-movements. We therefore built an ontology (in OWS format), as shown in Figure 12, that relates the conceptual artifacts of COSMIC FSM into a formal model when instantiated from textual requirements. It also builds traceable mapping between the segments of a requirements document and the artifacts of an FSM model.

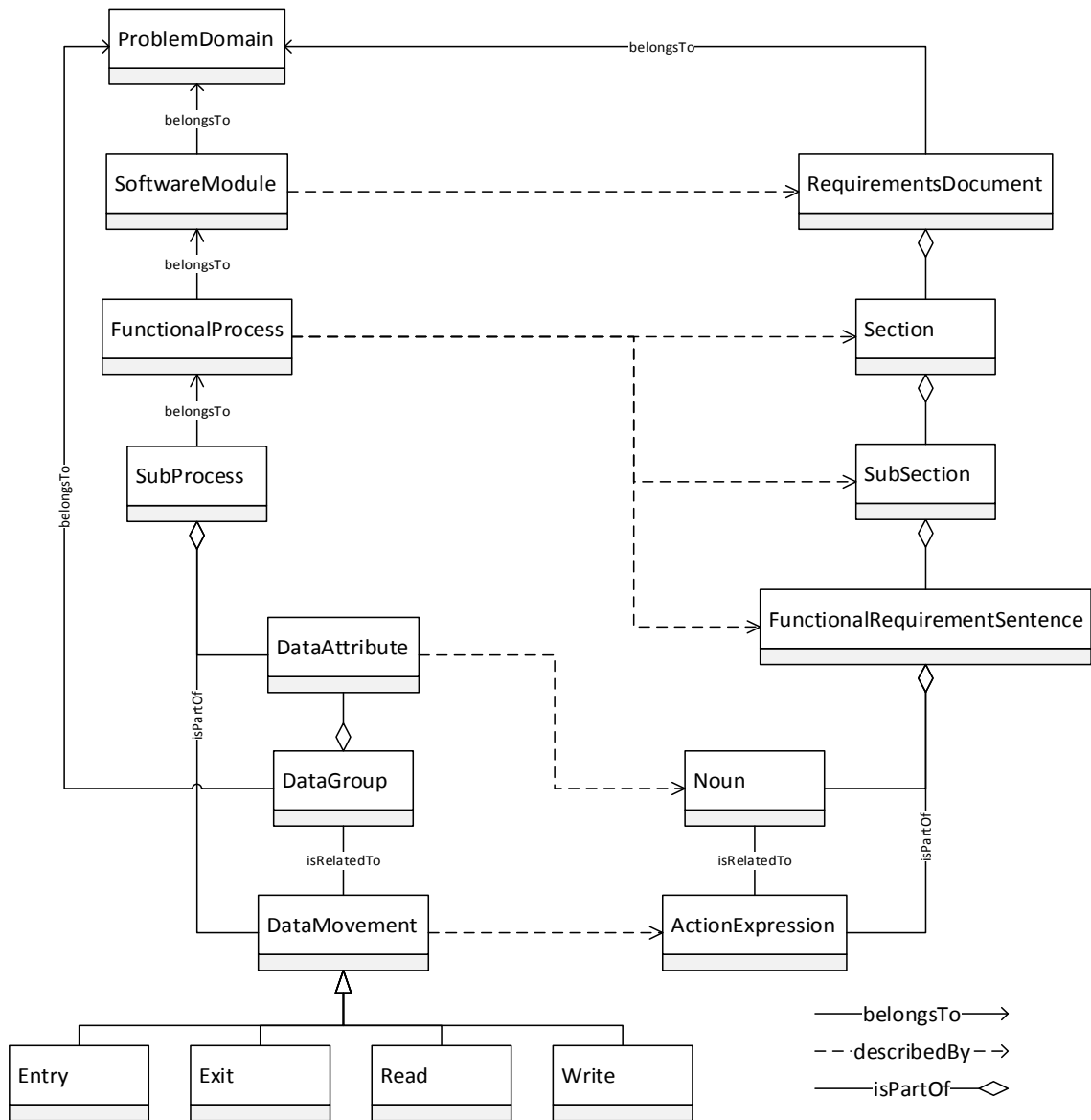


Figure 12: Ontology of COMIC FSM Model

We know that a numerical value of functional size is first assigned to each of the functional processes of a requirements document and then summed up to measure the size of the whole document, as discussed in Section 2.7.2. We formalize this counting process by first assuming that a requirements document, $DOCUMENT_p$, belonging to the problem domain p , contains the following set of n functional processes—

$$\{FPROC_1, FPROC_2, FPROC_3, \dots, FPROC_n\}$$

Thus, the functional size of the software as represented by $DOCUMENT_p$ is—

$$FunctionalSize(DOCUMENT_p) = \sum_{i=1}^n FunctionalSize(FPROC_i) \quad (1)$$

We also know, according to the COSMIC standard, that the set of all possible types of data-movements—

$$DM = \{Entry, Exit, Read, Write\}$$

And, let the set of all possible data-groups that can appear in the problem domain p be—

$$DG_p = \{data-group_1, data-group_2, \dots, data-group_m\}$$

—where $data-group_1, data-group_2, \dots, data-group_m$ are data-group names or domain entity names that belong to a specific problem domain x , as discussed in Section 5.9.3 of Chapter 5.

According to the COSMIC standard, whenever a data-group participates in any type of data-movement, we need to identify this incidence as one object that is later to be aggregated by functional process for counting its functional size. Thus, this object can be represented as a pair as follows:

$$(data-group\ name, data-movement\ type)$$

For our process of formalizing of the FSM counting process, we identify such pair as an *FSM object*.

Thus, the set of all possible FSM objects for the problem domain x is $\{(data-group_1, Entry), (data-group_1, Exit), (data-group_1, Read), (data-group_1, Write), (data-group_2, Entry), \dots, (data-group_m, Entry), (data-group_m, Exit), (data-group_m, Read), (data-group_m, Write)\}$.

This can also be written as—

$$DG_p \times DM$$

We define $FSMObjects(y)$ as a function that returns a set of our FSM objects that represent distinct types of data-movements of different data-groups that appear within the functional process, y .

Therefore, $FSMObjects(FPROC_i) \subset DG_p \times DM$

That is, our objective of FSM automation would to realize this function such that $FSMObjects(FPROC_i)$ would return the set of elements that are only *some* of the following pairs and appear within $FPROC_i$:

$(data-group_1, Entry), (data-group_1, Exit), (data-group_1, Read), (data-group_1, Write), (data-group_2, Entry), \dots (data-group_m, Entry), (data-group_m, Exit), (data-group_m, Read), (data-group_m, Write)$

Thus, due to the nature of sets (i.e. having no duplication of elements in a set leads to no duplication of our FSM objects counted within a functional process), the aggregated measure of functional size of $FPROC_i$ would simply be—

$$FunctionalSize(FPROC_i) = \|FSMObjects(FPROC_i)\| \quad (2)$$

And, therefore, by equations (1) and (2) —

$$FunctionalSize(DOCUMENT_p) = \sum_{i=1}^n \|FSMObjects(FPROC_i)\| \quad (3)$$

Here, in accordance with the COSMIC standard, the COSMIC functional size, measured in units of COSMIC Function Points (CFP), is equal to the sum of the frequencies of all these FSM objects that belong to each of the functional processes. Therefore, with such modeling approach, we can generate a traceable report to show the breakdown of the total CFP for each different type of data-movement and for each functional process, while linking each of these artifacts to its source segment of textual requirements.

Thus, in this section, our application of the ontology-based modeling techniques provided traceability links from the input textual requirements to the output functional size. We also described our method of formalizing the CFP counting process through our formulas that followed the COSMIC standard accordingly. We later used this idea into building our novel extension of the CFP counting process by defining the ranges of CFP automatically, as described in Section 7.8.

4.4 Linguistic Introspection of Size Measurement Analysis (LISMA)

In the introduction chapter of this thesis, the methodology of our solution for automated functional measurement was briefly introduced (see Section 1.3). We name this methodology as Linguistic Introspection of Size Measurement Analysis (LISMA), and our research discussed in this thesis evaluates feasibility of each of its steps with controlled experiments. Our plan for these experiments were distributed over several phases, which have been presented previously in Section 4.2.

In this section, the major intermediate steps of LISMA and their intermediate inputs and outputs are discussed. With LISMA, we integrated all of our linguistic analysis approaches used in the phases mentioned in the Section 4.2, to devise a comprehensive and innovative approach that takes textual software requirements written in unrestricted natural language as input and outputs the functional size of the software as reflected by the input requirements. It also outputs the corresponding functional size measurement (FSM) model and builds traceability links between the segments of the original textual requirements and the conceptual artifacts of the FSM model. The details of LISMA along with experiments that evaluate its feasibility are presented throughout the chapters of this thesis. Figure 13 briefly shows the intermediate tasks performed in LISMA.

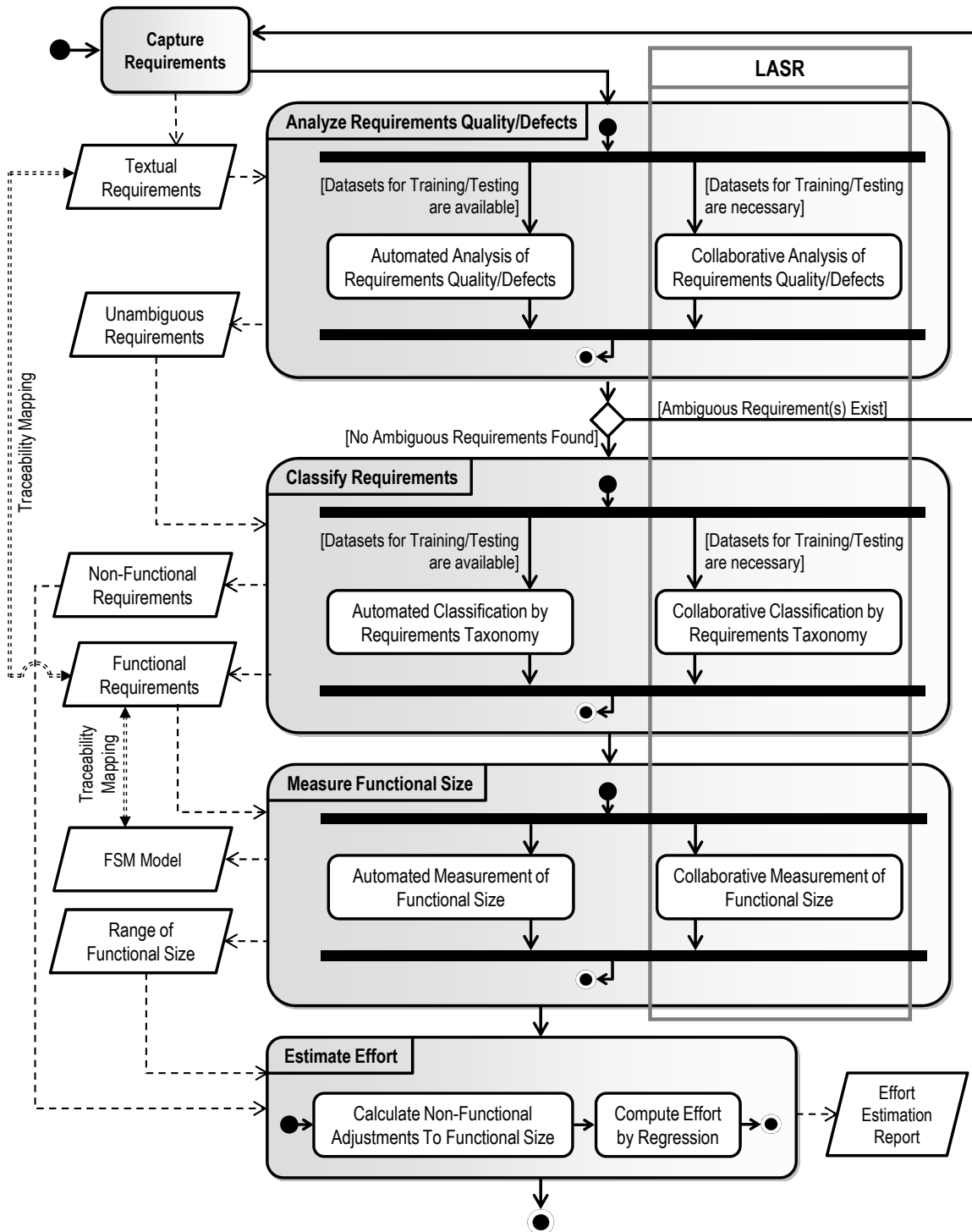


Figure 13: Inputs, Outputs and the Intermediate Steps of LISMA Supporting Early Effort Estimation

As shown in Figure 13, LISMA can be incorporated with the existing workflow of measuring functional size that is presented earlier in Figure 2 in Section 1.2, and introduces more details

in comparison. Thus, Figure 13, shows how LISMA starts by taking the captured textual requirements as input, passes the text through three intermediate steps and finishes by outputting the functional size for effort estimation and the corresponding Functional Size Measurement (FSM) model with traceability links. These three intermediate steps are:

- i.* Analysis of Requirements Quality/Defects
- ii.* Classification by Requirements Taxonomy
- iii.* Measurement of Functional Size

LISMA integrates the above three intermediate steps in such way that they can be executed both manually and automatically. LISMA allows both of our manual and automated approaches of these steps to be executed in parallel, independent of each other. The brief implementation details of our automated approaches for the above three steps are presented in Sections 4.4.1, 4.4.2 and 4.4.3 respectively. On the other hand, we designed all of the manual approaches related to the above three steps as collaborative annotation tasks, that are assisted by our uniquely designed annotation tool, called Live Annotation of Software Requirements (LASR), which is described in details in Chapter 5.

The software development effort estimation step, shown in Figure 13, is kept outside of the scope of LISMA. It reflects our innovative approach to effort estimation using functional size measurement that takes into account the impacts of different types of Non-Functional Requirements and different types of problem domains, published in (Abdukalykov, Hussain, Kassab, & Ormandjieva, 2011).

4.4.1 Analysis of Requirements Quality/Defects

To measure functional size from unrestricted textual requirements, a measurer first faces the challenge of cleaning up the text from all its ambiguities. Our manual approach of detecting ambiguities involve the task of manually annotating textual requirements, based on a quality model (Hussain, Ormandjieva, & Kosseim, 2007; Ormandjieva, Hussain, & Kosseim, 2007) that can discriminate between ambiguous and unambiguous requirements at lexical level. Our annotation tool, LASR, is equipped with text pre-processing modules that can extract

sentences from requirements documents and guide a human annotator to manually perform requirements annotation tasks.

However, the task of detecting ambiguity from text can be monotonous and error-prone for long requirements documents when done manually, resulting in the usage of poor quality requirements for functional size measurement, that eventually can contribute to poor quality results in the measurement.

To face this challenge, our previous work (Hussain, Ormandjieva, & Kosseim, 2007; Ormandjieva, Hussain, & Kosseim, 2007) showed that using a trained text-mining system we can successfully classify requirements text into ambiguous and unambiguous sentences. As requirements development process is iterative in nature (IEEE, 2004), such text mining system can aid the requirements analyst to extract the ambiguous sentences so that they can be elaborated upon or rephrased or dropped by the specifier and then reanalyzed by the system. Thus, the iteration continues until no ambiguous statements are left in the document, providing a better quality requirements document, ready for any FSM methods to be applied.

Thus, in this step, we use the Ambiguity Checker tool, described in (Hussain, Ormandjieva, & Kosseim, 2007; Ormandjieva, Hussain, & Kosseim, 2007) to facilitate a semi-automated environment to detect and resolve ambiguity in the textual requirements. The tool outputs a collection of sentences in the document as unambiguous, and the rest, classified as ambiguous, are fixed manually by the requirements specifier. Thus, the document moves to the next iteration resolving ambiguity on the way, and the process ends when all the sentences in the document are classified as Unambiguous. The accuracy of this classifier in detecting ambiguity, when 10-fold-Crossvalidation performed on 472 instances, was 88.56%, as recorded in our work presented in (Hussain, Ormandjieva, & Kosseim, 2007; Ormandjieva, Hussain, & Kosseim, 2007).

4.4.2 Classification by Requirements Taxonomy

The taxonomy of requirements categorizes requirements into many classes. However, since the functional size measurement methods work primarily on functional requirements (FR) only, the next challenge that a measurer faces is to extract the functional requirements

manually from the requirements document, where both functional and non-functional requirements can be mixed together in the same paragraphs. Again, the manual counterpart of our approach in LISMA prescribes this task to be performed as collaborative annotation over unambiguous textual requirements to classify them as either functional or non-functional requirements. Our annotation tool, LASR, as described in Chapter 5, supports executing and monitoring such requirements classification tasks over large repositories of textual requirements.

Moreover, our previous work (Hussain, Kosseim, & Ormandjieva, 2008) showed that, again, a text miner can effectively classify the requirements text into different classes of requirements taxonomy, e.g. functional and non-functional requirements, with a high accuracy of 98.56%, when 10-fold-Crossvalidation performed on 765 instances. It can therefore help the measurer extract the functional requirements automatically, so that the FSM methods can be applied.

Thus, in this step, we take the unambiguous textual requirements that we received from the previous step as input to this step, and we use the text miner, presented in (Hussain, Kosseim, & Ormandjieva,2008) to classify textual requirements into functional requirements (FR) and non-functional requirements (NFR). Thus, the set of FR sentences are extracted fully automatically and its performance are shown to attain a very high accuracy in the results. The NFR's extracted in this step are left to be used for further processing in the later on steps.

4.4.3 Measurement of Functional Size

Measuring Functional Size is the final and the most complex step of LISMA. Here, we use the unambiguous functional requirements, as extracted from the previous two steps, described in Sections 4.4.1 and 4.4.2 respectively; and, for our research, we choose to measure the functional size, in the units of COSMIC Function Points (CFP), that is entailed by these requirements. The overview of the approach for measuring functional size using the unambiguous functional requirements is illustrated in Figure 14.

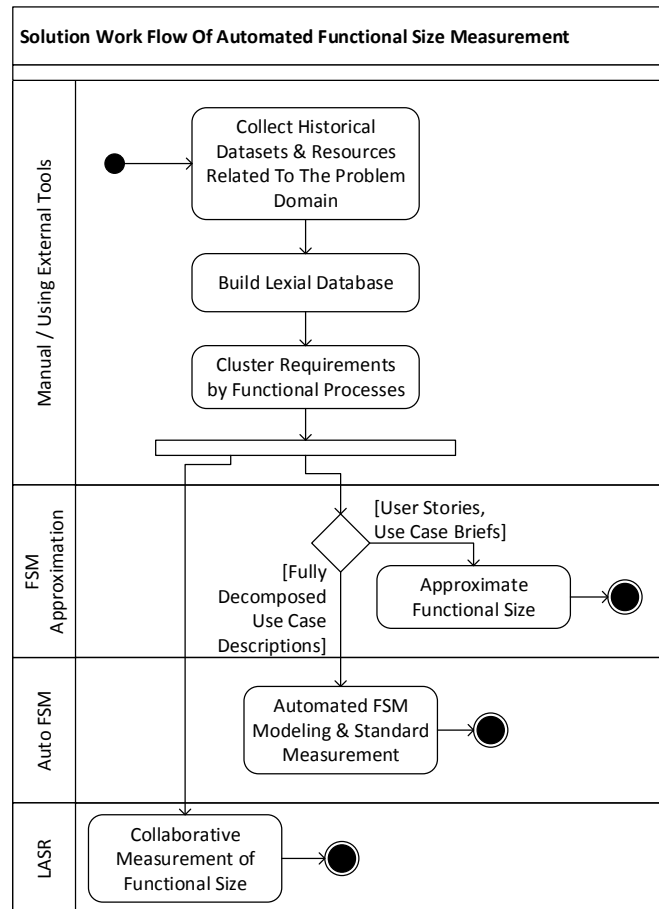


Figure 14: Our Approach for Measuring the Functional Size from Functional Requirements

As shown in Figure 14, our approach follows three initialization steps before starting to measure functional size. These steps:

(1) Collection of Archived Resources: Our approach first starts with collecting the historical datasets and textual requirements from the archived projects belong to a specific problem domain. We try to gather as much resources as possible to describe the problem domain, its common conceptual entity names, the actor names and the common verbs used to express the four types of data-movement.

(2) Building Lexical Database: After the collection of resources from archived projects, we start the process of building a lexical database for the specific problem domain. The database contains words and phrases of different categories. They are:

- i.* Domain Entity Names / Data-group Names

- ii. Attribute Names
- iii. Actor Names
- iv. Data-Movement Verbs
- v. Stative Verbs

The domain entity names or the data-group names are collected straight from the domain analysis of archived projects within the same problem domain (mostly by analyzing their static models, e.g. the domain model). Also, the collaborative environment of our annotation tool supports building the list of data-group names from the collective knowledge of the annotators, as shown in Section 5.2.1. We thus use both the approaches to build this vocabulary.

Then, our vocabulary of attributes contained words that can be used to represent members or properties or measurable attributes of any entity (i.e. data-group, for our work). We not only use the static models of our archived projects in this respect, but also manually selected a slice of nouns from WordNet (Miller, 1995; Princeton University, 2010) that belong to the super classes *Property*, *Relation* and *Communication* via their hypernym paths. We then manually modified some of the words to create new forms that may appear in textual requirements.

We select our vocabulary of actor names as mentions of people or their roles or their professions or their positions in an organization. We suggest including names of human actors to this vocabulary based on our historical knowledge of the archived projects from the same problem domains.

The data-movement verbs commonly appear to express the action of different types of data-movements and are not specific to any problem domain. These verbs are further grouped into four sets: *Entry Verbs*, *Exit Verbs*, *Read Verbs* and *Write Verbs*, based on the type of data movements they participate in.

We develop the vocabulary of *Stative Verbs* that are mostly used to describe the states of objects instead of describing actions over them. The usage of these verbs is also not specific to any problem domain. These verbs in most cases describe the state being or having or

spatial relations amongst objects. We again used WordNet (Miller, 1995; Princeton University, 2010) to manually extract our vocabulary of stative verbs.

Figure 15 shows our approach of building the lexical database. We first use the available resources from the archived projects to determine seed words for each of the aforesaid categories. We then expand the vocabulary of each category by using the synsets of WordNet (Miller, 1995; Princeton University, 2010) and adding new synonyms that are common to each pair of the seeded words in that category until no new synonym exists.

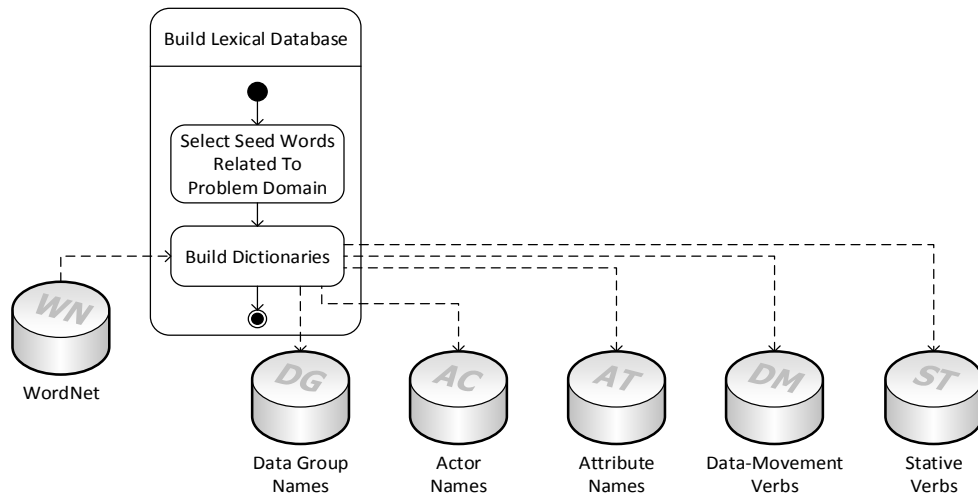


Figure 15: Our Approach for Building The Lexical Databases

These databases are later on used by our linguistic feature extractor to extract different lexical features from textual requirements for our automated FSM approach that implements text mining process (discussed in Section 7.4.1). However, these databases do not need to be a complete collection of names and verbs for a problem domain, as the text miners used in this research relies on many other syntactic features too (presented in Section 7.4.2) that also helps to generalize its classification approach across different less-familiar problem domains.

All the lexical databases that we used during our experiments are presented in Appendix A.

(3) Clustering of Textual Requirements: The COSMIC measurement standard (ISO/IEC 19761, 2011; COSMIC, 2014) prescribes to measure the size of a software by pieces, each of which is called a “functional process”. The COSMIC standard defines the functional process

as a set of data-movements that can stand alone to represent an individual functionality of a software and is triggered by an external event. Thus, a functional process corresponds to the use cases of the software requirements document (Jenner, 2011; Condori-Fernández, Abrahão, & Pastor, 2007). However, the functional requirement sentences, within a requirements document, may not appear together in groups that correspond to individual use cases or functional processes.

Thus, the FSM process in COSMIC starts by grouping these requirements sentences manually, where each group corresponds to one functional process. Our approach of measuring functional size in LISMA assumes the groups of functional requirements indicating the functional processes to have already been identified, either manually or by using an unsupervised text clustering technique as developed by our research group (Moradi-Seresht, Ormandjieva, & Sabra, 2008) for the Requirements Engineering Assistance Diagnostic (READ) project. This text clusterer can automatically group sentences in a requirements document into sets of sentences that correspond to individual use cases. Since COSMIC functional processes are analogous to use cases (Jenner, 2011; Condori-Fernández, Abrahão, & Pastor, 2007), our approach recommends using such text clustering technique, in case the sentences of functional requirements are not already tagged to indicate the functional processes that they belong to.

After finishing all of the above initialization steps, LISMA in parallel follows on with a manual process and an automatic process to measure the functional size, as earlier shown in Figure 14. The manual process allows human annotators of different expertise to collaboratively measure the functional size following a unique approach. On the other hand, the automatic process can either approximate the functional size, if the textual requirements to measure are not formally written (e.g. user stories); or it measures the functional size precisely by extracting the functional size measurement (FSM) model, if the textual requirements are written formally (e.g. fully-dressed use cases).

4.5 Conclusion

This chapter explained the details of the phases of our research methodology. It then introduced our formalization of the COSMIC functional size measurement process and its quantification by modeling an ontology. We then briefly described the steps of our methodology for automating functional size measurement.

The next chapter (Chapter 5) investigates the feasibility of performing the FSM activities with non-experts. It describes the details of the functional size measurement by manual annotation with non-experts and discusses how a linguistic annotation tool can effectively aid the complex tasks of functional size measurement from software requirements.

Chapter 5

Functional Size Measurement By Non-Experts

“The measure of a man is what he does with power.”
— *Plato*

5.1 Introduction

Manual measurement of functional size by analyzing textual requirements conventionally requires human experts for the task. Thus, it can be a very costly activity to perform in large-scale software projects or research work by engaging the human experts in time-intensive manipulation of textual requirements.

To address these issues, Phase I of this research investigates the possible means of reducing functional size measurement cost through requirements annotation work performed by non-experts. The related research objectives, research questions and our hypothesis have been presented in Section 4.2.2, while our tools, and the design and the results of our experiments are discussed in this chapter.

Our research in this phase led us to develop an annotation tool, called “LASR” (Hussain, Ormandjieva, & Kosseim, 2012), that assists software measurers and requirements analysts with collecting annotation data by supporting different types of requirements annotation tasks. It combines not only the common features of annotation tools (as presented in Section 3.6), but also some unique additional features, that are presented in this chapter in Section 5.2.2, making it specifically suitable for performing functional size measurement. The chapter also includes a brief discussion on LASR’s design and implementation.

LASR helps building large pools of annotated corpora for statistical data collection and improves the overall process of FSM by attaining more accurate annotations with less time spent by the annotators. Figure 16 presents a brief workflow diagram showing how LASR works.

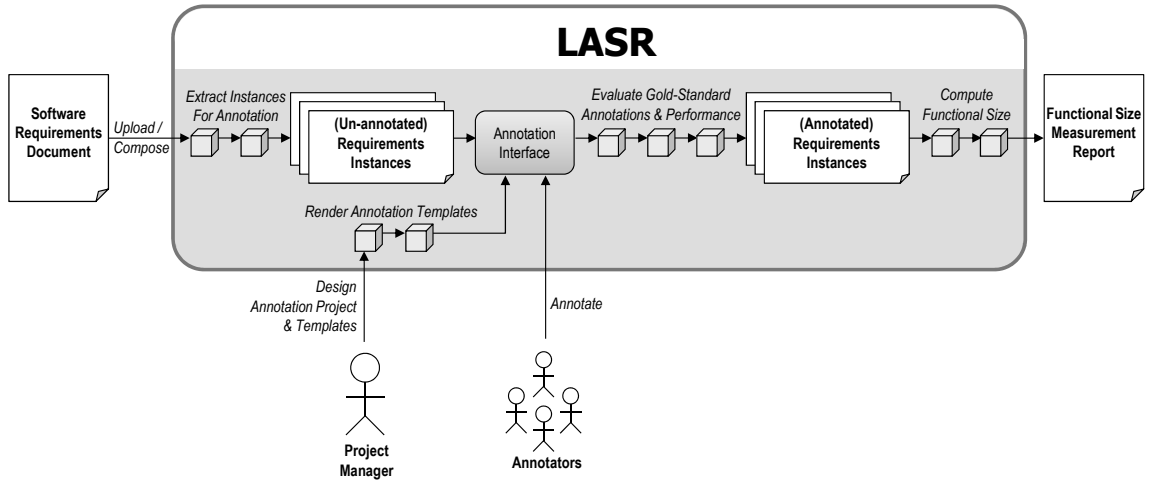


Figure 16: An Overview Workflow Diagram of LASR

As shown briefly in Figure 16, LASR not only aids the annotation tasks related to functional size measurement, but also provides a comprehensive environment that helps to pre-process and extract requirements instances, design and administer annotation projects, determine gold-standard annotations and compute the functional size measurement reports automatically. Here, in this chapter, we test the effectiveness of these additional features of LASR by executing different annotation experiments in relation to our research questions Q_1 , Q_2 and Q_3 , as presented in Section 4.2.2.

The chapter is organized as follows: we present brief details on the annotation interface and the unique features of our annotation tool LASR in Section 5.2, describe the overview of our annotation experiments in Section 5.3, present the details of our experiments and the analysis of their results in Sections 5.4 to 5.7, discuss the implementation details of LASR briefly in Sections 5.8, 5.9 and 5.10, and, finally, add our concluding remarks in Section 5.11.

5.2 LASR: Live Annotation of Software Requirements

As the available annotation tools did not provide the features required for our work, we developed LASR (Live Annotation of Software Requirements). LASR aids the collection of annotated corpora and the generation of training/testing datasets required by the supervised learning systems related to our work (Hussain, Ormandjieva, & Kosseim, 2009). It can be customized for use with both private and public annotation projects. All data of the private software projects can be kept protected from unauthorized access. Thus, software organizations can use LASR to manage their requirements documents, setup any kind of annotation tasks and share their data securely with designated users only.

5.2.1 Annotation Interface for Functional Size Measurement

LASR provides a rich graphical user interface that allows quick navigation and control during the annotation tasks. The frontend interface is implemented using PHP and jQuery libraries, making it dependant only on the JavaScript engine of an internet browser at the client-side. This makes LASR accessible from a wide-range of client-side platforms (e.g. PCs, smartphones, tablets etc.).

It also allows customization of its user interface for annotation via editable XML-based templates to support any types of requirements annotation tasks, that includes the annotation tasks related to functional size measurement as well. For example, Figure 17 shows its annotation interface for COSMIC functional size measurement.

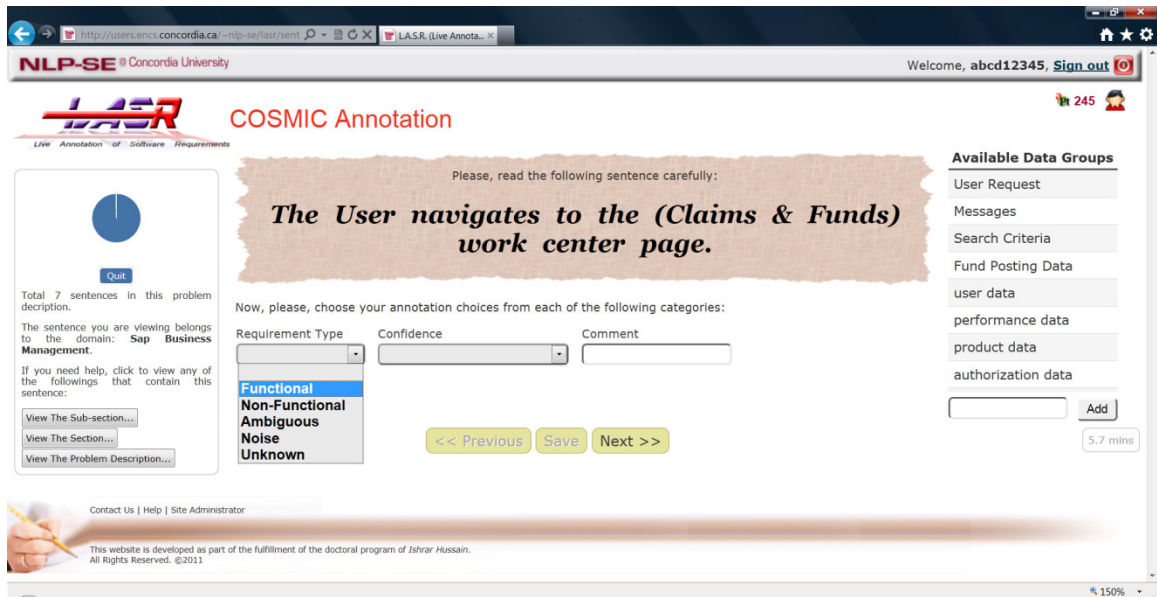


Figure 17: First Screenshot of LASR's Customized Annotation Interface for COSMIC Annotation.

Figure 17 shows that the customized annotation interface of LASR includes several key attributes that help to complete the COSMIC annotation tasks successfully. We describe these attributes of the annotation interface in details below:

(1) View of the requirement instance

As shown in Figure 17, the requirement instance to be annotated (a requirement sentence, in this example) is presented by default in the middle of LASR's annotation interface with comparatively larger typeface in an attempt to draw the focus of the annotators to the instance and allowing them to read it with ease. Here, in this example, the annotation interface shows one requirement sentence at a time to the annotator. Thus, hiding the context of the requirement sentence by default allows an annotator to consider one sentence at a time for annotation, minimizing the risk for an annotator to skip the annotation tasks necessary for a sentence due to losing his/her focus to its context.

(2) Navigation between requirement instances

There are also three navigation buttons at the bottom of the annotation interface, shown in Figure 17. Here, the left and the right buttons allow an annotator to navigate to the previous or the next requirements sentence respectively for performing new annotations, or reviewing and re-annotating his/her previous annotations. LASR auto-saves all the annotation work

during navigation. The middle button allows the annotator to save the annotation work that he/she performed so far on the currently viewing requirements sentence. This realizes the feature of LASR that allows an annotator to start, stop and resume the annotation tasks at a time of his/her convenience. The feature helps to eliminate the *fatigue effects* (Galesic & Bosnjak, 2009) of the annotators during large-scale requirements annotation work.

(3) View of the summary of the task

The left pane of the annotation interface, as shown in Figure 17, presents by default a quick summary of the annotation tasks including a graphical representation of the portions of the tasks completed versus the portions of that is remaining in terms of a pie chart. It also allows the annotators to take a quick glance at the related context of the requirement sentence, if necessary.

(4) View of the related data-groups

In Figure 17, we also find that the customized annotation interface is configured to include a widget at the right pane that collaboratively builds a list of the names (or, *dictionary*) of the related domain concepts (i.e. *data-groups*, in COSMIC) belonging to a specific problem domain. This widget allows the annotators to participate collaboratively in building this domain-based dictionary of data-groups, and thus, describe the problem domain to LASR as they encounter new data-groups by reading new requirements instances from the problem domain during the annotation tasks. Here, an annotator can view the collection of the data-group names that have been added so far to the specific domain-based dictionary by all the annotators participating in these annotation tasks. In case the annotator encounters a new domain concept, i.e. a new data-group name, while reading the current requirements instance he/she can add it to the dictionary using this widget.

(5) View of the elapsed time

LASR records the time spent by an annotator in completing the annotation tasks for each of the instances. This recorded time represents all the time spent over the viewing an instance and entering the annotations, including the time spent later on the same instance, if the annotator chooses to come back and re-annotate the instance. Figure 17 shows that the annotation interface at its bottom right corner includes a real-time display of this time with a

much smaller typeface in an attempt to be not too distracting for the annotators. The interface allows this display of the elapsed time, counting up in real-time, to be turned on optionally by the annotator to persuade him/herself to annotate faster. The annotator can also turn off the display anytime if it becomes too distracting during the annotation work.

(6) Annotation input fields

As shown in Figure 17, a set of input fields appear below the requirement sentence containing the choices of annotation. Here, following the process of functional size measurement in COSMIC, the customized annotation interface first asks the annotators to classify each requirements sentence into any of the four categories:

- i.* Functional Requirement
- ii.* Non-Functional Requirement
- iii.* Ambiguous Requirement
- iv.* Noise

The interface also requires the annotators to indicate their levels of confidence⁵ to their annotations. The annotators can also choose to add comments to their annotations.

If an annotator chooses to annotate a requirement sentence as a “Functional Requirements”, the customized annotation interface displays additional input fields that allow the annotator to annotate the base noun-phrases of the sentence to indicate their roles as COSMIC artifacts (e.g. data-attributes, data-groups, and data-movements), as shown in Figure 18. Thus, the annotator needs to check if any of these base noun-phrases represent any *data-attribute* that participate in COSMIC *data-movement*, based on what is stated in the respective functional requirement sentence. If so, the annotator annotates these noun-phrases as data-attributes by indicating what *data-groups* they belong to and what kinds of *data-movements* they participate in.

⁵ The level of confidence is a four-valued attribute in LASR that is associated with each annotation label submitted by an annotator. Its discrete values are {0.1, 0.4, 0.7, 1.0}. The annotation interface shows these values nominally as “I have no idea about this”, “Maybe, I’m not sure”, “Most likely”, and “Highly Confident” respectively. The value 0 is attached to all those annotation labels, which are not submitted by the annotator.

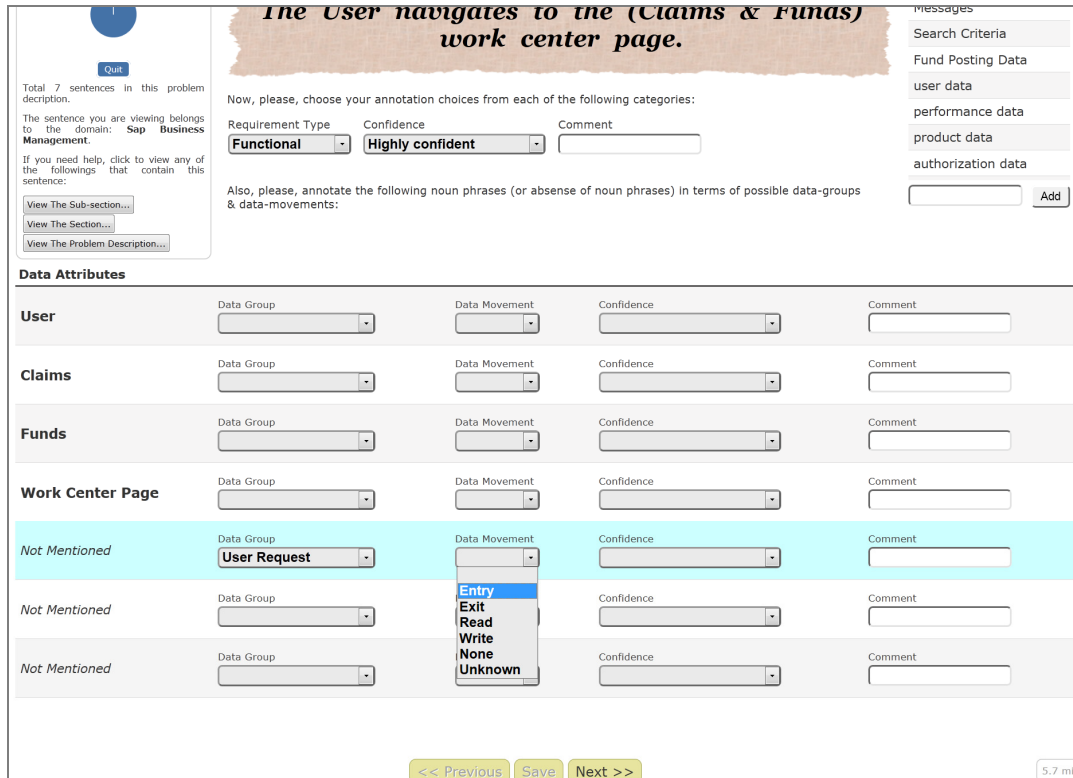


Figure 18: Second Screenshot of LASR's Customized Annotation Interface for COSMIC Annotation

5.2.2 Additional Features of LASR

In addition to the features discussed in Section 3.6, LASR includes a set of unconventional features that make it a unique annotation tool in comparison to the contemporary linguistic annotation tools, presented in Section 3.5. Practical usage of an annotation tool for the tasks functional size measurement demands some of these features as services to be available. Thus, these additional features are targeted to improve the overall experience of using an annotation tool for the functional size measurement tasks. We discuss these additional sets of features in the following sections.

(1) Features For Usability

We chose a unique set of features to be additionally included in LASR with an aim to improve its usability and support the tasks of functional size measurement. They are:

The Annotation Interface

The fully customizable annotation interface of LASR allows using some unique web-widgets that are especially designed to enhance the annotation experience during functional size measurement tasks. As discussed in Sections 1.2 and 2.7.2, functional size measurement activities in practice demands large scale annotation work that are costly in terms of the amount of time spent by the annotators during annotation. With an objective to reduce this annotation time, we unconventionally designed LASR's default annotation interface for functional size measurement, which is discussed in Section 2.7. For example, LASR's annotation interface by default shows only the instance to be annotated at one time, hiding its surrounding context, in an attempt of holding the focus of the annotator solely on the instance to annotate. Thus, the interface also predefines the scope of the instance to be annotated and relieves the annotator from the task of defining the scope of each requirements instance. Our target with this feature is not only to reduce the annotation time and effort, but also to minimize the chances of annotation errors and, thus, achieve a higher degree of agreement among the annotators during complex requirements annotation tasks, e.g. functional size measurement.

Collaborative Acquisition of Domain Knowledge

The web-based interface of LASR allows collaborative participation of the annotators, during the requirements annotation process, to build domain-based dictionaries containing probable annotation labels for *data-group* annotation. The feature of building these domain-based dictionaries is realized in LASR by a special widget that is presented earlier in Section 5.2.1. This feature is specifically added to support the requirements annotation tasks related to COSMIC functional size measurement, and, to our knowledge, no other annotation tool to date offers a similar feature. Thus, LASR holds a separate dictionary for each problem domain and allows the annotation of data-groups to be performed with ease, reducing the effort of an annotator to inspect a large set of requirements every time to identify the name of one data-group. Also, the feature imposes restrictions on the choices and scopes of data-group annotation tasks resulting in less annotation errors and higher agreement among the annotators.

(2) Feature For Efficiency

Here we list the features that are additionally included in LASR with the aim of attaining more accurate annotations over software requirements from non-experts.

When using an annotation tool for the tasks of functional size measurement, the accuracy of the measurements relies on the correctness of the gold-standard annotation label chosen for each of the requirements instances. Now, if non-experts are involved with the annotation tasks, their levels of skill in functional size measurement affects the correctness of their annotations. Thus, annotation tools may automatically compute wrong labels as gold-standard annotations by normalizing the annotations of the non-experts. This phenomenon is resolved in practice by avoiding automatic computation of gold-standard, and by manually reviewing and adjudicating the annotations collectively in group meetings to select the correct gold-standard annotation label for each instance. However, the process can be extremely costly and time-consuming for large scale requirements annotation tasks that are related to functional size measurement.

Thus, LASR includes the features of tracking the levels of skill of the annotators and their levels confidence in choosing the annotation labels. This allows LASR to logically infer gold-standard annotations for each instance following different statistical measures to weight the annotations of the non-experts based on their levels of skill and confidence. It, thus, eliminates the need of manual processing the annotation data to maintain the correctness of gold-standard annotations, and can generate annotated corpora automatically for training/testing different text mining systems that are intended for use in different areas of requirements engineering and functional size measurement. To address this, LASR includes a feature to track the levels of skill of the annotators and their levels of confidence when choosing the annotation labels. This allows LASR to logically infer gold-standard annotations for each instance following different statistical measures to weigh the annotations of the non-experts based on their levels of skill and confidence. It, thus, eliminates the need of manual processing the annotation data to maintain the correctness of gold-standard annotations, and can generate annotated corpora automatically for training/testing different text mining systems that are intended for use in different areas of requirements engineering.

LASR provides two different options to compute the gold-standard annotations automatically. They are:

- i.* Using Annotators' Confidence Level Only
- ii.* Using Annotators' Confidence and Skill Levels

We describe them below.

i. Using Annotators' Confidence Level Only

The first option LASR introduces is to compute the gold-standard annotations automatically using the level of confidence entered by the annotators. Here, LASR tries to compute the gold-standard annotation for each of the annotated instances, by first assigning a custom score to each of the annotation labels based on the level of confidence submitted by the annotators. Thus, the annotation label with the highest score, and that is also greater than 0.5⁶, is selected as the gold-standard annotation. LASR uses the formula below to calculate the probability score. It shows that if m_i annotators have annotated an instance i , and the class c is one of the possible class labels for annotating the instance i , then the probability score of class c for the instance i is —

$$Score(c | i) = \frac{1}{m_i} \times \sum_{x=1}^{m_i} (conf_{x,c,i}) \quad (1)$$

Thus, the final probability score of an annotation class c for an instance, i , is the arithmetic average of all the $conf_{x,c,i}$ values submitted by each annotator, x . The confidence of an annotator x , denoted by $conf_{x,c,i}$ in the formula (1), is equal to one of values of {0.1, 0.4, 0.7, 1.0} that is according to the level of confidence chosen by the annotator x , while annotating the instance i as class c . And, for all those classes, c' , that are not chosen the annotator x for the instance i , $conf_{x,c',i}$ will be equal to 0. Thus, $0 \leq conf_{x,c,i} \leq 1$.

⁶ LASR requires the annotators to attach fuzzy levels of their confidence to each of their annotations. The level of annotator's confidence is collected as a 4-value ordinal nominal variable, instead of a continuous numerical variable. The 4-value ratings are then translated into fuzzy numeric values that are chosen as positive real numbers ≤ 1 , all having equal intervals, and none being 0.5 or 0. This is because we wanted no annotation to be ignored because of a zero weight or be indecisive because of a 0.5 weight on the confidence level.

ii. Using Annotators' Confidence and Skill Levels

LASR also provides another option to compute gold-standard annotation automatically. This option requires the expert, who designs an annotation work, to seed the annotation data beforehand by setting true gold-standard annotations for at least a small portion of the unannotated corpora. LASR recommends seeding by annotating at least 10%⁷ of the unannotated instances randomly. The seeded annotations are regarded as the true gold-standard annotations; and LASR then measures the skill level, S_x , of each annotator, x , as the probability of his/her annotations agreeing with the true gold-standard annotations. For example, if the annotations of an annotator, x , agrees with the portion of the true gold-standard annotations seeded by the expert 60% of the times, then the skill level, S_x , measured by LASR would be 0.6. Thus, LASR uses a modified version of the formula (1) as below to calculate the score for selecting the gold-standard annotation labels for each instance—

$$Score(c | i) = \sum_{x=1}^{m_i} (conf_{x,c,i} \times S_x) \quad (2)$$

Thus, the annotation label c that achieves the highest score in terms of the formula (2) is selected as the gold-standard annotation for an instance i .

Unlike the previous option of using annotator's confidence level only, this option is dependent on an expert seeding the annotation data by annotating with a small portion of the unannotated corpus. Thus, to make the process of expert seeding optional, LASR provides both of these options to compute gold-standard automatically. Our experiment#2a presented in Section 5.5 compared the effectiveness of both features.

5.3 Experiments Overview

In this section, we will present the details of our experiments following the reporting guidelines of (Wohlin, Runeson, Höst, Ohlsson, Regnell, & Wesslén, 2012; Jedlitschka, Ciolkowski, & Pfahl, 2008). We will specifically discuss the evaluation of our research questions, Q_1 , Q_2 and Q_3 , and hypotheses $H_{1,0}$, $H_{1,a}$, $H_{2,0}$, $H_{2,a}$, $H_{3,0}$ and $H_{3,a}$, presented

⁷ In our experiment, for example, we used seeding by annotating 11.67% of the un-annotated instances randomly that yielded good results, as presented in 5.5.2.

earlier in Section 4.2.2. We carried out a number of controlled annotation experiments in order to validate our design choices of LASR's features, especially its features for efficiency, as presented in Section 5.2.2. In relation to our research questions, Q_1 , Q_2 and Q_3 , we had non-experts in our experiments performing annotation tasks using LASR over textual software requirements (similar to the task shown in Section 2.7.2) and compared its accuracy to that obtained with manual annotation, without any tool support.

5.3.1 Participants: The Annotators

Two groups of annotators and one expert (in the field of software requirements engineering and functional size measurement) participated for each case in the experiments. A total of two different experts worked on six different cases — one on two academic cases and the other on the remaining four industrial cases. The experts were graduate students, who were experts in the original problem domain and were actively involved with the development of these projects represented by the cases. They also had a year of measurement experience.

The first group of annotators (NE_{ft} ⁸), consisting of four people, all graduate students of the Master of Computer Science program, were trained for requirements annotation and functional size measurement. They could distinguish between functional, non-functional, and ambiguous requirements and sentences in requirements documents that are none of those, or “noise”. They can also measure functional size from a given set of textual requirements. All annotators first participated in preliminary tests to perform requirements annotation on test documents. The expert led the two-week-long training of the annotators, and also participated with them in all of our annotation experiments.

The other group (NE_{mt} ⁹) consisted of 26 people, all third-year students of the undergraduate software engineering program. They were introduced to the COSMIC standard (through a one-hour lecture and reference books), but were not thoroughly trained. Before the experiments were performed, no prior tests were conducted to verify their knowledge. However, they were all trained via a one-hour-long tutorial to work on LASR's annotation interface for requirements annotation.

⁸ NE_{ft} is the name for the group of “*Non-Experts, with full training*”

⁹ NE_{mt} is the name for the group of “*Non-Experts, with minimal training*”

5.3.2 Experimental Materials

The Corpus

Our corpus (sample set of requirements documents) used in these experiments was composed of six requirements documents belonging to six different software projects respectively. Four of these projects are of the same problem domain, while the remaining two are from different problem domains. The requirements documents have been collected from both the industry and academia. Some of these documents were complete, while others were “change requirements” describing only small required modifications to an existing system. Some statistics about these documents are presented in Table 7.

Doc. ID	Doc. Title	Source	Problem Domain	Extracted After Pre-processing	
				Total Sentences	Total Noun-Phrases
C1	<i>(undisclosed)</i>	SAP Labs, Montreal, Canada	Business	91	314
C2	<i>(undisclosed)</i>	SAP Labs, Montreal, Canada	Business	15	59
C3	Course Registration System	Concordia University	Academic (Private)	179	711
C4	IEEE Montreal Website	Concordia University	Web (Public)	467	1318
C5	<i>(undisclosed)</i>	SAP Labs, Montreal, Canada	Business	7	27
C6	<i>(undisclosed)</i>	SAP Labs, Montreal, Canada	Business	106	383

Table 7: Documents in the Corpus, Used in the Experiments

Table 7 shows the number of sentences extracted after pre-processing only those sections of the requirements documents that held textual user requirements.

Setups for Manual Annotation

All the sentences in our corpus were used to populate a spreadsheet in Microsoft Excel that served as the interface for the manual annotation task of our experiment. A macro-script in Excel guided the annotators through our experiment and also measured in the background the time spent by each annotators during their respective annotation tasks.

Setups for Annotation on LASR

A new annotation project was created in LASR by compiling the annotation templates in XML and loading our corpus. The annotators who were part of this experiment were already registered in LASR and were then assigned to their respective annotation tasks via LASR.

5.3.3 Experiment Execution

As presented in Section 5.3.1, we had two different groups of non-expert annotators, NE_{ft} and NE_{mt} , where we purposefully chose NE_{ft} to be more advantaged than NE_{mt} . Here, NE_{ft} received full training by the expert (with multiple practice sessions) to perform requirements annotation. On the other hand, NE_{mt} , our other non-expert group of annotators, received only minimal training for requirements annotation.

We designed our experiments, so that the fully-trained non-expert annotators of NE_{ft} annotated the requirements documents of our corpus manually, without the support of LASR, and the minimally-trained non-expert annotators from NE_{mt} use LASR to annotate the same documents. Also, we had our only expert annotate all of these documents manually. In the analyses of our experiments, we compared the performances of each of the non-expert groups NE_{ft} and NE_{mt} to that of the expert. The tasks performed by all our annotators during these experiments are summarized in Table 8.

Doc ID	Total Sent. To Annotate	Total Noun-Phrases To Annotate	Annotators Participating via LASR	Annotators Participating Manually	
			Minimally Trained Non-Experts (NE_{mt})	Fully Trained Non-Experts (NE_{ft})	Expert
C1	91	314	14		1
C2	15	59	14		1
C3	179	711	14	4	1
C4	467	1318	12	4	1
C5	7	27	12	2	1
C6	106	383	14	4	1

Table 8: Task of the Annotators

It should be mentioned that all the annotators of the groups did not participate in annotating sentences from all the documents. This was consciously done to reduce the average workload of annotators, and also due to some of their absences (e.g. the document, C5, was manually annotated by the expert and only two annotators from NE_{ft} , instead of four; and the documents, C1 and C2, were annotated by the expert only).

Simple Annotation Experiments over Sentences

In this set of experiments, the annotators performed simple sentence-level requirements annotation, where they had to annotate sentences into one of the following four classes:

- i.*** Functional Requirement
- ii.*** Non-Functional Requirement
- iii.*** Ambiguous Requirement
- iv.*** Noise

A total of 858 sentences from six documents were annotated using this experiment. The details of these experiments and along with the results and their analyses are presented in (Hussain, Ormandjieva, & Kosseim, 2012), where it successfully demonstrated that LASR annotation interface, as presented in Section 5.2.1, along with its other features, help multiple non-expert annotators to attain a high degree of inter-annotator agreement with an expert for the tasks of annotating software requirements.

Complex Annotation Experiments over Noun Phrases

These set of experiments aim to explore the answers to our research questions related to functional size measurement by manual annotation of non-experts, as presented in Section 4.2.2. In these experiments, our annotators performed relatively complex requirements annotation tasks at the noun-phrase-level, related to COSMIC functional size measurement.

A total of 2812 base noun-phrases that were extracted from the functional requirement sentences of all of the six documents, were annotated during this experiment. To relate with our targeted research question, we selected this requirements annotation work that is complex and is comprised of three different annotation tasks, i.e. to identify:

- i.*** if a base noun-phrase is a data-attribute or not;
- ii.*** what the data-group, which a data-attribute belongs to, is called; and
- iii.*** what kind of data-movement a data-attribute participates in.

In the following paragraphs, we will be analyzing the results of the above annotation tasks that our annotators performed during this experiment. In this section, we refer to the tasks—

(i) as “Data-Attribute Annotation” or *DA*, (ii) as “Data-Group Annotation” or *DG*, and (iii) as “Data-Movement Annotation” or *DM*—respectively.

For the data-attribute annotation (DA) task, the expert annotated 314 noun-phrases out of 2812 as data-attributes that participate in data-movements. And, for the data-group annotation (DG) task, the expert identified a total of 22 different data-groups from the requirements documents that belong to the three different problem domains, as presented in Section 5.3.2.

The data-movement annotation (DM) task involves annotating these data-attributes into one or more types of data-movements that they participate in. It is a multi-class annotation task where a data-attribute can be annotated in all of the following four ways:

- i.* As a data-attribute that participate in an *Entry* data-movement, or as one that does not do so;
- ii.* As a data-attribute that participate in an *Exit* data-movement, or as one that does not do so;
- iii.* As a data-attribute that participate in a *Read* data-movement, or as one that does not do so; and
- iv.* As a data-attribute that participate in a *Write* data-movement, or as one that does not do so.

Therefore, for the data-movement annotation (DM) task, the expert annotated each of the 742 data-attributes into one or more of the above four types of data-movements that they participate in based on the description presented by their source requirements. We identified the expert's annotation as the true gold-standard to compare all other annotations made during this experiment, both manually and by LASR. The distribution of the true gold-standard annotations for data-attributes and data-movements, as annotated by the expert, is shown in Figure 19.

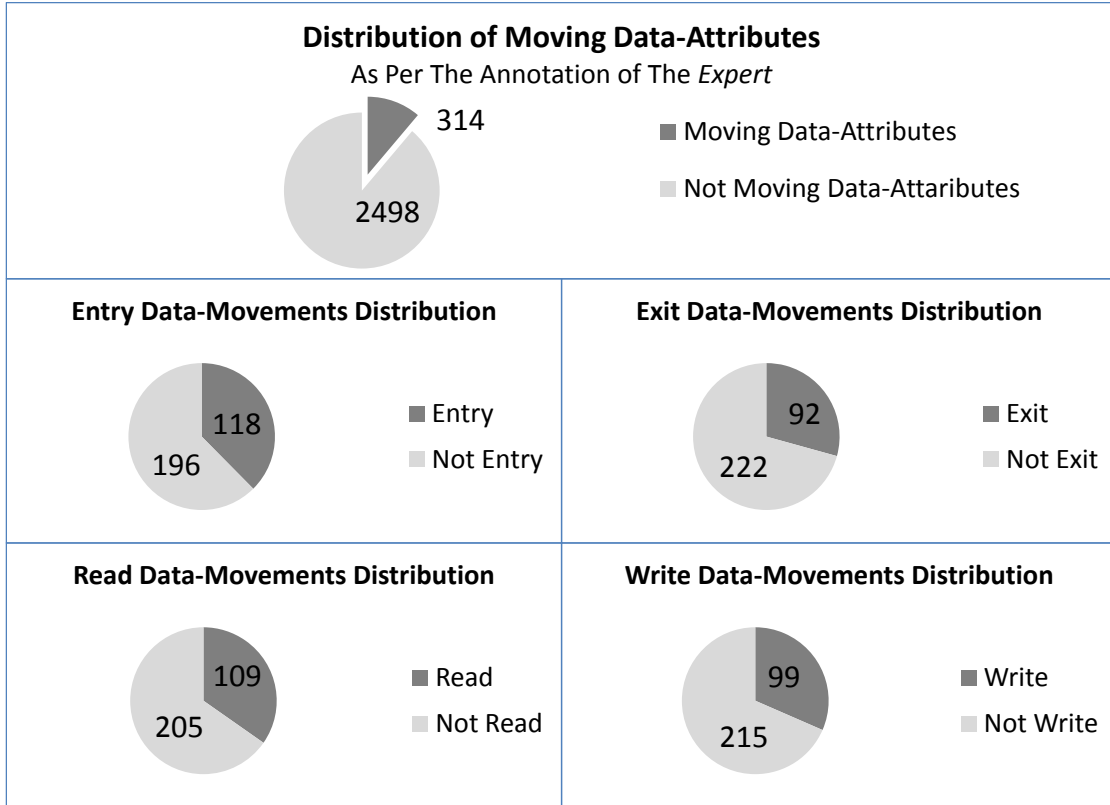


Figure 19: Distribution of the True Gold-Standards (As Annotated by the Expert) in Our Corpus

5.4 Experiment #1: Expert vs. Multiple Non-Experts

In this section, we present the details of our experiment #1 that addresses our research question Q_I .

5.4.1 Hypotheses and Variables

Firstly, we formalize our hypothesis, the related variables and metrics. As presented in Section 4.2.2, our research question, Q_I , and our corresponding null hypothesis $H_{I,0}$, are as follows:

Research Question, Q_I : “Can well-trained non-experts attain at least a moderate level of agreement with the expert for annotation tasks related to FSM?”

Null Hypothesis, $H_{I,0}$: Well-trained non-experts can never attain at least a moderate level of agreement with the expert for annotation tasks related to FSM.

Thus, we formalized our null hypothesis $H_{1,0}$ as follows:

$$H_{0,1} = AGREEMENT(NE_{ft}, E) < KAPPA_{moderate} \quad (1)$$

—where,

$AGREEMENT(A, E)$ = Average of the pair-wise inter-annotator agreement [measured in Cohen’s Kappa(Cohen, 1960)] between each annotator of group A and an expert annotator E.

NE_{ft} = Group of non-expert annotators, fully-trained (i.e. they participated in our two-week-long training)

E = An Expert Annotator (i.e. he/she has more than a year of FSM experience)

$KAPPA_{moderate}$ = The constant value of Kappa for moderate level of agreement, as determined by the evaluation scale of (Landis & Koch, 1977).

Therefore, based on our null hypothesis, $H_{1,0}$, we now form an alternative hypothesis $H_{1,a}$ as follows:

$$H_{0,1} = AGREEMENT(NE_{ft}, E) \geq KAPPA_{moderate} \quad (2)$$

Here, the *Independent Variables* are the “types of trained annotators (based on expertise)”, which are of nominal scale with binary values $\{ NE_{ft}, E \}$. Additionally, the *Dependent Variable* here is the “average of the pair-wise inter-annotator agreements”, which is of ratio scale with values $[0,1]$. We measure this average of the pair-wise inter-annotator agreements, $AGREEMENT(A, E)$, as the mean of the degrees of inter-annotator agreement between sets of annotation labels chosen by each annotator of group A for some given set of instances, and the set of the annotation labels chosen by an expert for the same set of instances. This can shown in the equation below:

$$AGREEMENT(A, E) = \frac{1}{n} \times \sum_{j=1}^n IA[a_j(I), E(I)] \quad (3)$$

—where,

$AGREEMENT(A, E)$ = Average of the pair-wise inter-annotator agreement [measured in Cohen’s Kappa(Cohen, 1960)] between each annotator of group A and an expert annotator E .

A = a group of n annotators $\{a_1, a_2, a_3, \dots, a_n\}$

n = Total number of annotators in group A

E = An Expert Annotator (i.e. he/she has more than a year of FSM experience)

$IA[P, Q]$ = The degree of inter-annotator agreement between two sets of annotation labels, P and Q .

$E(I)$ = A set of annotation labels chosen by an expert for a given set of instances, I

$a_j(I)$ = A set of annotation labels chosen by an annotator a_j for a given set of instances, I

Here, we used Cohen’s Kappa (Cohen, 1960) to measure the degree of inter-annotator agreement, $IA[P, Q]$, between two sets of annotation labels, e.g. P and Q .

5.4.2 Results and Analysis

In this section, we discuss the results of our experiment #1. As shown earlier in Table 8, the fully-trained non-expert annotators of NE_{ft} performed annotation manually on four documents only, instead of the six, for a total of 2425 noun-phrases. They used Microsoft Excel to manually annotate these noun-phrases. A macro-script on Excel measured the time spent by the annotators in the background. The distribution of the pair-wise agreement in terms of Cohen’s Kappa (Cohen, 1960) of all the annotation labels chosen by each of the four annotators of the NE_{ft} group with the true gold-standard labels chosen by the expert is shown in Figure 20.

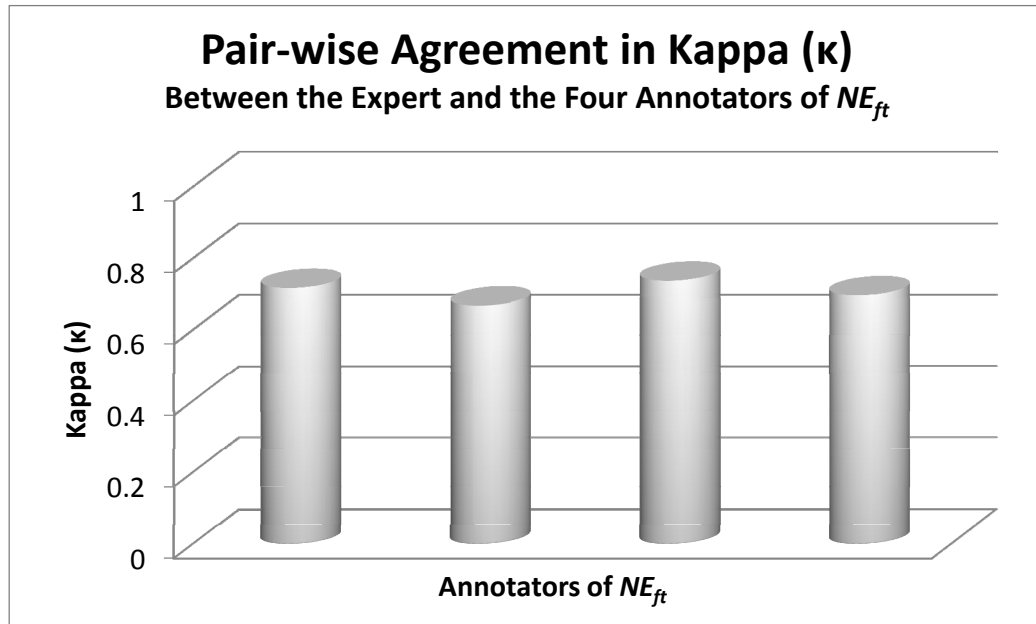


Figure 20: Pair-wise Agreements between the Four Annotators of NE_{ft} and the Expert

As shown in Figure 20, the average of the pair-wise agreements of all four annotators of NE_{ft} with the expert for all annotations is 0.7023, which indicates a moderate level of agreements according to the Kappa evaluation scale of (Landis & Koch, 1977).

The gold-standard annotations were computed by following the majority voting model. Whenever gold-standard annotation could not be resolved, adjudication was performed by the participation of all the annotators of NE_{ft} .

The distribution of the gold-standard annotations that resulted from the manual annotation work by the non-expert annotators of NE_{ft} is shown in Figure 21.

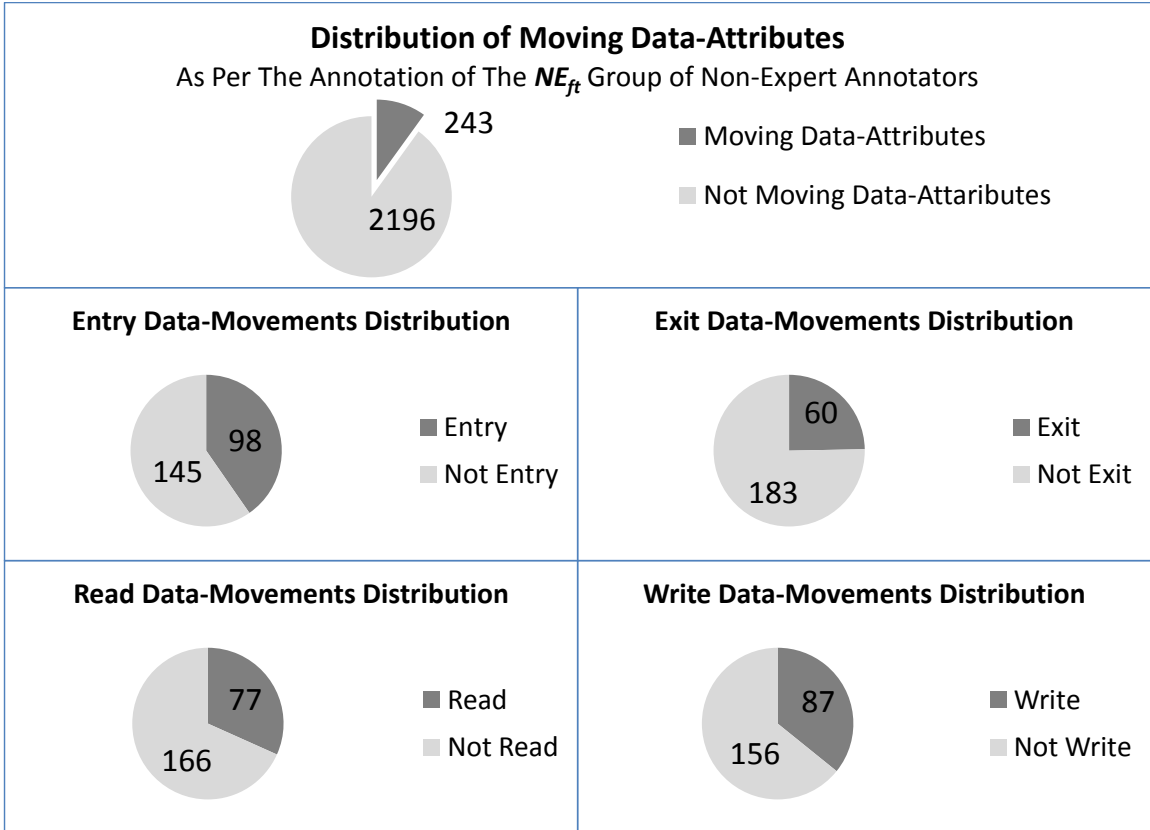


Figure 21: Distribution of the Gold-standard Annotations (As Annotated by NE_{ft})

We found a high degree of inter-annotator agreement in terms of Cohen’s Kappa (Cohen, 1960) between these gold-standard labels and the true gold-standard annotations, which are set by the expert. For example, the confusion matrix comparing these gold-standard labels to the true gold-standard annotations is shown in Table 9. The gold-standard annotations resulted from the manual annotation of the non-expert annotators of NE_{ft} show a very high degree of inter-annotator agreement, based on Cohen’s Kappa (Cohen, 1960), (i.e. Kappa = 0.9748), with the annotations of the expert (the true-gold-standards). This result however represents the best-case scenario, where all four annotators of NE_{ft} were properly trained and the gold-standard annotations were computed after holding meticulous adjudication sessions with their participation to resolve their points of disagreements.

		Annotated by NE_{ft}	
		Data-Attribute	Not Data-Attribute
Annotated by Expert	Data-Attribute	237	5
	Not Data-Attribute	6	2191

Table 9: Confusion Matrix for the Moving Data-Attribute Annotation by NE_{ft}

Thus, our results support our alternative hypothesis $H_{1,1}$ to be true, rendering our null hypothesis $H_{0,1}$ to be false. This satisfies our research objective#1 showing that the process of functional size measurement can be executed effectively with non-experts through requirements annotation.

5.5 Experiment #2a: Annotation Accuracy using LASR

We performed two experiments, #2a and #2b, that address our research question Q_2 . In this section, we present the details of our experiment #2a.

5.5.1 Hypotheses and Variables

Firstly, we formalize our hypothesis, the related variables and metrics. As presented in Section 4.2.2, our research question, Q_2 , and its corresponding null hypothesis $H_{2,0}$, are as follows:

Research Question, Q_2 : “Which type of FSM-related manual annotation tasks performed by non-experts attains a higher accuracy: the manual annotation task performed by well-trained non-experts, or the LASR-aided annotation task performed by minimally trained non-experts?”

Null Hypothesis, $H_{2,0}$: The FSM-related manual annotation tasks performed without LASR, but by well-trained non-experts and with disagreements resolved through the adjudication session, always attain a higher accuracy than the LASR-aided manual annotation tasks performed by minimally trained non-experts and with no adjudication process.

Thus, we formalized our null hypothesis $H_{2,0}$ as follows:

$$H_{2,0} = ACC(MA, NE_{ft}) > ACC(LA, NE_{mt}) \quad (4)$$

—where,

$ACC(X, Y)$ = The accuracy of X type of requirements annotation task, performed by Y type of annotators.

MA = Manual annotation task

LA = LASR-aided annotation task

NE_{ft} = Group of non-expert annotators, fully-trained (i.e. they participated in our two-week-long training)

NE_{mt} = Group of non-expert annotators, minimally-trained (i.e. they participated in an hour-long training)

Therefore, based on our null hypothesis, $H_{2,0}$, we now form an alternative hypothesis $H_{2,a}$ as follows:

$$H_{2,a} = ACC(MA, NE_{ft}) \leq ACC(LA, NE_{mt}) \quad (5)$$

Here, the *Independent Variables* are “type of annotation task” and “type of non-expert annotator (based on training level)”, both of which are of nominal scale with values $\{MA, LA\}$ and $\{NE_{ft}, NE_{mt}\}$ respectively. Additionally, the *Dependent Variable* is the “accuracy of annotation”, which is of ratio scale with values $[0,1]$.

For the experiment #2a, we measure this accuracy of annotation, $ACC(X,Y)$, as the degree of inter-annotator agreement between a set of gold-standard annotation labels generated from X type of annotation task, which is performed by Y type of annotators over some set of instances, and the set of the annotation labels chosen by an expert for the same set of instances. This is shown in the equation below:

$$ACC(X, Y) = IA[GS_X(I, Y), E(I)] \quad (6)$$

—where,

$ACC(X, Y)$ = The accuracy of X type of requirements annotation task, performed by Y type of annotators.

$IA[P, Q]$ = The degree of inter-annotator agreement between two sets of annotation labels, P and Q .

$G_X(I, Y)$ = A set of gold-standard annotation labels generated from X type of annotation task, which is performed by Y type of annotators over a given set of instances, I

$E(I)$ = A set of annotation labels chosen by an expert for a given set of instances, I

Here, we used Cohen’s Kappa(Cohen, 1960) to measure the degree of inter-annotator agreement, $IA[P, Q]$, between two sets of annotation labels, e.g. P and Q .

5.5.2 Results and Analysis

Here, we discuss the results of experiment #2. The 26 minimally-trained non-expert annotators of NE_{mt} performed annotation on all the six documents for a total of 2408 noun-phrases. They used LASR to annotate these documents, and LASR, by default, kept account of the time they spent during annotation.

Before testing out the feature of LASR that auto-computes the gold-standard annotations (discussed in Section 5.2.2), we compute the gold-standard annotations manually using the simple majority voting rule (we refer to this method as MV). We find that the computed gold-standard annotations this way agree quite highly with those submitted by the expert. For example, the Kappa measure for the data-attribute annotations (DA) here was 0.91376. However, we find the method, MV , could not also resolve the gold-standard annotations for the 263 of the noun-phrases, indicating high degree of disagreements for those instances. A typical solution to resolve this issue would be to run adjudication sessions for all the 263 instances with the participation of all the 26 annotators. It indicates that the process can, therefore, be very costly with real annotation tasks performed over larger corpora.

To address this problem, we first test out the feature of LASR that computes the gold-standard annotations automatically using the level of confidence entered by the annotators, as presented in Section 5.2.2. Here, we name this method of computing gold-standard annotations with LASR as *CL* for our experiment.

When using LASR to compute the gold-standard annotation labels according to *CL*, the computed gold-standard annotations demonstrate an even higher agreement with those submitted by the expert. For example, the Kappa measure for the data-attribute annotations (DA) here was 0.94881. However, there still remain 89 instances for which the gold-standard annotations could not be resolved.

We then applied the next option computing gold-standard with LASR, as presented in Section 5.2.2, that uses the annotators' levels of confidence, along with their levels of skill together. As described earlier in Section 5.2.2, LASR measures the annotators' levels of skill automatically during the execution of the annotation tasks by using seeded annotations from the expert for 100 randomly chosen instances (i.e. 11.67% of the size of our corpus).

We name this method of computing gold-standard annotations with LASR as *CL+SL* for our experiment. Thus, we finally used LASR to compute the gold-standard annotation labels automatically according to *CL+SL*. Their final distribution is shown in Figure 22.

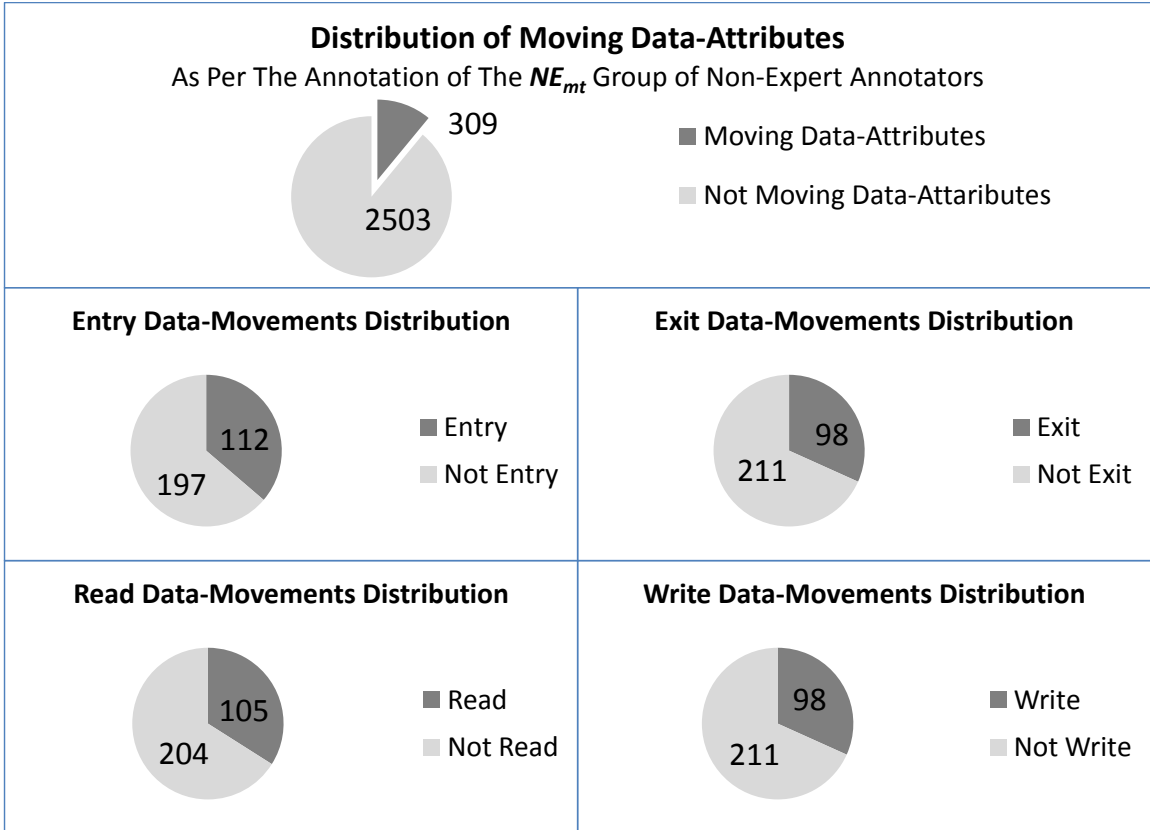


Figure 22: Distribution of the Gold-Standard Annotations (As Annotated by NE_{mt})

The computed gold-standard annotations now have the highest degree of inter-annotator agreement with those submitted by the expert. For example, the Kappa measure for the data-attribute annotations (DA) here was 0.98014. Moreover, there remained no instances, where their gold-standard annotation labels are unresolved, eliminating the need for conducting costly adjudication sessions.

The confusion matrices presented in Table 10 and Table 11 show the detailed results of this experiment.

		Annotated by NE_{mt}	
		Data-Attribute	Not Data-Attribute
Annotated by Expert	Data-Attribute	306	8
	Not Data-Attribute	3	2495

Table 10: Confusion Matrix for Data-Attribute Annotation by NE_{mt}

		Annotated by NE_{mt}	
		Entry	Not Entry
Annotated by Expert	Entry	109	9
	Not Entry	3	2691

		Annotated by NE_{mt}	
		Read	Not Read
Annotated by Expert	Read	103	6
	Not Read	2	2701

		Annotated by NE_{mt}	
		Exit	Not Exit
Annotated by Expert	Exit	91	1
	Not Exit	7	2713

		Annotated by NE_{mt}	
		Write	Not Write
Annotated by Expert	Write	94	5
	Not Write	4	2709

Table 11: Confusion Matrices for Data-Movement Annotation by NE_{mt}

Figure 23 summarizes the results of the sentence annotation experiments, showing the quality of the computed gold-standard annotations (based on the annotations submitted by NE_{mt}) for MV , CL and $CL+SL$, in terms of their degrees of agreement (in Kappa) with the true gold-standard annotations chosen by the expert. It also compares these results to that of the gold-standard annotations after NE_{ft} performed the task manually.

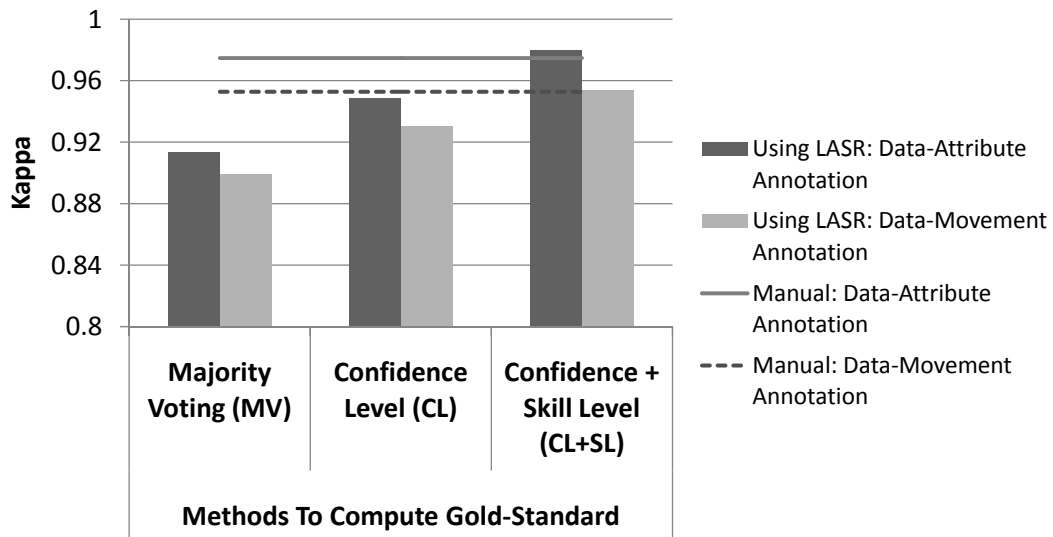


Figure 23: Quality of the Different Gold-Standard Annotations in Terms of Their Agreements (in Kappa) with the True Gold-Standard Annotations.

Here, it should be noted that the 26 annotators of the group NE_{mt} were not as well-trained for the COSMIC annotation tasks, as the group NE_{ft} . Their average level of skill were comparatively much lower than that of the trained annotators of NE_{ft} (e.g. the average degree of the pair-wise agreement between the annotations of each annotator of NE_{mt} and the true gold-standard annotations of the expert, in Kappa, was only 0.57392). However, our experiment with $CL+SL$ shows that LASR was able to weight the probability scores of formula (2) accordingly, based on the levels of skill of the annotators of NE_{mt} , and, thus, selected the gold-standard annotation labels that agreed the most with the true gold-standard annotations, as shown in Figure 23. This indicates that LASR can automatically extract gold-standard annotations that can be reliable enough, even when the group of annotators are not fully trained.

It should also be noted here that LASR's annotation interface allowed minimally-trained non-experts of NE_{mt} to always attain at least moderate level of agreement with the expert's true gold-standard annotations. This partially indicates the effectiveness of the additional usability features of LASR, listed in Section 5.2.2.

Thus, our results support our alternative hypothesis $H_{2,a}$ to be true, rendering our null hypothesis $H_{2,\theta}$, as presented earlier in Section 5.5.1, to be false. This satisfies our research objective#2, presented in Section 4.2.2, and indicates that LASR can automatically extract gold-standard annotations that can be reliable enough, even when the group of annotators are not fully trained.

5.6 Experiment #2b: Size Measurement Accuracy with LASR

Similarly to our previous experiment, #2a, this experiment, #2b, also addresses our research question Q_2 . In this section, we present the details of this experiment.

Here, our experiment uses the same formalization of our hypothesis, the related variables and metrics, as discussed in Section 5.5.1. Thus, our null and alternative hypotheses for this experiment are same as equation (4) and (5), as presented in Section 5.5.1.

However, for this experiment #2b, we define, $ACC(X,Y)$, as the “accuracy of functional size measurement”, which we measured in terms of Mean Magnitude of Relative Error or MMRE. In this experiment, we used the same materials (the corpus and the participants) that were used in our previous experiment #2a, and investigate if LASR aided the minimally-trained group of non-expert annotators (NE_{mt}) to accurately measure COSMIC functional size.

During the tasks of data-attribute annotation (DA) and data-group identification (DG) in experiment #1, the expert has annotated 314 noun-phrases out of 2812 as data-attributes. He also identified 10 unique data-group labels (i.e. entity names) in the “Business” problem domain, 3 in “Academic”, and 9 in “Web”, and associated the data-attributes with the data-group labels. We identify them as the true gold-standard annotations for the tasks DA and DG .

Table 12 summarizes what the other annotators of group NE_{ft} and NE_{mt} extracted in comparison for the same tasks.

Doc. ID	Noun-Phrases	Problem Domain	Manually Annotated				Annotated Using LASR	
			Expert Annotator		Annotators of NE_{ft}		Annotators of NE_{mt}	
			Data Attrib.	Data Groups	Data Attrib.	Data Groups	Data Attrib.	Data Groups
C1	314	Business	64	10	-	6 ¹⁰	63	11
C2	59		8		-		7	
C5	27		4		4		4	
C6	383		96		97		95	
C3	711	Academic	68	3	67	4	67	3
C4	1317	Web	74	9	75	11	73	9
Total	2812		314	22	243	21	309	23

Table 12: Frequency of Data-Attributes & Data-groups

By associating the annotated data-attributes with the extracted data-groups, we aggregate multiple movements of data-attributes into one data-movement. Thus, it reduces the total number of data-movement annotations applied over the data-attributes during our experiment #2a, and rightly follows the COSMIC standard to measure the functional size by counting the movement of the associated data-groups.

¹⁰ The annotators of group NE_{ft} did not annotate two of the four documents, belonging to the “Business” problem domain (as mentioned in section 5.3.1); hence, they could identify only 16 data-groups from the rest of the documents.

Table 13 shows the new aggregated frequencies of the four types of data-movements- Entry (E), Exit (X), Read (R) and Write (W) -that are extracted based on the gold-standard annotations of our previous experiment #2a. These are the actual COSMIC data-movements, the total count of which equals to the COSMIC functional size, measured by the unit *CFP* (COSMIC Function Point).

Doc. ID	Manually Annotated								Annotated Using LASR			
	Expert Annotator				Annotators of NE_{ft}				Annotators of NE_{mt}			
	E	X	R	W	E	X	R	W	E	X	R	W
C1	19	26	15	5					18	23	17	5
C2	7	6	2	1					7	6	3	2
C3	26	25	14	15	27	26	16	16	25	22	15	15
C4	55	43	12	23	51	39	13	22	54	41	13	22
C5	4	4	2	0	4	4	3	0	4	4	2	0
C6	16	16	9	8	15	12	10	7	15	13	9	8

Table 13: Aggregated Frequencies of Data-Movements

Thus, using the results from Table 13, we can compute the total CFP values and the magnitude of relative error (MRE) of all the documents in our corpus, as measured during the experiment #2a by the expert and our annotators of the groups NE_{ft} and NE_{mt} . These results and the mean magnitude of relative errors for both the groups are shown in Table 14.

Doc. ID	Total CFP				MRE	
	Expert	Annotators of		Annotators of		
		NE_{ft}	NE_{mt}	NE_{ft}	NE_{mt}	
C1	65		63		0.031	
C2	16		18		0.125	
C3	80	85	77	0.062	0.037	
C4	133	125	130	0.060	0.022	
C5	10	11	10	0.100	0	
C6	49	44	45	0.102	0.082	
MMRE =				0.081	0.049	

Table 14: Total Measured CFP and MMRE Results

Considering the annotations of the expert as the true gold-standard, we find that the results of Table 13 and Table 14 indicate that LASR helped the minimally-trained non-expert annotators of the group NE_{mt} to achieve near accurate results (MMRE¹¹ = 0.049) in measuring the functional size in terms of COSMIC FSM, even though the annotators were not as well-trained as the annotators of NE_{ft} . The results of Table 12 also show that using the annotation interface of LASR helped to limit the number of data-group labels during the DG annotation task for the annotators of NE_{mt} , while the annotators of NE_{ft} working manually introduced more unnecessary data-group labels during the same task.

Thus, the experiment successfully investigated our research question #2, presented in Sections 4.2.2 and 5.5.1, by demonstrating that LASR's efficiency feature, mentioned in Section 5.2.2, can help multiple non-experts to attain accurate measurements of COSMIC functional size.

5.7 Experiment #3: Record of Annotation Time

Finally, experiment #3 aims to explore the answer to our research question #3, presented in Section 4.2.2. In this experiment, we used the time data collected during our experiments #2a and #2b to investigate if LASR's usability features, presented in Section 5.2.2, can reduce the time spent in annotation.

We measured the time spent by each annotator from both groups, NE_{ft} and NE_{mt} , to perform the annotation tasks over each sentence, where they first annotated the sentence as Functional, Non-Functional, Ambiguous or Noise. Then, if an annotator annotated a sentence as a Functional requirement, he/she then had to perform the data-attribute (DA), data-group (DG) and data-movement (DM) annotation tasks (as mentioned in Section 5.5) for each of the noun-phrases that belonged to the sentence. In case of annotating by LASR, the annotators of NE_{mt} also had to include their levels of confidence for each of their annotations.

We found that to complete all these tasks manually, an annotator from the group NE_{ft} spent on average 2.441 minutes per sentence. On the other hand, to do the same annotation task on LASR over the same set of sentences, an annotator from the group NE_{mt} spent on average

¹¹ MMRE stands for Mean Magnitude of Relative Error.

1.769 minutes per sentence. This shows that LASR can help quicken the annotation tasks related to COSMIC FSM by a large extent, especially when annotating large sets of corpora. Thus, our experiment showed that LASR helped minimally-trained non-expert annotators to complete the complex annotation tasks related to COSMIC functional size measurement faster than fully-trained non-expert annotators, while retaining comparable accuracy.

5.8 Design & Implementation of LASR

We followed a highly modular approach when implementing LASR's architecture. LASR uses a client-server architecture at the highest-level of the logical view. On the server-side, it implements a three-tier-architecture, comprising of the Presentation, Application and Services layers. The application layer then further implements the model-view-controller architecture, via the CakePHP framework (Cake Software Foundation, 2014). Figure 24 shows LASR's architecture in details.

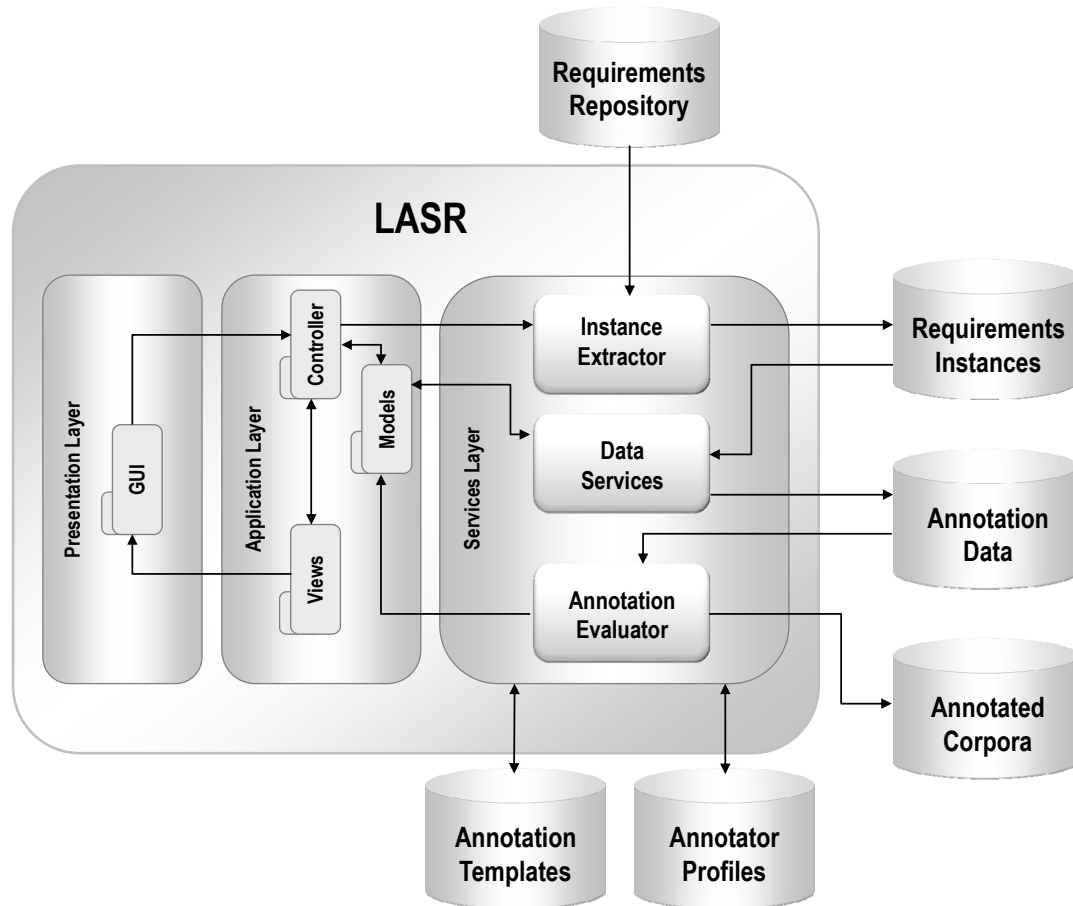


Figure 24: Architecture of LASR

Here, the *Requirements Repository* at the backend holds the requirements documents contributed by its users. The *Instance Extractor* module of LASR is equipped with lightweight NLP-based tools, e.g. a sentence delimiter and a noun or noun compound chunker, that can automatically extract requirements instances at the levels of passages, sentences and noun instances from the requirements documents and save them to the backend. *Annotation Templates* define the annotation work to be performed at a particular level of requirements instance (e.g. at the passage level, or sentence level or base noun-phrase level). The templates are stored as XML files at the backend file-system, and contains configuration details on the annotation interface as well, making the interface customizable by the curator.

5.9 Domain Model of LASR

LASR addresses the complex domain of managing requirements annotation projects, where annotation can be performed at various levels (document, section, sub-section, sentence and noun-phrase). It integrates concepts related to different fields, e.g. software project management, software requirements, measurement and linguistic annotation, to interact with each other. Figure 25 presents the visualization of the domain, via the UML domain model, showing different domain entities and their relationships, as realized by LASR.

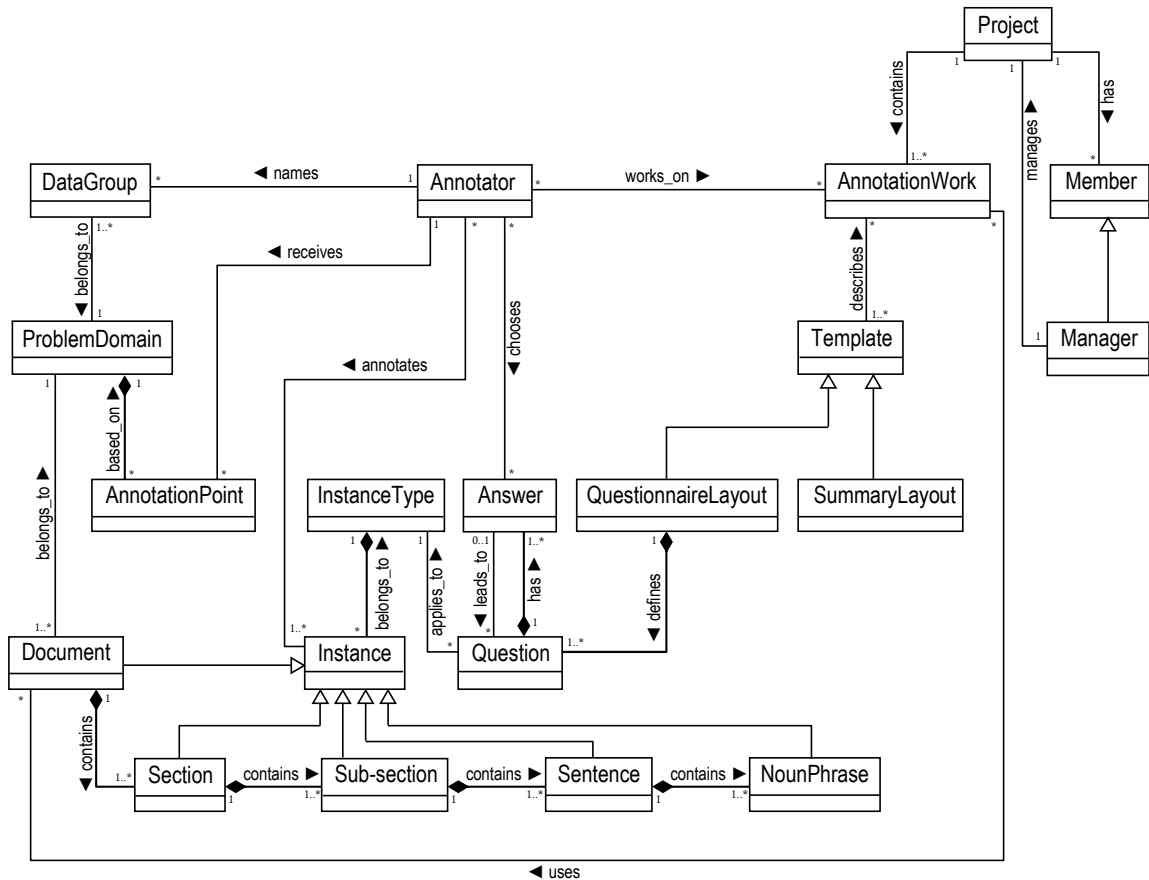


Figure 25: Domain Model of LASR

Some of the important conceptual entities of LASR, as shown in Figure 25, are described in the following.

5.9.1 Project

LASR identifies a *Project* as a concept analogous to a software project, coordinated by a team of people. A project in LASR is managed by one special user, called *Manager*, and contains a group of special users, called *Members*, who accepts/rejects the requests of Annotators to contribute to a project. These user roles are discussed in Section 5.10.

5.9.2 Annotation Work

The *Annotation Work* is the most important concept of LASR. A project in LASR is comprised of several annotation tasks that address different types of requirements analysis tasks. The manager defines an annotation work by associating the requirements documents to it and designing its interface via the *Templates*. The *Documents* (requirements documents) associated with an annotation work are uploaded by the manager. LASR also allows the manager to keep the documents private, and, thus, limit the permissions of viewing and annotating of the documents to a selected groups of annotators only. This helps applying the constraints of signing to non-disclosure agreements by the annotators before contributing to private annotation work.

The interface of an annotation work is described by two templates in LASR: (i) *Questionnaire Template*, and (ii) *Summary Template*. Both templates are stored as XML files at the backend. The questionnaire templates contain details on:

- What type of requirements *Instances* are to be annotated by the annotators: “*Noun-Phrases*”, or “*Sentences*”, or “*Sub-sections*”, or “*Sections*”, or the whole “*Document*”.
- What *Questions* to ask the annotators.
- What possible choices of *Answers* to provide to the annotators, for each question (here, the answers contain the annotation labels).
- And, what additional related questions to ask the annotators, only when they choose specific answer(s) to certain question(s) earlier (this helps to implement the hierarchical dependency among the annotation labels).

The summary template, on the other hand, contains details on what results to show to an annotator, after he/she completes the annotation work. The template also provides options to include simple arithmetic computations (i.e. count, sum, average, +, -, *,/) over given answers. This is important for defining some annotation work performed during requirements analysis phase that demand instant feedback with additional calculations on the given annotations (e.g. COSMIC FSM).

5.9.3 Problem Domain

A requirements document that is uploaded to LASR must belong to one specific *Problem Domain*. A problem domain describes the domain of a group of software problems. The classification of problem domains varies from one organization to another based on their internal needs. For example, Microsoft Corporation (Microsoft Corp., 2011) prescribes 40 different classes of problem domains for software products. LASR, therefore, allows the decomposition of problem domains into open categories that can be customized to have an organization-specific classification. The following attributes describe a problem domain in LASR,

```
id:  $\mathbb{N}$ 
name: STRING
application_type: { "desktop", "web", "plug-in", "real-time", "developer",
                   "publisher", "embedded", "business", "utility", "game",
                   "academic", "communication", "system", "portable",
                   "graphics", "multimedia", "driver", "framework",
                   "research", "prototype", "component", "other" }
deployment_type: { "private", "public-open", "public-closed" }
```

— where *id* allows us identify each problem domain uniquely, and *application_type* and *deployment_type* provide additional nominal features for higher level classification of the problem domains in LASR.

Other concepts in the domain model include the Data-group that is analogous to domain entity, which is an aggregation of data-attributes, and belongs to a specific problem domain. Each data-group name associated with a problem domain is created by an annotator during the annotation of a requirements document belonging to the same problem domain.

5.10 User Roles of LASR

The collaborative environment of LASR is targeted to serve users with different goals securely. It provides a hierarchical arrangement of its access levels, each supporting the roles of its users having different sets of permissions. The five different user roles in LASR are: (i) the unregistered users (guests), (ii) the registered users (annotators), (iii) the project members, (iv) the project managers, and (v) the administrator. Table 15 presents the user roles, their respective access levels and the activities permitted to the users by LASR.

Access Level	<i>Anonymous</i>	<i>Low</i>	<i>Medium</i>	<i>High</i>	<i>Administrator</i>
User Role(s)	<i>Unregistered Users</i>	<i>Registered Users</i>	<i>Project Members</i>	<i>Project Managers</i>	<i>Administrator</i>
Learn about LASR / View brief description of all annotation work / Register an account	x	x	X	x	x
View results of all annotation work / Request to annotate / Perform annotation / Submit requirements document / Apply to be a project manager		x	X	x	x
Accept or reject requests to annotate / Create & publish detailed profile			X	x	x
Setup a project / Define annotation work / Create annotation templates				x	x
Have super user permissions for backend maintenance / Accept application to be a project manager					x

Table 15: Activities Permitted to Different User Roles in LASR

5.11 Conclusion

In this chapter, we discussed how a linguistic annotation tool with the addition of a few key features, listed in Section 5.2.2, can effectively aid the complex tasks of functional size measurement from software requirements. We presented our annotation tool, LASR, that includes these features, and showed how it helped a group of annotators with minimum training to measure the functional size following the COSMIC FSM standard accurately.

The unique feature set of LASR not only helped in attaining size measurements of higher accuracy, but also helped eliminating the need of running adjudication sessions to resolve disagreement of the annotators, and, thus, reducing the cost of large scale annotation. It improved the idea of crowd-sourcing by introducing the method of expert seeding of the true gold-standard annotations for a portion of the corpora, and thus, allowing real-time evaluation of the skills of annotators. The interface of LASR also helped minimizing the time for the annotation tasks related to COSMIC FSM.

In the next chapter (Chapter 6), we investigate if functional size can be approximated without requirements formalization. We use a supervised learning-based approach to determine the most discriminating linguistic features of informally written textual requirements for approximating functional size. We then ran our experiments to check if a text mining approach can predict the nominal functional size classes of textual requirements and moderately agree with the gold-standard size classifications.

Chapter 6

Automated Approximation of Functional Size

“Simplicity and repose are the qualities that measure the true value of any work of art.”
— *Frank Lloyd Wright*

6.1 Introduction

This chapter describes our approach of automated approximation of functional size. It extends the idea presented in the Estimation by Analogy approach (Shepperd & Cartwright, 2001) and the Easy and Quick (E&Q) measurement approach (Meli, 1997), that was originated in the IFPUG standard (ISO/IEC 20926, 2003). The applicability of this approach in COSMIC was manually demonstrated by (Santillo *et al.*, 2005). Our approach automates the process by using the historical data of an organization that needs to be stored for the purpose of generating the datasets for training and testing our classifier. The required historical data must contain sets of textual user requirements of any quality, where each set corresponds to a unique functional process and is measured in terms of COSMIC function points (CFP) by human measurers.

6.2 Overview of Our Approach

Our approach uses these sets of textual requirements and their recorded CFP measurements from all the achieved projects to automatically select and extract the linguistic features that discriminate the functional processes by their CFP sizes and train our text classifier to automatically classify new sets of textual requirements into a predefined number of size

classes. The output size classes indicate the approximated range of functional size. Our approach also outputs a linguistic model for classifying by functional size classes that provides traceability links between the outputs functional size and the input textual requirements. Figure 26 illustrates an overview of this workflow showing the inputs and outputs of our approach.

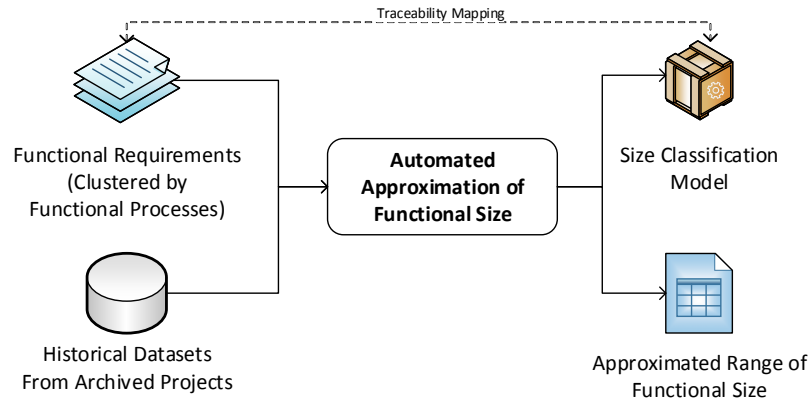


Figure 26: Inputs and Outputs of Our Approach for Automated Approximation of Functional Size

We present the details of our approach in the following sections.

6.2.1 CFP Measurement

In case the historical database of an organization is not available or is not in the form required by our approach, our first step would be to build the historical dataset by manually measuring the COSMIC size of the functional processes in units of CFP (COSMIC Function Point). The available textual description of the user requirements corresponding to each functional process is used for this purpose. Here, for each requirements statement belonging to a functional process, the human measurer first identifies how many different types of data-movements are expressed by the statement, and then, how many data-groups participate in each of the types of data-movements present in the statement. Following COSMIC, the sum of number of data-groups for each type of data-movements indicates the total CFP size of one requirements statement. The measurer repeats this step for the rest of the requirements statements within the functional process to measure their sizes in CFP. Summing up their sizes results in the CFP count for the whole functional process. The measurer then again adds

the CFP sizes of all of the functional processes together to obtain the respective CFP count of the whole system. Table 16 illustrates the CFP counting process with a hypothetical example of a system consisting of two functional processes.

Functional processes	User requirements	Types of Data-movements expressed by the statement:	Number of Data-groups involved a data-movement	Size in CFP
FPROC ₁	1.1 User requests to view the detailed information of one item.	Entry	2	2
		Read	1	1
	Size of statement 1.1 =			3
	1.2 System displays detailed item information.	Exit	1	1
		Size of statement 1.2 =		
Total size of FPROC ₁ =			3+1 = 4	
FPROC ₂	2.1 When user requests to add the item to the shopping cart, system adds it and displays the cart.	Entry	2	2
		Write	1	1
		Exit	1	1
	Size of statement 2.1 =			4
	Total size of FPROC ₂ =			4
Total size of the whole system =				4 + 4 = 8

Table 16: A Hypothetical Example of CFP Calculation

Our approach requires these measurement data to be saved in the historical database for the past completed projects. For this work, we will need the CFP count for each of the functional processes that have been measured, along with the textual requirements associated to a functional process. Figure 27 illustrates the steps of building a historical database, when a historical database is not already available.

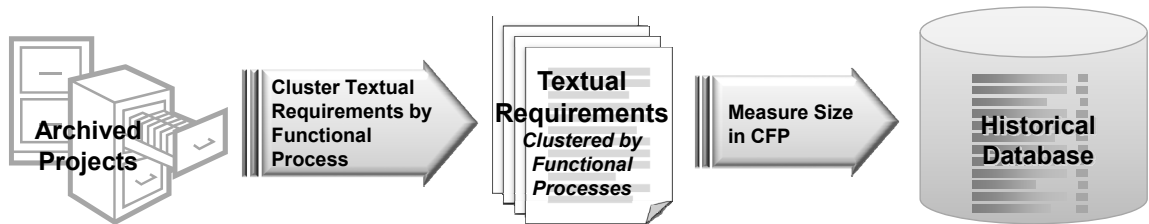


Figure 27: Building a Historical Database

6.2.2 Class Annotation of Functional Processes

Once we have gathered the historical dataset, we need to define classes of functional processes, based on their sizes in CFP, to be used later in the automatic classification task. To do this, we performed a box-plot analysis on the CFP size values from our historical dataset, to produce four different classes of functional processes, based on their sizes in CFP. Table 17 shows the defined ranges of these classes.

Size Classes	Ranges
Small	[0, Lower Quartile)
Medium	[Lower Quartile, Median)
Large	[Median, Upper Quartile)
Complex	[Upper Quartile, ∞)

Table 17: Ranges of CFP Values to Define the Classes

Here, the lower quartile would cut off the lowest 25% of all the recorded CFP size data from the historical database. The median would divide the dataset by 50%, and the upper quartile cuts off the highest 25% of the dataset.

These four sets of ranges allow us to annotate the textual requirements of the functional processes automatically into four fuzzy size classes. In our class ranges, we keep the minimum and the maximum values as 0 and ∞ , respectively, instead of the sample minimum or the sample maximum, like in an actual box-plot analysis; so that, if the new unseen sample is an outlier compared to the historical dataset, it would still get classified into a class.

After defining the class boundaries automatically, we then calculate the mean, the minimum and the maximum for each of the classes, to designate the range of the approximate size for each of the classes. Figure 28 illustrates the process of automatic class annotation described in this section.

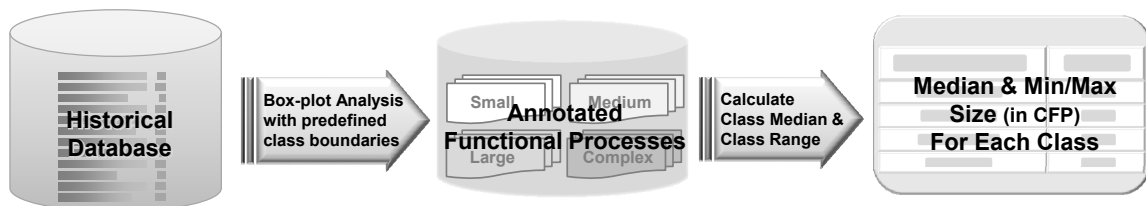


Figure 28: Class Annotation by Box-Plot Analysis

6.2.3 Text Mining

Our next step consists of extracting linguistic features from the textual requirements belonging to each of the functional processes from our training dataset, to train a text classification algorithm that can automatically classify a new set of textual requirements belonging to a functional process into one of the classes defined above (i.e. *Small*, *Medium*, *Large* or *Complex*). The classifier will then simply approximate the size of the functional processes by outputting its size as the calculated mean value of the class it belongs to, along with the minimum and the maximum seen *CFP* value for that class to indicate possible variation in the approximation; and, thus, provide the quickest possible approximation of the COSMIC functional size from textual requirements that are not formalized and can be written in any quality. Figure 29 shows the steps of this process.

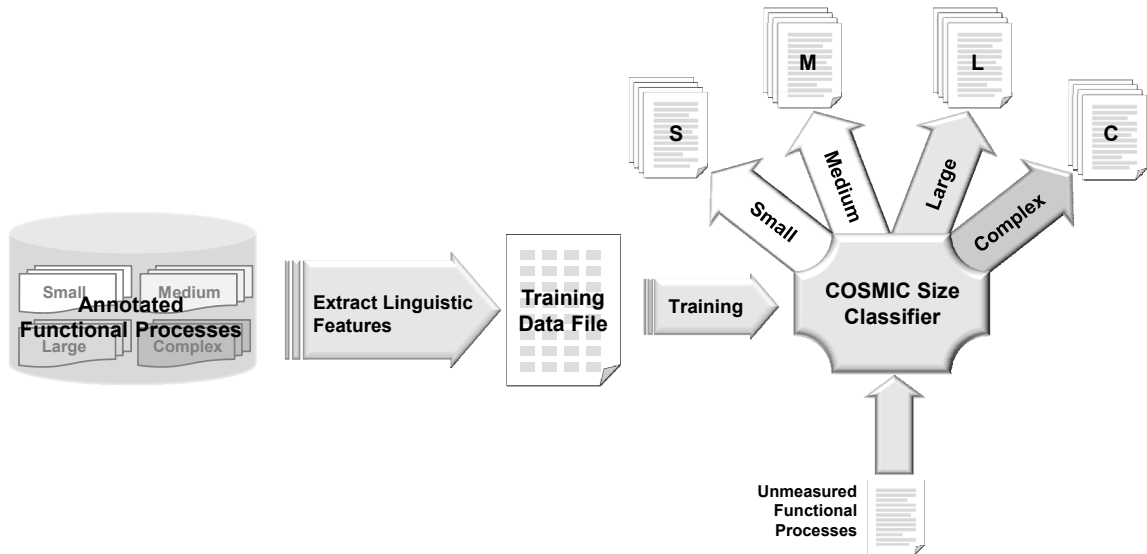


Figure 29: Text Mining for Fast Approximation of COSMIC Functional Size

6.3 Experiments

As a proof of concept of our approach, we addressed our research question Q_4 , as presented in Section 4.2.3, by performing a preliminary experiment with four different case studies: two industrial projects from SAP Labs, Canada, and two university projects. They are all completed projects and are from different domains. Their requirements documents vary in size (from about 2,000 words to 11,000 words) and contain from 3 to 32 distinct functional processes. Table 18 shows some characteristics of these case studies.

ID	Source	Title	Type of Application	Size of Requirements Document	Functional Processes extracted
C3	University	Course Registration System	Academic	3,072 words	14
C4	University	IEEE Montreal Website	Web (Public)	5,611 words	32
C5	Industry (SAP)	<i>(undisclosed)</i>	Business	11,371 words	12
C6	Industry (SAP)	<i>(undisclosed)</i>	Business	1,955 words	3
Total number of functional processes extracted =					61

Table 18: Summary of the Case Studies

We manually pre-processed these requirements to extract sets of requirements sentences each of which belong to a distinct functional process. This mimics the available set of user requirements before an iteration starts in an agile development process. From all four requirements documents, we were able to extract 61 sets of textual requirements, each belonging to a distinct functional process.

We used five human measurers, all graduate students skilled to perform COSMIC FSM from requirements documents, to measure the CFP of these 61 functional processes, similarly to what is shown in Table 1. The CFP values and the textual requirements of the 61 functional processes built our historical dataset. The frequency distribution of the CFP values in our historical database is shown in Figure 30. The figure shows that most functional processes (17 of them) were of size **6 CFP**. The box-plot on top of the histogram points out the lower quartile, the upper quartile, the sample minimum and the sample maximum, and also indicates that the median size is **6 CFP** in our historical database.

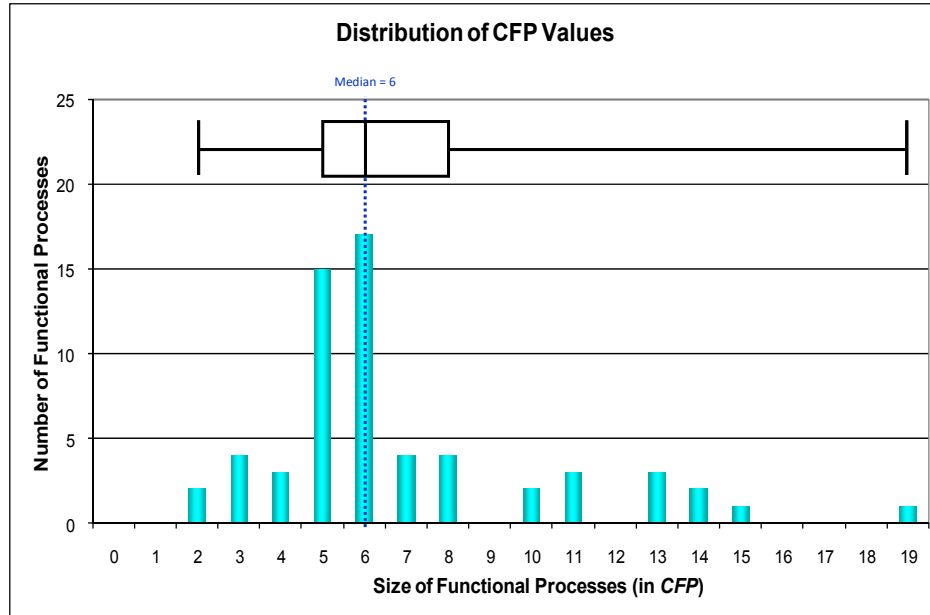


Figure 30: Distribution (with a box plot) of CFP Values in Our Historical Database

6.3.1 Corpus Annotation

As mentioned in Section 6.2.2, in order to define the ranges of our four size classes, we performed a box-plot analysis on the *CFP* values of our historical database. The resulting boundary points are:

Median	= 6 <i>CFP</i>
Lower Quartile	= 5 <i>CFP</i>
Upper Quartile	= 8 <i>CFP</i>
Sample Minimum	= 2 <i>CFP</i>
Sample Maximum	= 19 <i>CFP</i>

Therefore, according to the ranges defined in Table 17 in Section 6.2.2, the actual *CFP* ranges for the four size classes for our historical database are:

Small:	[0,5)
Medium:	[5,6)
Large:	[6,8)
Complex:	[8,∞)

We then followed these ranges to automatically annotate the sets of textual requirements belonging to the 61 functional processes into the four size classes - where 9 (15%) functional processes were annotated as *Small*, 15 (25%) were *Medium*, 21 (34%) were *Large*, and 16 (26%) were annotated as *Complex*.

We then collected from our historical database the class data, i.e. the mean, the minimum and the maximum sizes for each of these classes, so that the size of a newly classified functional process belonging to any of these four classes can be approximated by its class data. The resultant class data is shown in Table 19.

Class	Median Size	Minimum Size	Maximum Size	Approximation Error
Small	3 <i>CFP</i>	2 <i>CFP</i>	4 <i>CFP</i>	[-1,1] <i>CFP</i>
Medium	5 <i>CFP</i>	5 <i>CFP</i>	5 <i>CFP</i>	0 <i>CFP</i>
Large	6 <i>CFP</i>	6 <i>CFP</i>	7 <i>CFP</i>	[0,1] <i>CFP</i>
Complex	11 <i>CFP</i>	8 <i>CFP</i>	19 <i>CFP</i>	[-3,8] <i>CFP</i>

Table 19: Data to be Associated with a Functional Process to Approximate Its Size

It should be noted that due to the small number of functional processes that we currently have collected in our historical database, Table 19 does not show much variation of size among the classes, especially between the classes *Medium* and *Large*. This drastically reduces the error margin of our approximation and, therefore, a correctly approximated size will be more precise in the current case. For example, when a functional process will be correctly classified as *Medium* by our text miner, our system will indicate, according to the class data, shown in Table 19, that its approximate (i.e. the mean) size is **5 CFP**, which actually is the precise size value of the functional process instead of an approximation. This is because only the functional processes of size **5 CFP** are set to the *Medium* class by our box-plot analysis. As *CFP* size values are always integer numbers, it allows zero margin of error in our approximation of the size of a functional process that belongs to the *Medium* class. Similarly, the error margin of the *Small* and the *Large* classes are also very small. This will also make the task of discriminating between the close classes harder than discriminating between widely-varying classes.

6.3.2 Syntactic Feature Selection

To perform the classification task, we considered a large pool of linguistic features that can be extracted by a syntactic parser. In this regards, we used the Stanford Parser (Klein & Manning, 2003) (equipped with Brill's POS tagger (Brill, 1992) and a morphological stemmer) to morphologically stem the words and extract many linguistic features, e.g. the frequency of words appearing in different parts-of-speech categories. As we have the actual CFP values in our historical dataset, we sorted the linguistic features based on their correlation with the CFP values. The ten highest correlated features are listed in Table 20.

ID	Features (<i>Frequency of..</i>)	Correlation with CFP
1	Noun Phrases	0.4640
2	Parentheses	0.4408
3	Active Verbs	0.4036
4	Tokens in Parentheses	0.4001
5	Conjunctions	0.3990
6	Pronouns	0.3697
7	Verb Phrases	0.3605
8	Words	0.3595
9	Sentences	0.3586
10	Uniques (<i>hapax legomena</i>)	0.3317

Table 20: Ten Linguistic Features Most Highly Correlated with CFP

The correlation shows the ten syntactic features that influence COSMIC functional size the most. The intuitive reasons for them are explained below.

Frequency of Noun Phrases (#1): No matter how poorly a requirement is described, the involvement of a data-group in a particular data-movement is typically indicated by the presence of a noun phase. Therefore, if a functional process contains more noun phrases, the chances are that its data-movements involve more data-groups and its size is larger.

Frequency of Parentheses (#2) & Number of tokens inside parentheses (#4): When complex functional processes are described in textual requirements, parentheses are often used to provide brief explanations in a limited scope. Thus, a higher number of parentheses/Number of tokens inside parentheses can sometimes indicate a complex functional process.

Frequency of Active Verbs(#3) & Verb Phrases(#7): Verbs in active form are frequently used to describe actions and, hence, are often used in larger numbers in textual requirements to explain data-movements, as data-movements result from actions carried out by the user or the system or an external system.

Frequency of Pronouns(#6): A longer description in textual requirements for a functional process often indicates its complexity, and requires the use of more pronouns and other referring expressions within the functional process to maintain cohesion.

Number of Words(#8), Conjunctions(#5), Sentences(#9) and Uniques(#10): In general, lengthy descriptions of the requirements (hence, a higher frequency of words, sentences and unique words) often indicate a more complex functional process.

In addition to the above syntactic features, we also looked at possible keywords that can be used in our classification task.

6.3.3 Lexical Feature Selection

Studies [e.g. (Hussain, Kosseim, & Ormandjieva, 2008; Wiebe, Wilson, Bruce, Bell, & Martin, 2004)] have shown that using keywords grouped into particular part-of-speech categories can help to obtain good results in various text mining problems. For our work, we have, therefore, considered lists of keywords, where each list belongs to a given part-of-speech category. We chose three open-class part-of-speech groups for these keywords to be selected. They are: Noun-keywords (coded as: *NN_keyword*), Verb-keywords (coded as: *VB_keyword*), and Adjective-keywords (coded as: *JJ_keyword*).

We generate finite lists of these keywords based on two different probabilistic measures, as described in (Hussain, Kosseim, & Ormandjieva, 2008), that take into account how many more times the keywords occur in one class of the training set than the other class. A cut-off threshold is then used to reduce the list to keep only the top most discriminating words. For example, the three lists that were automatically generated by this process from our training set during a single fold of 10-fold cross-validation are shown in Table 21.

NN_keyword	VB_keyword	JJ_keyword
user	ensure	supplied
category	get	current
quota	choose	previous
content	start	available
default	restart	
chart	fill	
...	...	

Table 21: Some of the Keywords of POS Category: Noun, Verb and Adjective

These three lists constituted three additional features for our classification task. Thus, when we extract the features, we counted one of the keyword feature, for example, as how many times words from its keyword-list appears in the set of requirements of a functional process, and appearing in the same part-of-speech class.

6.3.4 Feature Extraction and Classification

To classify the sets of textual requirements belonging to different functional processes, we developed a Java-based feature extraction program that uses the Stanford Parser (Klein & Manning, 2003) to extract the values of all the syntactic and keyword features mentioned above, and uses Weka (Witten & Frank, 2005) to train and test the C4.5 decision tree learning algorithm (Quinlan, 1993). We used the implementation of the C4.5 (revision 8) that comes with Weka (as J48), setting its parameter for the minimum number of instances allowed in a leaf to 6 to counter possible chances of over-fitting. The results are discussed in the next section. We also trained/tested with a Naïve Bayes classifier (John & Langley, 1995), and a logistic classifier (le Cessie & van Houwelingen, 1992). The C4.5 decision tree-based classifier performed the best in comparison to the other classifiers with more consistent results during 10-fold cross-validation.

6.4 Results and Analysis

The results of the classification were very moderate when using the whole dataset for training and testing. Since the dataset was not very large, we could not use a separate dataset for testing, and we could only use cross-validation, which can be very harsh on the performance, when the number of instances is very low. Yet, the classifier results did not drop significantly. Table 22 shows a summary of the results.

	Scheme	Correctly Classified Sentences	Incorrectly Classified Sentences	Kappa	Comment
Corpus Size = 61 (sets of textual requirements, each set representing a functional process)	Training + Testing on same set	45 (73.77%)	16 (26.23%)	0.6414	Tree is of desirable characteristics, not sparse, and also not flat. None of the branches are wrongly directed.
	Cross-validation (10 Folds)	41 (67.21%)	20 (32.79%)	0.5485	

Table 22: Summary of the Results

The resultant decision tree after training on the complete dataset is shown in Figure 31. As the figure shows, the tree came out well-formed and of desirable characteristics — not sparse, and also not flat. Also, none of its branches are wrongly directed.

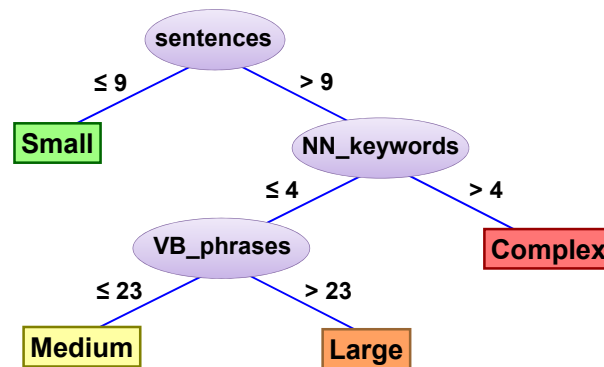


Figure 31: The Resultant C4.5 Decision Tree after Training with the Complete Dataset

Although the Kappa results of Table 22 shows stable and moderate results in terms of performance with the 10-fold-crossvalidation, the confusion matrix of Table 23 shows that the classifier struggled to classify functional processes of size *Medium* into the correct class; classifying only 47% of them (7 out of 15) correctly. We can also see that the mistakes the classifier made with the *Medium* sized functional processes are mostly because it confused them as *Large* (for example, it classified another 7 out of the 15 *Medium* functional processes incorrectly into the size class *Large*). The reason for this can be understood by the fact discussed in Section 6.3.1, where, in Table 19, we see that our box-plot analysis automatically chose zero approximation error for the class *Medium*. It, therefore, became the hardest class to classify among the other classes, carrying very minute differences from its adjacent class *Large*, which also has a smaller margin of approximation error. Thus, when our classifier correctly classifies a functional process as *Medium*, it does not really

approximates its size; rather it accurately identifies its precise size value, which is **5 CFP**. Again, when the classifier mistakenly classified a *Medium* functional process as *Large*, the error in size approximation that it made is of only **1 CFP** value. If we had a larger number of instances, there would have been wider variation of size values in our historical database. We believe that this would make the classification task easier for our classifier allowing the learning algorithm to find the threshold values for the other unused linguistic features and, thus, utilize them in making fine-grained distinction and render better results.

By analyzing Table 23, we can see that the classifier had difficulty in identifying the functional processes of size *Small*. Although it classified 7 out of 9 *Small* functional processes correctly as *Small*, it misclassified some *Medium*, *Large*, and even *Complex* functional processes as *Small* (see the 1st column of Table 23). Here, again, we believe that the small size of our dataset (e.g. we had only 9 instances of size *Small*) may be the cause. It should be noted that these results were extracted during cross-validation of 10 random folds, which can significantly reduce the number of training instances for a particular class during a single fold in a skewed corpus. In our case, for example, during one fold, the number of training instances for the *Small* class went minimum of only 2 instances, which were inadequate for the learning algorithm to discover the thresholds of most of the discriminating linguistic features that we selected for this work. We, therefore, believe that these results would improve with the introduction of more instances in our dataset.

	Classified as			
	Small	Medium	Large	Complex
Small	7	0	1	1
Medium	1	7	7	0
Large	2	1	16	2
Complex	2	0	3	11

Table 23: Confusion Matrix When Using 10-fold Cross-Validation

These phenomena are also reflected in the precision and recall values shown in Table 24. Moreover, it also shows that a good performance on average attained by the classifier with such a small dataset.

Size Class	Precision	Recall	F-Measure
Small	0.583	0.778	0.667
Medium	0.875	0.467	0.609
Large	0.593	0.762	0.667
Complex	0.786	0.688	0.733
Mean	0.709	0.673	0.669

Table 24: Precision, Recall and F-Measure, When Using 10-fold Cross-Validation

We can also demonstrate by showing that if we had less number of classes, i.e. two or three size classes, the available number of instances would have been enough for a more realistic classification task. To show that, we developed both a two-class size classifier (classifying functional processes into *Small* and *Large* classes), and a three-class size classifier (classifying functional processes into *Small*, *Medium* and *Large* classes) using the same principles and the same sets of features described earlier in this chapter. The results were significantly better, attaining mean f-measures of 0.802 and 0.746 for the 2-class and the 3-class classifiers respectively. The summary of using 10-fold cross-validation with both classifiers is shown in Table 25.

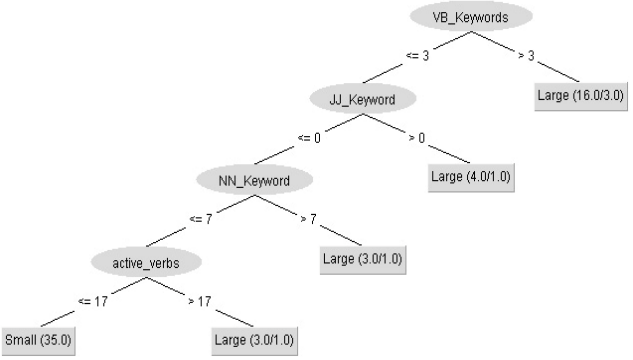
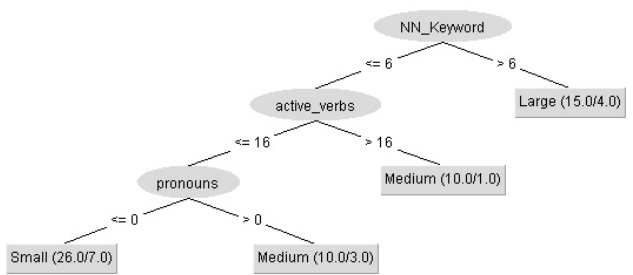
Classifier	Kappa	Size Classes	Precision	Recall	Tree
2-class Classifier	0.606	Small	0.895	0.829	
		Large	0.696	0.8	
3-class Classifier	0.575	Small	0.677	0.875	
		Medium	0.794	0.721	
		Large	0.9	0.73	

Table 25: 10-fold Cross-Validation Results of Using a 2-Class and 3-Class Classifier

6.5 Conclusion

In this chapter, we have presented our research activities that shows that the classification of textual requirements in terms of their functional size is plausible using linguistic features. Since our work uses a supervised text mining approach, where we need experts to build the historical database by manually measuring the COSMIC functional size from textual requirements, we could not train and test our system with a large number of samples. Yet, the results that we were able to gather by cross-validating on such small number of samples show a promising behavior of the classifier in terms of its performance. We have been able to identify automatically a set of highly discriminating linguistic features that can effectively classify textual requirements in terms of functional size.

It should be mentioned that we have not yet tested our approach as to be used with requirements written in variable level of quality. We believe that this approach would be organization-specific, where textual requirements saved in the historical dataset should all be written in the same format or writing style having similar quality. This would allow our classifier to pick the best set of features and set the best thresholds that would classify new requirements written in similar style and quality.

This chapter has described our approach of exploring linguistic features for extracting the conceptual artifacts of the COSMIC functional size measurement standard from textual software requirements. Here, we used the annotated corpus, presented in the previous chapter in Section 5.3.2, to identify discriminating features at different syntactic and lexical levels. We then used the gold-standard annotation labels of the corpus to devise a large number of training and testing datasets to experiment our supervised learning-based and rule-based approaches for functional size measurement (FSM).

In the next chapter (Chapter 7), we experiment with the linguistic features in a heuristic-based and supervised learning-based approaches for extracting the artifacts of an FSM model. The accuracy and usability of the manual process of FSM are then compared with our automated FSM approach.

Chapter 7

Automated FSM Modeling

“The only place success comes before work is in the dictionary.”
— *Vince Lombardi*

7.1 Introduction

This chapter describes our approach of exploring linguistic features for automatically extracting the conceptual artifacts of the COSMIC functional size measurement (FSM) standard from textual software requirements. In this phase, we used our annotated corpus, as presented in Section 5.3.2, to identify discriminating features at different syntactic and lexical levels. We then used the gold-standard annotation labels of the corpus to devise a large number of training and testing datasets to experiment our supervised learning-based and rule-based approaches for functional size measurement (FSM).

Studies (Saadaoui, Majchrowski, & Ponsard, 2009; COSMIC, 2011; Habela, Głowacki, Serafiński, & Subieta, 2005; Rule & Rule, 2011) show that when the requirements are formalized and decomposed enough to make the FSM modeling artifacts clearly identifiable to the expert measurers, COSMIC FSM can be applied objectively to achieve one correct measurement across multiple measurers. Thus, (Condori-Fernández, Abrahão, & Pastor, 2007) also demonstrated that a static set of rules can be applied to extract COSMIC functional size from requirements that have been highly formalized to UML System Sequence diagrams. However, when dealing with software requirements written in natural language, the work depends on the expertise of human measurers to presume the conceptual models that can be realized from these requirements and use this idea to base their judgment on identifying the COSMIC modeling artifacts from the requirements.

Our automation approaches discussed in this chapter take as inputs the functional requirements of a software in textual form clustered by functional processes and the lexical resources related to the problem domain of the same software. The outputs of our approaches are the range of functional size of the software and a measurement model for tracing the output of functional size to its originating textual requirements. Figure 32 illustrates an overview of the workflow of our approach.

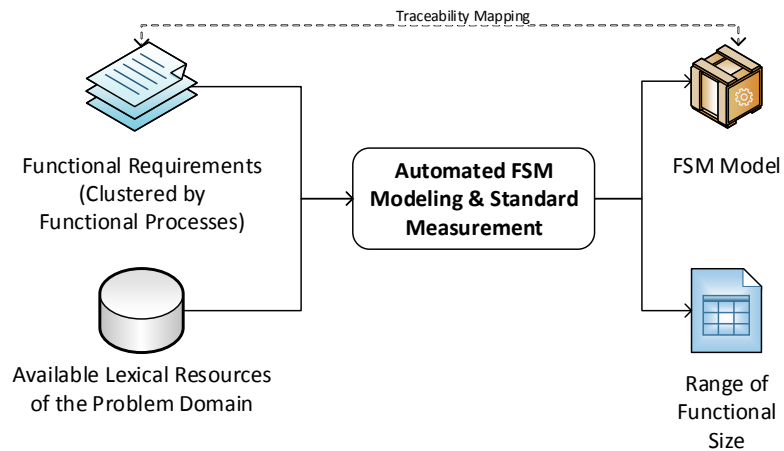


Figure 32: Inputs and Outputs of Our Approach for Automated FSM Modelling

7.2 Our Approach

Our approach aims to learn the linguistic aspects of textual software requirements that will guide us in automatically extracting the conceptual artifacts of our formalization of COSMIC FSM model, as presented in Section 4.3. In particular, we would be extracting COSMIC data-movements and the COSMIC data-groups from our input functional requirements that have already been clustered into functional processes. Our approach thus not only outputs the range of functional size of the functional process in CFP, but also helps us to provide traceability on output CFP values to textual requirements.

In our observation of applying COSMIC FSM over textual requirements, we noticed that the semantics of the COSMIC data-movements can be realized from textual requirements by using both the syntactical and the lexical information embedded in the requirements, as

reflected by our FSM model, as discussed in Section 4.3. Figure 33 illustrates the pipeline of our approach showing the workflow of the major steps for automated FSM modeling and measuring the functional size in COSMIC Function Points (CFP).

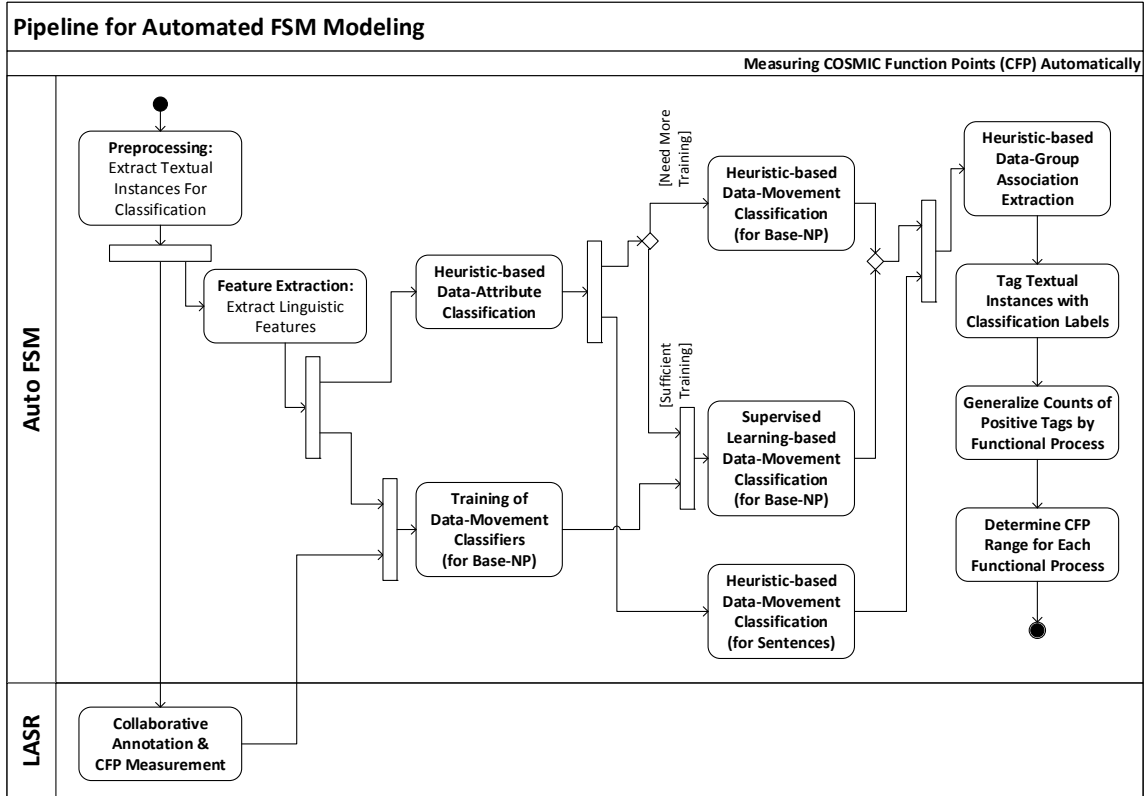


Figure 33: Major Steps of Our Approach for Automated FSM Modeling

We discuss the details of each of these steps, as illustrated in Figure 33, in the following sections.

7.3 Preprocessing

In the preprocessing step, we use lightweight natural language processing techniques to perform sentence segmentation and parts-of-speech tagging over our input textual instances of functional requirements, which had already been clustered into functional processes by the steps mentioned in Section 4.4.3. The workflow of our preprocessing step is shown in Figure 34.

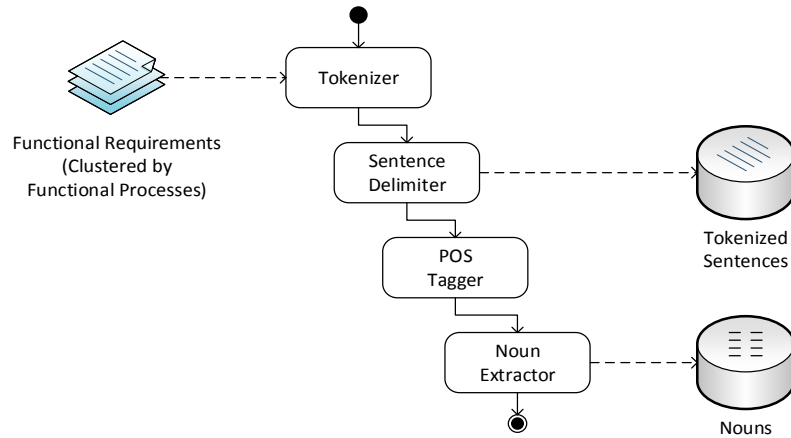


Figure 34: Workflow of the Preprocessing Step

As shown in Figure 34, we first tokenize the textual instances by words and punctuations, use our custom rules for sentence segmentation, use a custom-trained transformational tagger like the Brill tagger (Brill, 1992) to compute the part-of-speech (POS) class of the word tokens from the sentences, and, finally, identify both the pronoun-type words (i.e. with POS tags *PRP*) and the longest chunks of consecutive noun-type words (i.e. with POS tags *NN*, *NNS*, *NNP*, *NNPS* etc.) as individual instances of base noun-phrases that are to be used in our next step. We also identify their heads (rightmost noun-type word, if an instance is a noun compound, or the word itself, if the instance is noun of single word, or a pronoun). Moreover, we use a morphological stemmer to find the roots and affixes of all words in the sentence. We then use the extracted sentences and noun-phrases for collaborative manual annotation with LASR, as discussed in Chapter 5, and also, in parallel, move on to perform our next step of feature extraction.

7.4 Extraction of Linguistic Features

The step of extracting the linguistic features first takes the tokenized sentences the previous step and extract the syntactical dependencies between each pair of tokens using a syntactic parser, e.g. the Stanford Parser (Klein & Manning, 2003). We then evaluate a set of syntactic features from our extracted noun-phrases based on the dependency relationships, and use the lexical database built from one of our previous steps, as discussed in Section 4.4.3, to identify a set of their lexical features. We then also compute a set of complex semantic features

combining both the extracted syntactic and lexical features. Finally, we compose the feature values into a data file that is to be used for training/testing our classifiers. The workflow of our feature extraction step is presented in Figure 35.

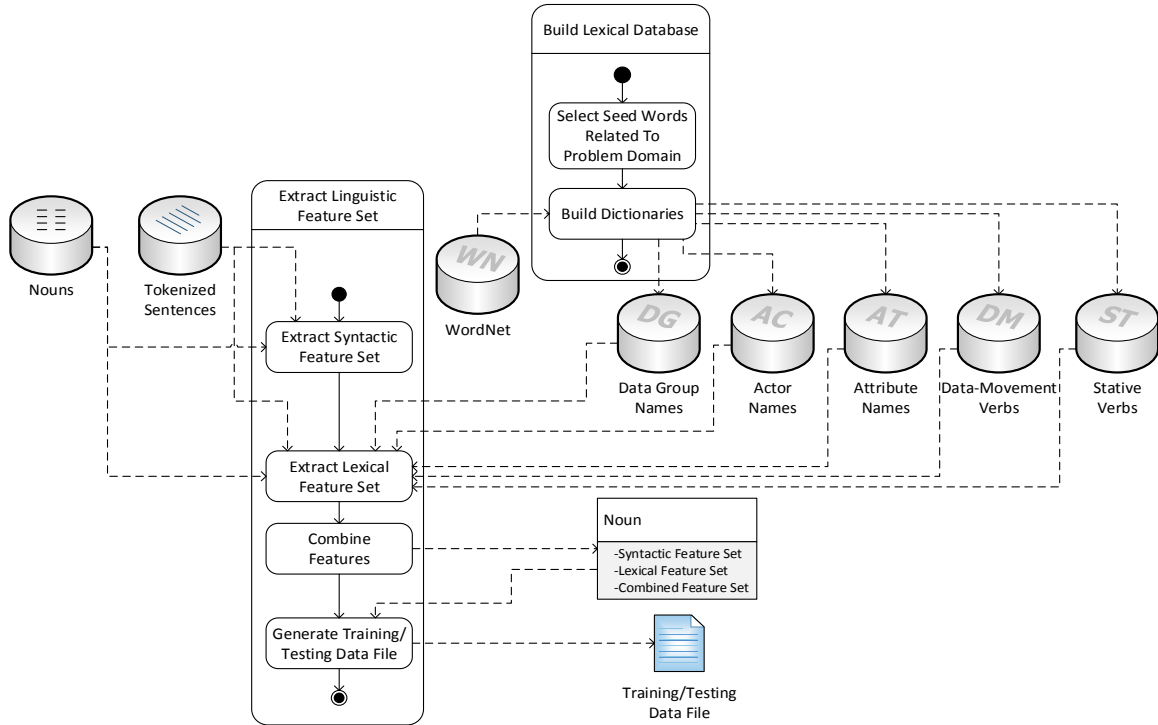


Figure 35: Workflow of the Feature Extraction Step

As shown in Figure 35, we extracted three different sets of features that are presented below.

7.4.1 Lexical Feature Set

We select our lexical database, as described in Section 4.4.3, corresponding to the problem domain of our input requirements to identify discriminating lexical features that take into account the domain-specific keywords used in requirements text (the keywords, used for our experiments, are listed in Appendix A). We then extract these features from our input noun-phrases and verb-type word tokens. Some lexical features, e.g. “the noun-phrase *is an*

*actor*¹²” attached to the noun-phrases and “the verb *is a data-movement verb*¹³” attached to the verb-type words, are only used to derive our set of combined features, as presented in Section 7.4.3.

On the other hand, the two lexical features, which are extracted from the noun-phrases and retained in our final pool of features, are listed below:

- F1: “The noun-phrase partly contains an attribute name (possible value: *true*, or *false*)”
- F2: “The noun-phrase partly contains a data-group name (possible value: *true*, or *false*)”

Here, the attribute and data-group names are matched from our selected lexical database. We enriched the content of our lexical database using WordNet (Miller, 1995; Princeton University, 2010), as described in Section 4.4.3.

7.4.2 Syntactic Feature Set

We extracted our syntactic features from our input noun-phrases based on its syntactic relation with other words of the tokenized sentences as derived by using a syntactic parser. We used the Stanford Parser (Klein & Manning, 2003), together with our custom linguistic rules that were written in the JAPE scripting language of the GATE environment (Cunningham H., *et al.*, 2011), for building our syntactic feature extractor. Our custom rules extended the scope of the syntactic parser to consider the subjects and the objects of a verb like agents and patients of an action respectively, even when the verb is in passive form. These rules also take into account of conjunctions and co-references caused by the uses of

¹² If the stemmed *head* of a noun instance contains a substring that exists in the vocabulary of actor names from our lexical database, we identify that the noun *is an actor*. Here, the head of our noun instance is selected as follows: (i) if the instance consists of only one noun token, then its head is the noun token itself; (ii) if the instance is noun compound or consists of multiple noun tokens, then its head is the rightmost noun token; and (iii) if the instance is a pronoun, then its head is the pronoun itself.

¹³ If a stemmed verb token exists in the vocabulary of a certain type (*Entry*, *Exit*, *Read*, or *Write*) of data-movement verbs from our lexical database, we then identify that the verb *is a data-movement* and of the respective type.

personal pronouns and relative pronouns, and find the appropriate subject and object relations between verbs and noun-phrases to attach the feature value accordingly. Moreover, our rules also identify the presence of the negation modifiers in various syntactic relations within a sentence to denote a noun-phrase (and also a verb) as part of a negative expression¹⁴.

Our final syntactic features are listed below:

- F3: “The noun-phrase is a direct object (possible value: *true*, or *false*)”
- F4: “The noun-phrase is in an *object-like*¹⁵ position (possible value: *true*, or *false*)”
- F5: “The noun-phrase is a subject (possible value: *true*, or *false*)”
- F6: “The noun-phrase is part of a dependent clause (possible value: *true*, or *false*)”
- F7: “Type of dependent clause that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”
- F8: “The noun-phrase is part of negative expression (possible value: *true*, or *false*)”

7.4.3 Combined Feature Set

We extracted these features by combining some of the lexical features of the noun-phrases and the verb-type words with the syntactic features of our input noun-phrases to derive some complex semantic attributes of the noun-phrases by applying a set of custom rules. There are also written in the JAPE scripting language of the GATE environment (Cunningham H., *et al.*, 2011). We added the combined features to our final pool of features in two groups: i) General Combined Features, and ii) Data-movement Verb-related Combined Features. All these features are listed below:

¹⁴ Our rules consider, along with negation modifiers, some negative implicative verbs (Karttunen, 1971), e.g. *fail*, *stop*, *refuse*, *reject*, *ignore*, *deny*, *cancel*, *forbid*, *dismiss*, *refrain*, *decline*, *disapprove*, *disallow* etc., to identify negative expressions in a sentence (see Appendix C.15 for details).

¹⁵ If a noun appears as a direct or indirect object to a verb (or as a syntactic nominal subject to a verb in passive form), or as a prepositional object to a direct/indirect object of a verb (or to a syntactic nominal subject to a verb in passive form), or in a *chain* of prepositional objects linked to a direct/indirect object of a verb (or to a syntactic nominal subject to a verb in passive form), then we identify that the noun appears in an *object-like* position.

i. General Combined Features:

- F9: “The noun-phrase *owns an attribute*¹⁶ (possible value: *true*, or *false*)”
- F10: “The noun-phrase *belongs to a data-group*¹⁷ (possible value: *true*, or *false*)”
- F11: “The noun-phrase is a subject of a Stative verb (possible value: *true*, or *false*)”
- F12: “The noun-phrase is related to an attribute name by a *chain*¹⁸ of prepositional objects (possible value: *true*, or *false*)”
- F13: “The noun-phrase is related to a data-group name by a *chain* of prepositional objects (possible value: *true*, or *false*)”
- F14: “Type of the subject of the verb, which has the noun-phrase in its object-like position (possible value: *actor subject*, or *non-actor subject*, or *none/no subject*)”
- F15: “Type of the subject of the verb, which has the noun-phrase in its dependent clause (possible value: *actor subject*, or *non-actor subject*, or *none/no subject*)”

ii. Data-Movement Verb-related Combined Features:

Our set of combined features that are related to data-movement verbs can further be grouped into five categories, based on the types of data-movement verbs, as per their corresponding vocabularies from our lexical database, as described in Section 4.4.3.

¹⁶ If an attribute name appears as a possessive determiner of the head of a noun, we then identify that the noun *owns an attribute*. For example, in “Device User’s address”, the attribute name “address” appears as a possessive determiner of the head “User” of the noun compound “Device User”. Therefore, we identify that the noun compound “Device User” owns an attribute. Again, if a noun appears as a prepositional object of an attribute name with an associated preposition “of”, we then also identify that the noun *owns an attribute*. For example, in “address of the Device User”, the noun compound “Device User” appears as a prepositional object of the attribute name “address” with an associated preposition “of”. Therefore, we identify here also that the noun compound “Device User” owns an attribute.

¹⁷ If a noun appears as a possessive determiner of a data-group name, then we identify that the noun *belongs to a data-group*. For example, in “Item’s price”, the noun-phrase “price” appears as a possessive determiner of the mention of the data-group “Item”. Therefore, we identify that the noun “price” belongs to a data-group. Again, if a data-group name appears as a prepositional object of the noun-phrase with an associated preposition “of”, we then also identify that the noun *belongs to a data-group*. For example, in “price of the Item”, the data-group name “Item” appears as a prepositional object of the noun “price” with an associated preposition “of”. Thus, we identify here also that the noun “price” belongs to a data-group.

¹⁸ When a noun appears as a prepositional object to another noun which may be a prepositional object to another noun and so on, we then define the boundary, starting from the first character of the first noun and ending at the last character of the last noun, as a *chain of prepositional objects*.

(a) For Any Type of Data-movement Verb

- F16: “The noun-phrase is in an *object-like* position to a Data-Movement verb (possible value: *true*, or *false*)”
- F17: “Type of the dependent clause of a data-movement verb that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”

(b) For *Entry* Verb

- F18: “The noun-phrase is in an *object-like* position to an *Entry* verb (possible value: *true*, or *false*)”
- F19: “Type of the dependent clause of an *Entry* verb that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”

(c) For *Exit* Verb

- F20: “The noun-phrase is in an *object-like* position to an *Exit* verb (possible value: *true*, or *false*)”
- F21: “Type of the dependent clause of an *Exit* verb that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”

(d) For *Read* Verb

- F22: “The noun-phrase is in an *object-like* position to a *Read* verb (possible value: *true*, or *false*)”
- F23: “Type of the dependent clause of a *Read* verb that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”

(e) For *Write* Verb

- F24: “The noun-phrase is in an *object-like* position to a *Write* verb (possible value: *true*, or *false*)”
- F25: “Type of the dependent clause of a *Write* verb that contains the noun-phrase (possible value: *adverbial clause*, or *clausal complement*, or *open clausal complement*, or *none*)”

Although we have extracted more linguistic features than the 25 features mentioned here, all of these 25 features were chosen to be included in our final pool of features based on their relevance in expressing the actions of data-movements in textual requirements. More details about many of these features, their significance and our associated rules for extracting them is presented in Appendix C. In our feature extraction step, we record these feature values in our training/testing data file for our corpus of functional processes.

7.5 Heuristic-based Classification

We consulted with the expert measurers, who participated in our annotation experiments, as mentioned earlier in Section 5.3.1, about their experiences of interpreting different syntactic patterns of textual requirements as indications of the occurrence of data-attributes, data-groups and data-movements. We led them to try out different linguistic forms of textual requirements and found, for example, that noun-phrases that are parts of negative senses of actions within a sentence (as also determined by our feature F8 mentioned in Section 7.4.2) do not usually convey the sense of a moving data-group. Again, in relation to the COSMIC data-movement, the noun-phrases in functional requirement that express the meaning of owning attributes, are often mentions of data-group names. Then, the noun-phrases that indicate the mentions of moving data-attributes/data-groups, tend to appear in object-like positions to data-movement verbs (as determined by our features F16, F18, F20, F22 and F24 mentioned in Section 7.4.3). Moreover, the noun-phrases that indicate the mentions of data-attributes/data-groups participating in *Entry*-type data-movements, tend to appear in object-like positions to verbs that have actor subjects (as also determined by our features F14 mentioned in Section 7.4.2).

By devising our heuristic-based classification approach, we generalized the experts' process of interpreting these complex linguistic cues of textual requirements into the COSMIC modeling artifacts. These heuristics are static rules applied over the linguistic features, that are extracted from using our previous step, as mentioned in Section 7.4. Thus, with the inclusion of new requirements instances to our corpus over time, only the (lexical) features gets enriched and evolves, but not these heuristics. However, this heuristic-based classification approach can be necessary to implement our methodology in practice,

especially when we do not have sufficient data for training our supervised learning-based classification approach, presented later in Section 7.6.

Our heuristics thus use the same linguistic features, which are extracted from using our previous step, as mentioned in Section 7.4, to classify noun-phrases and sentences from textual requirements into classes that represent COSMIC's modeling artifacts, e.g. the data-attributes, the data-groups and the data-movements. We present the details of these heuristics for each type of classifications, in the following sections.

7.5.1 Data-Attribute Classification

During the data attribute classification step, we identify the noun-phrases that appear in functional requirements sentences to represent the data-attributes. The lexical feature F1, as mentioned in Section 7.4, already indicate noun-phrases that contain the attribute names as their substrings. Additionally, we apply the following algorithm to classify a noun-phrase as a data-attribute:

```
Step 1. For each noun-phrase instance x in the input functional process d :
    1.1. If x.F1 = true and the root of the head of x exists in our vocabulary of
        attribute names, then:
        1.1.1. Set x.isDataAttribute = true ;
Step 2. For each noun-phrase instance x in the input functional process:
    2.1. If x is a Personal Pronoun (i.e. x.POS=PRP) and x (lowercased) is "he",
        "she", "it" or "they", then:
        2.1.1. If x is not pleonastic [which is detected using Gate's ANNIE tool
            (Cunningham H., Maynard, Bontcheva, & Tablan, 2002)], then:
            Resolve pronominal anaphoric references for x, by identifying the
            nearest matching antecedent y (i.e. that matches the number and the
            gender of x) and copying all the feature values of the antecedent y
            to x ; (Thus, if y.isDataAttribute = true, then also,
            x.isDataAttribute = true.)
Step 3. For each Wh-determiner y in the input functional process d :
    3.1. Set y (temporarily) as a noun-phrase instance ;
    3.2. Identify the nearest matching antecedent x (i.e. that matches the number and
        the gender of y) and copy all the feature values of the antecedent x to
        y ; (Thus, if x.isDataAttribute = true then also, y.isDataAttribute = true.)
```

We used the JAPE scripting language of the GATE environment (Cunningham H., *et al.*, 2011) to implement our above heuristic.

7.5.2 Data-Movement Classification

During the data-movement classification step, we identify the noun-phrases that appear in functional requirements sentences to represent the data-attributes participating in one or more type of data-movements. Our combination of lexical and syntactic features, as mentioned in Section 7.4, already identify verbs within functional requirement sentences that express the senses of *Entry*, *Exit*, *Read* and *Write* types of data-movements, and also the relationships of the noun-phrases to those verbs. We therefore apply the following algorithm that uses these features to classify the type of data-movement that a data-attribute (expressed by a noun-phrase) is involved with:

- ```
Step 1. For each noun-phrase instance x in the input functional process d :
 1.1. If x.F8 = true, then:
 1.1.1. Skip and continue to the next iteration with a new x;
 1.2. If x.F18 = true, then:
 1.2.1. If x.F14 = "actor subject", then:
 Set x.isRelatedToEntryVerb = true ;
 1.2.2. Else If x.F14 = "none/no subject", then:
 Set x.isRelatedToEntryVerb = true ;
 1.3. Else If x.F20 = true, then:
 Set x.isRelatedToExitVerb = true ;
 1.4. Else If x.F22 = true, then:
 Set x.isRelatedToReadVerb = true ;
 1.5. Else If x.F24 = true, then:
 Set x.isRelatedToWriteVerb = true ;

Step 2. For each noun-phrase instance x in the input functional process d :
 2.1. If the head of x is connected to another head of a noun-phrase y with a
 coordinating conjunction, causing a conjunct relation [which is detected
 using the Stanford Dependency Parser (Cer, de Marneffe, Jurafsky, & Manning,
 2010), then:
 2.1.1. Copy all the feature values of x to y ; (Thus, if x.isRelatedToEntryVerb
 = true and/or x.isRelatedToExitVerb = true and/or x.isRelatedToReadVerb =
 true and/or x.isRelatedToWriteVerb = true, then also,
 y.isRelatedToEntryVerb = true and/or y.isRelatedToExitVerb = true and/or
 y.isRelatedToReadVerb = true and/or y.isRelatedToWriteVerb = true
 respectively.)
 2.2. If x is a Wh-determiner, and y is its antecedent, then:
 2.2.1. Copy all the feature values of x to y ; (Thus, if x.isRelatedToEntryVerb
 = true and/or x.isRelatedToExitVerb = true and/or x.isRelatedToReadVerb =
 true and/or x.isRelatedToWriteVerb = true, then also,
 y.isRelatedToEntryVerb = true and/or y.isRelatedToExitVerb = true and/or
```

```

 y.isRelatedToReadVerb = true and/or y.isRelatedToWriteVerb = true
 respectively.)
2.2.2. Remove x from the collection of noun-phrase instances.

```

Here, where a noun-phrase instance is tagged with the feature value indicating that it is related to a type of data-movement verb, it means that the noun-phrase may potentially be counted as a data-movement of a data-attribute based on the semantics of the sentence. However, as the COSMIC standard suggests that a data-movement occurs only at the level of data-groups, the actual classification of data-movement is finalized when the noun-phrases are associated with data-groups using our heuristics, presented in Section 7.7.

Moreover, we found that the measurers often detect implicitly specified occurrences of two special types of data-movements while reading functional requirement sentences with certain verbs. They are: (1) the *Entry*-type data-movements of the *triggering events* of functional processes, and (2) the *Exit*-type data-movements of *System Message* data-group. These two data-groups usually do not appear in the list of possible domain entities of a problem domain, and they cannot also be associated to any noun-phrase of a functional requirement sentence, as no data-attribute can belong to them. Hence, the data-movements can only be tagged to the sentences that express the senses of these data-movements implicitly, as shown in our example in Section 2.7.2.

To detect these two special cases of implicitly specified data-movements, we first built two more vocabularies of verbs<sup>19</sup>: the ones that convey the idea of the implicit occurrence of the *Entry*-type data-movements of the *triggering events*, and the others that do the same for the *Exit*-type data-movement of the *System Messages* data-group. We then used the following additional algorithm to identify these implicitly specified data-movements:

```

Step 1. For each sentence x in the input functional process d :
1.1. For each verb y in x:
1.1.1. If the root of y exists in our vocabulary of "Triggering-Entry" verbs
 (i.e. the verbs that convey the idea of the implicit occurrence of the
 Entry-type data-movements of the triggering events), then:
1.1.1.1. For each noun-phrase instance p, which appears as a subject of y
 in the sentence:
1.1.1.1.1. If the root of the head of p exists in our vocabulary of
 "actor names", then:

```

---

<sup>19</sup> These two additional vocabularies that we built during our tests are presented in Appendices A.6 and A.7.

```

 Set x.classified_TriggeringEntryDataMovement = true ;
1.1.1.1.2. Else if head of p (lowercased and with non-alphabet
 characters removed) contains a substring that is equal to any
 of the strings of in the set {"case", "usecase", "procedure",
 "process", "function", "method", "service", "task"}, then:
 Set x.classified_TriggeringEntryDataMovement = true ;
1.1.1.2. If y has no subject, then:
 Set x.classified_TriggeringEntryDataMovement = true ;
1.1.2. If the root of y exists in our vocabulary of "System-Message-Exit" verbs
 (i.e. the verbs that convey the idea of the implicit occurrence of the
 Exit-type data-movements of the System Message data-group), then:
1.1.2.1. For each noun-phrase instance p, which appears as a subject of y
 in the sentence:
1.1.2.1.1. If the root of the head of p does not exist in our
 vocabulary of "actor names", then:
 Set x.classified_SystemMessageExitDataMovement = true ;
1.1.2.2. If y has no subject, then:
 Set x.classified_SystemMessageExitDataMovement = true ;

```

We again used the JAPE scripting language of the GATE environment (Cunningham H., *et al.*, 2011) to implement all of our above heuristics. It should be mentioned that our feature values that indicated the nominal subjects and the direct objects of verbs were distributed (or copied) amongst the noun-phrase instances and the verbs accordingly, whenever there are conjunctions joining the noun-phrase instances together or the verbs together that result from the occurrences of commas and coordinating conjunction-type words (i.e. and, or, either, neither, nor etc.) between them, as mentioned in Section 7.4.2.

Our above data-movement classification approach depends on many features from our final pool of features. However, we designed our heuristics to use a minimum number of features from our feature pool to keep the rules generalized enough to work with new textual requirements from unseen problem domains and fail gracefully in case of the less-frequent exceptions due to unrestricted natural language are encountered. We, therefore, also included a supervised learning-based data-movement classification approach that can evolve with new training instances and fine tune its learnt models by introducing additional unused features from our feature pool. For example, it may utilize the unused features, like the ones that deal with different kinds of dependent clauses within the sentences (i.e. features F17, F19, F21, F23 and F25, as presented in Section 7.4.3), or the one that detects the Stative verbs and that

subjects within the sentences (i.e. feature F11, as presented in Section 7.4.2) etc, to tackle the exceptions of unrestricted natural language.

In the next section, we discuss our supervised learning-based data-movement classification approach in details.

## 7.6 Supervised Learning-based Classification

During the supervised learning-based classification step, we take the datasets generated by the feature extraction step (discussed in Section 7.4) as input. Here, to use them as training datasets, we also need noun-phrase instances to be annotated by expert measurers, either manually or using our annotation tool LASR. Our supervised learning-based classification approach then applies these datasets to train four different binary classifiers: (1) *Entry* Data-Movement Classifier, (2) *Exit* Data-Movement Classifier, (3) *Write* Data-Movement Classifier, and (4) *Write* Data-Movement Classifier.

### 7.6.1 Feature Selection

We trained each of our data-movement classifiers with a total of nineteen different features from our feature pool. They are:

- **For *Entry* Data-Movement Classification:** The chosen linguistic features are F1-F19, as described in Section 7.4.
- **For *Exit* Data-Movement Classification:** The chosen linguistic features are F1-F17, F20 and F21, as described in Section 7.4.
- **For *Read* Data-Movement Classification:** The chosen linguistic features are F1-F17, F22 and F23, as described in Section 7.4.
- **For *Write* Data-Movement Classification:** The chosen linguistic features are F1-F17, F24 and F25, as described in Section 7.4.

## 7.6.2 Choice of Learning Algorithms

We built all four data-movement classifiers (i.e. the *Entry* classifier, the *Exit* classifier, the *Read* classifier and the *Write* classifier) based on Weka's (Hall, Frank, Holmes, Pfahringer, Reutemann, & Witten, 2009) implementation of the C4.5 decision tree learning algorithm (Quinlan, 1993). The C4.5 decision tree learning algorithm not only attained some of the best results compared to other learning algorithms (e.g. Naïve Bayes or Neural Network) in our preliminary tests, but also generated rules, whose semantics could also be verified for correctness. We fine-tuned its parameters to first build an un-pruned decision tree, and then prune it by applying the restriction of classifying at least 5 instances in each leaf of the final decision tree.

For our experiments with these supervised learning-based data-movement classifiers, we used the annotated corpus, which was generated by the annotation experiments discussed in Section 5.3.3, and built our training/testing datasets by extracting the features, mentioned in Section 7.6.1. After training, these classifiers can take each unlabeled noun-phrase from the textual requirements of a functional process as input, extract the feature values from the noun-phrase and classify it either as the one participating in a corresponding type of data-movement, or as the one not doing so. For example, training our *Entry* data-movement classifier with the dataset extracted from the whole *Entry* corpus, generates a decision tree as shown in Figure 36.

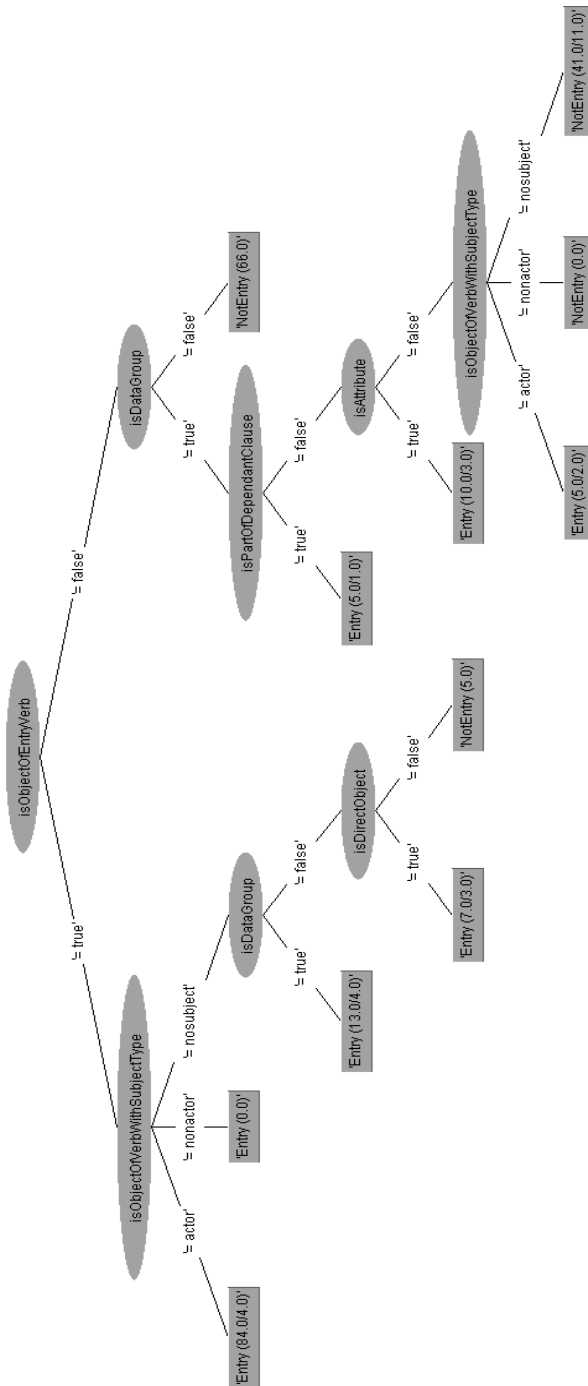


Figure 36: Example Decision Tree to Classify Entry Data-Movement (when the whole dataset is used to train the C4.5 algorithm)

In Figure 36, the feature nodes are labelled with descriptive names instead of using F1, F2, F3 etc. to improve readability. This auto-generated decision tree shows semantically agreeable rules for Entry classification, where none of its branches are misdirected.



Although the tree, shown in Figure 36, is generated by using the complete dataset for training our classifier, we only used portions of our datasets for training, while the rest used for testing, when running our experiments in this phase. The details on how we used these datasets in our experiments for training and testing each of our data-movement classifiers are presented in Section 7.9.

## 7.7 Data-Group Association

In the data-group association step, we associate the data-group names accordingly to the base noun-phrase instances (i.e. to the nouns, the noun compounds and pronouns) that appear in the functional requirement sentences of our input functional process. This step also finalizes the output classification of the data-movements, as discussed in Sections 7.5.2 and 7.6. Here, we use the following algorithm to associate the data-group names to the noun-phrase instances and also finalize their data-movement classification labels based on the associated data-groups:

```

Step 1. For each noun-phrase instance x in the input functional process d :
 1.1. If x.F2 = true and x matches fully/partially to any y of all the data-group
 names, then:
 1.1.1. If x.isRelatedToEntryVerb = true or x.isRelatedToExitVerb = true or
 x.isRelatedToReadVerb = true or x.isRelatedToWriteVerb = true, then:
 1.1.1.1. Set foundMovingDataAttributes = false ;
 1.1.1.2. Set x.classified_AssociatedToDataGroup = y ;
 1.1.1.3. Set d.lastMostLikelyKnownDataGroup = y ;
 1.1.1.4. If x.isADataAttribute = true, then:
 1.1.1.4.1. Set x.classified_DataAttribute = true ;
 1.1.1.4.2. Set foundMovingDataAttributes = true ;
 1.1.1.5. Else:
 1.1.1.5.1. Set x.classified_DataAttribute = false ;
 1.1.1.5.2. Set foundMovingDataAttributes = false ;
 1.1.1.6. If x.F9 = true and x owns a noun-phrase instance z (the feature of
 owning a noun-phrase instance is presented in F9 in Section 7.5.2)
 and z.isADataAttribute = true and foundMovingDataAttributes = false,
 then:
 1.1.1.6.1. Set foundMovingDataAttributes = true ;
 1.1.1.6.2. Set z.classified_AssociatedToDataGroup = y ;
 1.1.1.6.3. Set z.classified_DataAttribute = true ;
 1.1.1.7. If x.F12 = true and foundMovingDataAttributes = false and x is
 related to z by a chain of prepositional objects (the feature of
 being related to a noun-phrase instance by a chain of prepositional

```

objects is presented in F12 in Section 7.5.2), such that *z.isADataAttribute = true*, then:

- 1.1.1.7.1. Set *foundMovingDataAttributes = true* ;
- 1.1.1.7.2. Set *z.classified\_AssociatedToDataGroup = y* ;
- 1.1.1.7.3. Set *z.classified\_DataAttribute = true* ;
- 1.1.1.8. If *foundMovingDataAttributes = false*, then:
  - 1.1.1.8.1. Set *x.classified\_DataAttribute = true* ;
  - 1.1.1.8.2. Set *foundMovingDataAttributes = true* ;
- 1.1.2. Else:
  - 1.1.2.1. If *x* is not a subject to a data-movement verb, then:
    - 1.1.2.1.1. Set *d.lastMostLikelyKnownDataGroup = y* ;
  - 1.1.2.2. Else:
    - 1.1.2.2.1. Set *d.lastLessLikelyKnownDataGroup = y* ;
- 1.2. Else:
  - 1.2.1. If *x.isRelatedToEntryVerb = true* or *x.isRelatedToExitVerb = true* or *x.isRelatedToReadVerb = true* or *x.isRelatedToWriteVerb = true*, then:
    - 1.2.1.1. Set *foundMovingDataAttributes = false* ;
    - 1.2.1.2. Set *foundDataGroupName = false* ;
    - 1.2.1.3. If *x.isADataAttribute = true*, then:
      - 1.2.1.3.1. Set *x.classified\_DataAttribute = true* ;
      - 1.2.1.3.2. Set *foundMovingDataAttributes = true* ;
      - 1.2.1.3.3. Set *mostLikelyNewDataGroup = null* ;
      - 1.2.1.3.4. Set *lessLikelyNewDataGroup = null* ;
      - 1.2.1.3.5. If *x.F10 = true* and *x* belongs to a noun-phrase instance *z* (the feature of belonging to a noun-phrase instance is presented in F10 in Section 7.5.2), then:
        - 1.2.1.3.5.1. Set *mostLikelyNewDataGroup = headOf(z)* ;
        - 1.2.1.3.5.2. If *z* matches fully/partially to any *y* of all the names in our vocabulary of data-group names and *foundDataGroupName = false*, then:
          - 1.2.1.3.5.2.1. Set *x.classified\_DataAttribute = true* ;
          - 1.2.1.3.5.2.2. Set *x.classified\_AssociatedToDataGroup = y* ;
          - 1.2.1.3.5.2.3. Set *d.lastMostLikelyKnownDataGroup = y* ;
          - 1.2.1.3.5.2.4. Set *foundDataGroupName = true* ;
      - 1.2.1.3.6. If *x.F13 = true* and *foundDataGroupName = false*, then:
        - 1.2.1.3.6.1. For each noun-phrase instance *z* that is related to *x* by a chain of prepositional objects (the feature of being related to a noun-phrase instance by a chain of prepositional objects is presented in F13 in Section 7.5.2):
          - 1.2.1.3.6.1.1. Set *lessLikelyNewDataGroup = headOf(z)* ;
          - 1.2.1.3.6.1.2. If *z* matches fully/partially to any *y* of all the names in our vocabulary of data-group names, then:
            - 1.2.1.3.6.1.2.1. Set *x.classified\_DataAttribute = true* ;
            - 1.2.1.3.6.1.2.2. Set *x.classified\_AssociatedToDataGroup = y* ;
            - 1.2.1.3.6.1.2.3. Set *d.lastMostLikelyKnownDataGroup = y* ;

```

1.2.1.3.6.1.2.4. Set foundDataGroupName = true ;
1.2.1.3.6.1.2.5. Break out of the for loop;
1.2.1.3.6.1.2.6. Set foundDataGroupName = true ;
1.2.1.3.7. If foundMovingDataAttributes = false, then:
1.2.1.3.7.1. If d.lastMostLikelyKnownDataGroup is not null, then:
1.2.1.3.7.1.1. Set y = d.lastMostLikelyKnownDataGroup ;
1.2.1.3.7.1.2. Set x.classified_DataAttribute = true ;
1.2.1.3.7.1.3. Set x.classified_AssociatedToDataGroup = y ;
1.2.1.3.7.1.4. Set foundDataGroupName = true ;
1.2.1.3.7.2. Else If mostLikelyNewDataGroup is not null, then:
1.2.1.3.7.2.1. Set y = mostLikelyNewDataGroup ;
1.2.1.3.7.2.2. Set x.classified_DataAttribute = true ;
1.2.1.3.7.2.3. Set x.classified_AssociatedToDataGroup = y ;
1.2.1.3.7.2.4. Set foundDataGroupName = true ;
1.2.1.3.7.3. Else If lessLikelyNewDataGroup is not null, then:
1.2.1.3.7.3.1. Set y = lessLikelyNewDataGroup ;
1.2.1.3.7.3.2. Set x.classified_DataAttribute = true ;
1.2.1.3.7.3.3. Set x.classified_AssociatedToDataGroup = y ;
1.2.1.3.7.3.4. Set foundDataGroupName = true ;
1.2.1.3.7.4. Else:
1.2.1.3.7.4.1. Set x.classified_AssociatedToDataGroup = unknown ;
1.2.1.3.7.4.2. Set x.classified_DataAttribute = true ;

```

All of our above heuristics were implemented using the Jape scripting language of the GATE environment (Cunningham H., *et al.*, 2011). Our heuristics to associate data-group names to noun-phrases (as discussed in details in Section 7.7), uses multiple passes of loops that are sequentially executed, and the loops depend on the outcomes of their previous passes. Based on these heuristics, we also find that process of data-group association is strictly an algorithmic approach that do not fully depend on our pool of features, but on the outputs generated by our previous step of data-movement classification (as discussed in Sections 7.5.2 and 7.6). Thus, the rules we set are complete enough deal with any problem domain, and do not need to evolve over time with the inclusion of new textual instances of software requirements from unseen problem domains.

## 7.8 Extending COSMIC by CFP Range Measurement

In Section 4.3, we presented our formalization of the COSMIC functional size measurement process (COSMIC, 2014) by modeling of the ontology for the COSMIC FSM artifacts and the formulas for assigning the numeric value of functional size to a piece of software. Our

approaches discussed in this chapter automates the extraction of the COSMIC FSM modeling artifacts from the textual requirements, and thus maintains the traceability of the FSM process (i.e. from input textual requirements to the output measurement) by instantiating the classes of our ontology.

Now, using our approaches as described in Sections 7.5, 7.6 and 7.7, we can automatically identify the following sets of classification labels for the noun-phrases and sentences from the textual requirements that describe a functional process,  $FPROC_i$  :

- Set *A* containing labels for noun-phrases, that identify them as mentions of *Data-Attributes*, which participate in at least one of the four specific types of *Data-Movements*, and that most likely belong to some already known *Data-Group(s)*.
- Set *B* containing labels for noun-phrases, that identify them as mentions of *Data-Attributes*, which participate in at least one of the four specific types of *Data-Movements*, and that less likely belong to some already known *Data-Group(s)*.
- Set *C* containing labels for noun-phrases, that identify them as mentions of *Data-Attributes*, which participate in at least one of the four specific types of *Data-Movements*, and that most likely belong to some new and previously unknown *Data-Group(s)*.
- Set *D* containing labels for noun-phrases, that identify them as mentions of *Data-Attributes*, which participate in at least one of the four specific types of *Data-Movements*, and that less likely belong to some new and previously unknown *Data-Group(s)*.
- Set *E* containing labels for noun-phrases, that identify them as mentions of *Data-Attributes*, which participate in at least one of the four specific types of *Data-Movements*, and that belong to some unknown *Data-Group(s)*.

- Set  $F$  containing labels for sentences, that identify them as implicit expressions of the *Entry*-type data-movement of the *Triggering Events*.
- Set  $G$  containing labels for sentences, that identify them as implicit expressions of the *Exit*-type data-movement of the *System Messages Data Group*.

Thus, each classification label as an element of the sets  $A, B, C, D$  and  $E$  can be designated as the following pair:

$$(data\text{-}group_x, data\text{-}movement\text{-}type_y)$$

where,  $data\text{-}group_x \in \text{Set of } m \text{ possible data-groups in the input problem domain, } p$   
 $data\text{-}movement\text{-}type_y \in \{ Entry, Exit, Read, Write \}$

And, the classification labels as elements of the sets  $F$  and  $G$  can be designated as the pairs (*TriggeringEvent*, *Entry*) and (*SystemMessage*, *Exit*) respectively.

Thus, each element of the sets  $A, B, C, D, E, F$  and  $G$  is actually an FSM object, as defined in Section 4.3, and therefore can be any of the following pairs:

$(data\text{-}group_1, Entry), (data\text{-}group_1, Exit), (data\text{-}group_1, Read), (data\text{-}group_1, Write), (data\text{-}group_2, Entry), \dots (data\text{-}group_m, Entry), (data\text{-}group_m, Exit), (data\text{-}group_m, Read), (data\text{-}group_m, Write)$

Our experiments that are described later in Sections 7.9, 7.10 and 7.11, produced classification labels similar to the above *FSM Objects*, and thus can also be considered as elements of the sets  $A, B, C, D, E, F$  and  $G$ .

Now, according to the equation (2) presented in Section 4.3 of Chapter 4, we know that the aggregated measure of functional size in CFP of  $FPROC_i$  is—

$$FunctionalSize(FPROC_i) = \|FSMObjects(FPROC_i)\|$$

We now use the above formula to extend CFP measurement process into measuring the range of CFP, by defining the minimum, maximum and most-likely functional size. These definitions are shown by the following formulas:

$$FunctionalSize_{\min}(FPROC_i) = \max(2, \|A\|)$$

$$FunctionalSize_{\text{most-likely}}(FPROC_i) = \max(2, \|A \cup C \cup F \cup G\|)$$

$$FunctionalSize_{\max}(FPROC_i) = \max(2, \|A \cup B \cup C \cup D \cup E \cup F \cup G\|)$$

These formulas ensure the ranges of functional size in CFP for a functional process, such that,  $2 \leq FunctionalSize_{\min}(FROC_i) \leq FunctionalSize_{\text{most-likely}}(FROC_i) \leq FunctionalSize_{\max}(FROC_i)$ , and thus, retain the rule of COSMIC FSM standard (COSMIC, 2014), which states that the minimum possible size of a functional process is 2 CFP. Also, the set  $F$  is optionally considered in these formulas; i.e. it can be omitted from these formulas, in case triggering events are not to be counted as additional Entry-type data-movements.

Thus, using the above formulas for  $FunctionalSize_{\min}(FROC_i)$ ,  $FunctionalSize_{\text{most-likely}}(FROC_i)$  and  $FunctionalSize_{\max}(FROC_i)$ , we can extend the equation (3), presented in Section 4.3, to measure the CFP range of a requirements document,  $DOCUMENT_p$ , as follows:

$$FunctionalSize_x(DOCUMENT_p) = \sum_{i=1}^n \|FSMObjects_x(FPROC_i)\|$$

where  $x$  can be either “min” or “most-likely” or “max”, referring to the minimum, most-likely and maximum functional sizes respectively defining the limits of the CFP ranges. The measured CFP ranges of the input requirements documents can thus be used to estimate the development effort in range too. We thus used the above formulas to derive the CFP ranges from the classification labels produced by our experiments, which are described next in Sections 7.9, 7.10 and 7.11. We then analyzed the results in Section 7.12.

## 7.9 Overview of the Classification Experiments

As described in Sections 7.5 and 7.6, we developed two different approaches for identifying the types of data-movements that a noun-phrase participates in: (1) a heuristic-based approach, that is to be applied in absence of an annotated dataset, and (2) a supervised learning-based approach, that is to be applied to dynamically adapt itself to evolving environments by training it with annotated datasets. In this section, we present an overview of the experiments that we conducted to validate both of our classification approaches.

### 7.9.1 The Corpora

The true gold-standard annotation labels of each base noun-phrase instance, as discussed in Section 5.3.3, formed our annotated corpora that are to be used for our experiments. Thus, we

experimented with our data-movement classification approaches with highly skewed corpora, as presented by the distributions of the gold-standard class labels shown in Figure 37.

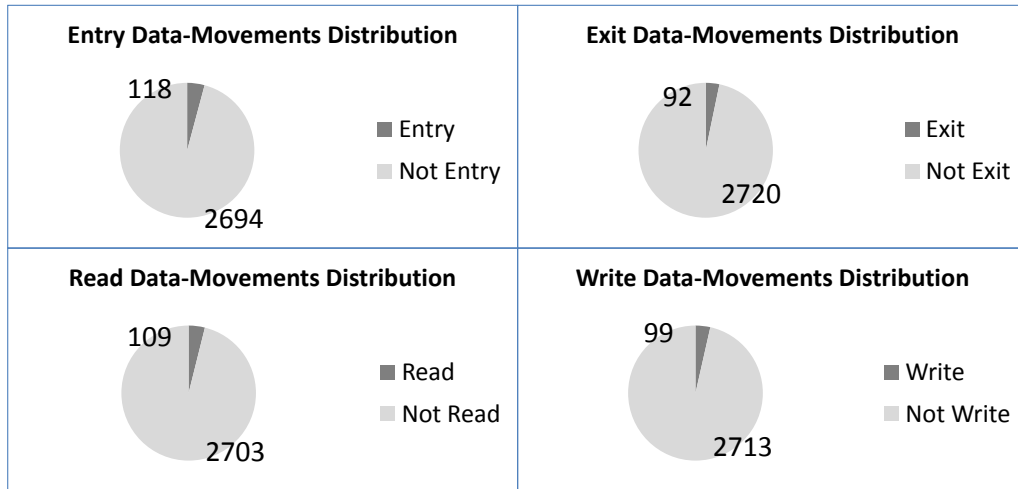


Figure 37: Distribution of the Gold-Standard Class Labels for Data-Movement Classification

Table 26 shows further details of the corpora distribution by presenting the number of instances, their sources and their corresponding class labels, which are: Entry (E+), Not Entry (E-), Exit(X+), Not Exit (X-), Read (R+), Not Read (R-), Write (W+) and Not Write (W-).

| Doc Id | Document Name              | Source               | Extracted Base NP's | Entry Classification |      | Exit Classification |      | Read Classification |      | Write Classification |      |
|--------|----------------------------|----------------------|---------------------|----------------------|------|---------------------|------|---------------------|------|----------------------|------|
|        |                            |                      |                     | E+                   | E-   | X+                  | X-   | R+                  | R-   | W+                   | W-   |
| C1     | (undisclosed)              | Industry (SAP)       | 314                 | 14                   | 300  | 32                  | 282  | 27                  | 287  | 6                    | 308  |
| C2     | (undisclosed)              | Industry (SAP)       | 59                  | 4                    | 55   | 3                   | 56   | 2                   | 57   | 1                    | 58   |
| C3     | Course Registration System | Concordia University | 711                 | 22                   | 689  | 16                  | 695  | 19                  | 692  | 24                   | 687  |
| C4     | IEEE Montreal Website      | Concordia University | 1318                | 31                   | 1287 | 16                  | 1302 | 17                  | 1301 | 26                   | 1292 |
| C5     | (undisclosed)              | Industry (SAP)       | 27                  | 2                    | 25   | 3                   | 24   | 4                   | 23   | 0                    | 27   |
| C6     | (undisclosed)              | Industry (SAP)       | 383                 | 45                   | 338  | 22                  | 361  | 40                  | 343  | 42                   | 341  |
| Total  |                            |                      | 2812                | 118                  | 2694 | 92                  | 2720 | 109                 | 2703 | 99                   | 2713 |

Table 26: Frequency of Data-movement Class Labels in the Corpora

The corpora held multi-class annotation data, i.e. the same noun-phrase instance could be annotated for Entry classification (as E+ or E-), Exit classification (as X+ or X-), Read classification (as R+ or R-), and Write classification (as W+ or W-). We therefore divided the corpora into four different binary corpora: (1) the Entry classification corpus, (2) the Exit classification corpus, (3) the Read classification corpus, and (4) the Write classification

corpus. We used these annotated corpora to experiment with all of our classification approaches.

## 7.9.2 Testing Methods

Since we have a very small number of positive instances in our corpora, we applied three different types of testing methods to perform detailed and conclusive analyses for our experiments over our classification approaches. These three types of testing methods are:

### (1) Using the Complete Dataset for Testing

We used the complete dataset to test our heuristic-based classification approach, presented in Section 7.5. Since we use a heuristic in this approach to identify the noun-phrases that does not depend on the availability of the training set, we could use our complete dataset here for testing.

### (2) Batch Testing for Incremental Learning

We applied batch testing method to test our supervised learning-based classification approach, presented in Section 7.6. Since our classifiers here need to be trained first with an almost evenly distributed training dataset before testing, we ran five different batches of training-testing trials, where, in each batch, we gradually increased the number of documents used for training our classifiers, while the rest of the documents are used for testing. Our testing algorithm carefully constructed each trial within a batch so that none of the documents used for training a classifier in one trial, is not used for testing the classifier in the same trial. The trials are then composed of all possible combinations of documents that we could use from our corpus for training and testing each of our classifier. Our testing algorithm here also made sure that the extracted training dataset for each trial within a batch contains equal number of positive and negative instances. Finally, we then average the test results of all the trials in each batch to analyze the results across all the batches.

As shown earlier in Table 26, the documents of our corpora has variable numbers of noun-phrase instances. Therefore, the number of training and testing instances in each trial of our batches also varied. Thus, Table 27 shows, for example, the total number of noun-phrase instances from our *Entry* corpus and their source documents that are used for training and



testing our *Entry* classifier during each trail of batch #1. A total of 6 trials are executed in this batch, each using one different document for training our classifier, and a different combination of the remaining five documents for testing our classifier.

| Batch no. | Trial no. | Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances |
|-----------|-----------|----------------------|---------------------|--------------------------|-------------------------|
| 1         | 1.1       | C1                   | C2,C3,C4,C5,C6      | 28                       | 2498                    |
|           | 1.2       | C2                   | C1,C3,C4,C5,C6      | 8                        | 2753                    |
|           | 1.3       | C3                   | C1,C2,C4,C5,C6      | 44                       | 2101                    |
|           | 1.4       | C4                   | C1,C2,C3,C5,C6      | 62                       | 1494                    |
|           | 1.5       | C5                   | C1,C2,C3,C4,C6      | 4                        | 2785                    |
|           | 1.6       | C6                   | C1,C2,C3,C4,C5      | 90                       | 2429                    |

Table 27: Training and Testing Instances in Batch Test #1 for the *Entry* Classifier

Similarly, Table 28 shows, for example, the total number of noun-phrase instances and their source documents that are used for training and testing our *Entry* classifier during each trail of batch #2. A total of 15 trials are executed in this batch, each using a different combination of two documents for training our classifier, and a different combination of the remaining four documents for testing our classifier.

| Batch no. | Trial no. | Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances |
|-----------|-----------|----------------------|---------------------|--------------------------|-------------------------|
| 2         | 2.1       | C1,C2                | C3,C4,C5,C6         | 36                       | 2439                    |
|           | 2.2       | C1,C3                | C2,C4,C5,C6         | 72                       | 1787                    |
|           | 2.3       | C1,C4                | C2,C3,C5,C6         | 90                       | 1180                    |
|           | 2.4       | C1,C5                | C2,C3,C4,C6         | 32                       | 2471                    |
|           | 2.5       | C1,C6                | C2,C3,C4,C5         | 118                      | 2115                    |
|           | 2.6       | C2,C3                | C1,C4,C5,C6         | 52                       | 2042                    |
|           | 2.7       | C2,C4                | C1,C3,C5,C6         | 70                       | 1435                    |
|           | 2.8       | C2,C5                | C1,C3,C4,C6         | 12                       | 2726                    |
|           | 2.9       | C2,C6                | C1,C3,C4,C5         | 98                       | 2370                    |
|           | 2.10      | C3,C4                | C1,C2,C5,C6         | 106                      | 783                     |
|           | 2.11      | C3,C5                | C1,C2,C4,C6         | 48                       | 2074                    |
|           | 2.12      | C3,C6                | C1,C2,C4,C5         | 134                      | 1718                    |
|           | 2.13      | C4,C5                | C1,C2,C3,C6         | 66                       | 1467                    |
|           | 2.14      | C4,C6                | C1,C2,C3,C5         | 152                      | 1111                    |
|           | 2.15      | C5,C6                | C1,C2,C3,C4         | 94                       | 2402                    |

Table 28: Training and Testing Instances in Batch Test #2 for the *Entry* Classifier

Again, Table 29 shows, for example, the total number of noun-phrase instances and their source documents that are used for training and testing our *Entry* classifier during each trail of batch #3. A total of 20 trials are executed in this batch, each using a different combination of three documents for training our classifier, and a different combination of the remaining three documents for testing our classifier.

| Batch no. | Trial no. | Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances |
|-----------|-----------|----------------------|---------------------|--------------------------|-------------------------|
| 3         | 3.1       | C1,C2,C3             | C4,C5,C6            | 80                       | 1728                    |
|           | 3.2       | C1,C2,C4             | C3,C5,C6            | 98                       | 1121                    |
|           | 3.3       | C1,C2,C5             | C3,C4,C6            | 40                       | 2412                    |
|           | 3.4       | C1,C2,C6             | C3,C4,C5            | 126                      | 2056                    |
|           | 3.5       | C1,C3,C4             | C2,C5,C6            | 134                      | 469                     |
|           | 3.6       | C1,C3,C5             | C2,C4,C6            | 76                       | 1760                    |
|           | 3.7       | C1,C3,C6             | C2,C4,C5            | 162                      | 1404                    |
|           | 3.8       | C1,C4,C5             | C2,C3,C6            | 94                       | 1153                    |
|           | 3.9       | C1,C4,C6             | C2,C3,C5            | 180                      | 797                     |
|           | 3.10      | C1,C5,C6             | C2,C3,C4            | 122                      | 2088                    |
|           | 3.11      | C2,C3,C4             | C1,C5,C6            | 114                      | 724                     |
|           | 3.12      | C2,C3,C5             | C1,C4,C6            | 56                       | 2015                    |
|           | 3.13      | C2,C3,C6             | C1,C4,C5            | 142                      | 1659                    |
|           | 3.14      | C2,C4,C5             | C1,C3,C6            | 74                       | 1408                    |
|           | 3.15      | C2,C4,C6             | C1,C3,C5            | 160                      | 1052                    |
|           | 3.16      | C2,C5,C6             | C1,C3,C4            | 102                      | 2343                    |
|           | 3.17      | C3,C4,C5             | C1,C2,C6            | 110                      | 756                     |
|           | 3.18      | C3,C4,C6             | C1,C2,C5            | 196                      | 400                     |
|           | 3.19      | C3,C5,C6             | C1,C2,C4            | 138                      | 1691                    |
|           | 3.20      | C4,C5,C6             | C1,C2,C3            | 156                      | 1084                    |

Table 29: Training and Testing Instances in Batch Test #3 for the *Entry* Classifier

Then, Table 30 shows, for example, the total number of noun-phrase instances and their source documents that are used for training and testing our *Entry* classifier during each trail of batch #4. A total of 15 trials are executed in this batch, each using a different combination of four documents for training our classifier, and a different combination of the remaining two documents for testing our classifier.

| Batch no. | Trial no. | Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances |
|-----------|-----------|----------------------|---------------------|--------------------------|-------------------------|
| 4         | 4.1       | C1,C2,C3,C4          | C5,C6               | 142                      | 410                     |
|           | 4.2       | C1,C2,C3,C5          | C4,C6               | 84                       | 1701                    |
|           | 4.3       | C1,C2,C3,C6          | C4,C5               | 170                      | 1345                    |
|           | 4.4       | C1,C2,C4,C5          | C3,C6               | 102                      | 1094                    |
|           | 4.5       | C1,C2,C4,C6          | C3,C5               | 188                      | 738                     |
|           | 4.6       | C1,C2,C5,C6          | C3,C4               | 130                      | 2029                    |
|           | 4.7       | C1,C3,C4,C5          | C2,C6               | 138                      | 442                     |
|           | 4.8       | C1,C3,C4,C6          | C2,C5               | 224                      | 86                      |
|           | 4.9       | C1,C3,C5,C6          | C2,C4               | 166                      | 1377                    |
|           | 4.10      | C1,C4,C5,C6          | C2,C3               | 184                      | 770                     |
|           | 4.11      | C2,C3,C4,C5          | C1,C6               | 118                      | 697                     |
|           | 4.12      | C2,C3,C4,C6          | C1,C5               | 204                      | 341                     |
|           | 4.13      | C2,C3,C5,C6          | C1,C4               | 146                      | 1632                    |
|           | 4.14      | C2,C4,C5,C6          | C1,C3               | 164                      | 1025                    |
|           | 4.15      | C3,C4,C5,C6          | C1,C2               | 200                      | 373                     |

Table 30: Training and Testing Instances in Batch Test #4 for the *Entry* Classifier

Table 31 shows, for example, the total number of noun-phrase instances and their source documents that are used for training and testing our *Entry* classifier during each trial of batch #2. A total of 6 trials are executed in this batch, each using a different combination of five documents for training and the remaining one document for testing our classifier.

| Batch no. | Trial no. | Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances |
|-----------|-----------|----------------------|---------------------|--------------------------|-------------------------|
| 5         | 5.1       | C1,C2,C3,C4,C5       | C6                  | 146                      | 383                     |
|           | 5.2       | C1,C2,C3,C4,C6       | C5                  | 232                      | 27                      |
|           | 5.3       | C1,C2,C3,C5,C6       | C4                  | 174                      | 1318                    |
|           | 5.4       | C1,C2,C4,C5,C6       | C3                  | 192                      | 711                     |
|           | 5.5       | C1,C3,C4,C5,C6       | C2                  | 228                      | 59                      |
|           | 5.6       | C2,C3,C4,C5,C6       | C1                  | 208                      | 314                     |

Table 31: Training and Testing Instances in Batch Test #5 for the *Entry* Classifier

These tables show that a large number of negative instances are randomly removed from the training dataset of each trial so that the number of positive and negative instances in the training datasets remain equal. However, the total number of training instances increases on average across every subsequent batch. This result is shown in Figure 38.

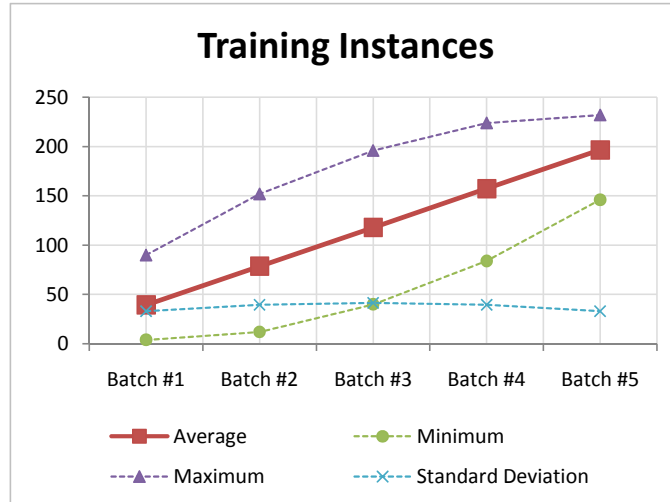


Figure 38: Number of Training Instances Used in Each Batch Test for the *Entry* Classifier

Through our batch testing experiments, we have shown that our data-movement classifier is feasible for practical implementation (i.e. where sufficient number of training instances is available), if its average accuracy results also improve across each subsequent batch.

### (3) Using 10-fold Cross-Validation

We also apply 10-fold cross-validation to test our supervised learning-based classification approach, presented in Section 7.6. Since our classifiers need to be trained with almost evenly distributed datasets before testing, we first randomly removed a portion negative instances in each of our datasets, so that the number of positive and negative instances are equal. We then performed 10-fold cross-validation tests that divides the dataset into 10 equal sized parts (i.e. almost equal number of instances), and then take the average results of 10 different trials, where each trial uses each combination of nine different parts of the dataset to train our classifier and uses the remaining one part to test the classifier.

## 7.9.3 Types of Classification Experiments

Based on the testing methods we used, as discussed in Section 7.9.2, all of the experiments in this phase of the research can be classified into the types, as shown in Figure 39.

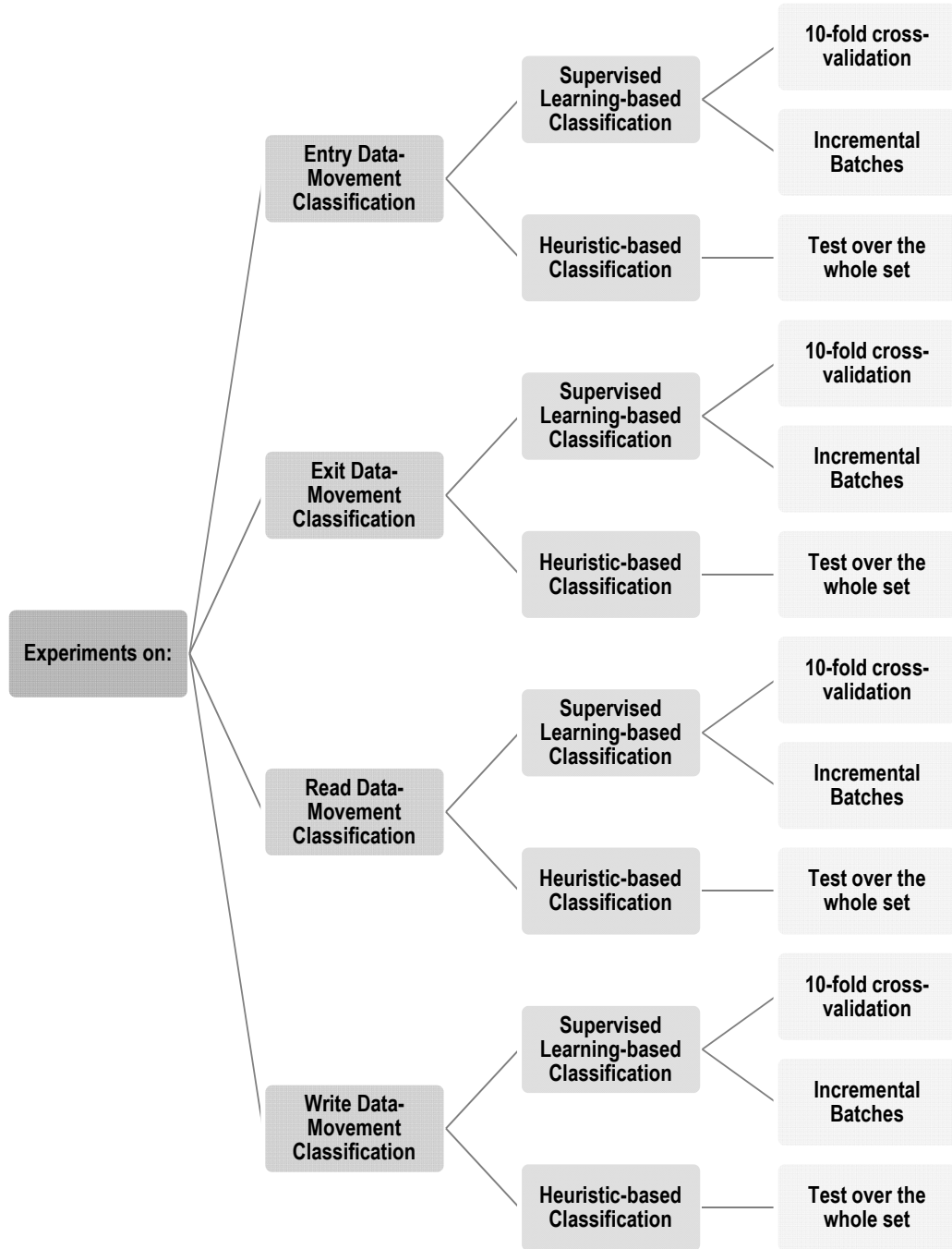


Figure 39: Types of Experiments to Validate the Different Data-Movement Classification Approaches

As we conducted these different types of experiments, shown in Figure 39, we gathered various types of results. We will be discussing these results of our experiment and analyze them in details in the following sections.

## 7.10 Heuristic-based Classification Results

In this section, we discuss and analyze the results of the experiments that we executed to verify our heuristic-based approach, presented in Section 7.5, to classify the noun-phrase instances of different types of data-movements. Table 32 shows the resultant confusion matrices when we tested our heuristic-based classification approach over all instances of our corpora, which is described in Section 7.9.1.

|               |           | Classified As |           |
|---------------|-----------|---------------|-----------|
|               |           | Entry         | Not Entry |
| Gold-Standard | Entry     | 96            | 22        |
|               | Not Entry | 37            | 2657      |

|               |          | Classified As |          |
|---------------|----------|---------------|----------|
|               |          | Read          | Not Read |
| Gold-Standard | Read     | 76            | 33       |
|               | Not Read | 41            | 2662     |

|               |          | Classified As |          |
|---------------|----------|---------------|----------|
|               |          | Exit          | Not Exit |
| Gold-Standard | Exit     | 78            | 14       |
|               | Not Exit | 27            | 2693     |

|               |           | Classified As |           |
|---------------|-----------|---------------|-----------|
|               |           | Write         | Not Write |
| Gold-Standard | Write     | 82            | 17        |
|               | Not Write | 39            | 2674      |

Table 32: Confusion Matrices for Heuristic-based Data-Movement Classification

Using the confusion matrices in Table 32, we can compute the results for this experiment, as shown in Table 33.

| (For Positive Class of) Data-Movement Type | Accuracy (Ratio of Correct) | Kappa | Precision | Recall | F-Measure |
|--------------------------------------------|-----------------------------|-------|-----------|--------|-----------|
| Entry                                      | 0.979                       | 0.754 | 0.722     | 0.813  | 0.765     |
| Exit                                       | 0.985                       | 0.784 | 0.743     | 0.848  | 0.792     |
| Read                                       | 0.974                       | 0.659 | 0.650     | 0.697  | 0.672     |
| Write                                      | 0.980                       | 0.735 | 0.678     | 0.828  | 0.745     |
| <b>Average =</b>                           | 0.979                       | 0.733 | 0.698     | 0.797  | 0.744     |

Table 33: Results of Heuristic-based Data-Movement Classification

Thus, we find in this experiment, as shown in Table 33, that the average accuracy of our heuristic-based approach is very high and is comparable to human performance and the average agreement of its classifications with the true gold-standard labels set by the expert [shown in terms of Cohen's Kappa (Cohen, 1960) in Table 33] is better than the average pairwise agreement of fully-trained human annotators with the same expert, as recorded by the results of our experiments, presented in Section 5.4.2. This shows that our heuristic-based approach can feasibly be applied in practice to identify data-movement types from textual requirements, especially when annotated datasets are not available for training the supervised learning-based classifiers.

## **7.11 Supervised Learning-based Classification Results**

The supervised learning-based data-movement classification approach uses four different classifiers, as mentioned in Section 7.6. Now, to experiment with each of these classifiers, we applied two different testing methods: (1) incremental learning and testing by batches, and (2) 10-fold cross-validation, as described in Section 7.9.2. In the following sections, we discuss and analyze the results of these experiments.

### **7.11.1 Results of Batch Testing for Incremental Learning**

We used our batch testing wrapper scripts, to automate a large number of trials that are grouped into five different batches. Each subsequent batch gradually adds a document from our corpus to the training dataset and uses the rest of the documents for testing. The trials in a batch uses all possible combination of the documents, while never using a document for testing which is already used for training. The training datasets for all trials are also automatically down-sampled, by randomly removing a portion of negative instances, so that the number of the positive and the negative instances are equal in each of these datasets. The procedure of the batch testing experiment is described with examples in Section 7.9.2.

We ran the batch testing experiments individually over each of our data-movement classifiers, described in Section 7.6, using their respective datasets. All these tests were run using our wrapper script over Weka's training and testing modules. The detailed outputs produced by

our script for these batch testing experiments are included in Appendix B.2. We present the summary of these results in Figures 40, 41, 42 and 43.

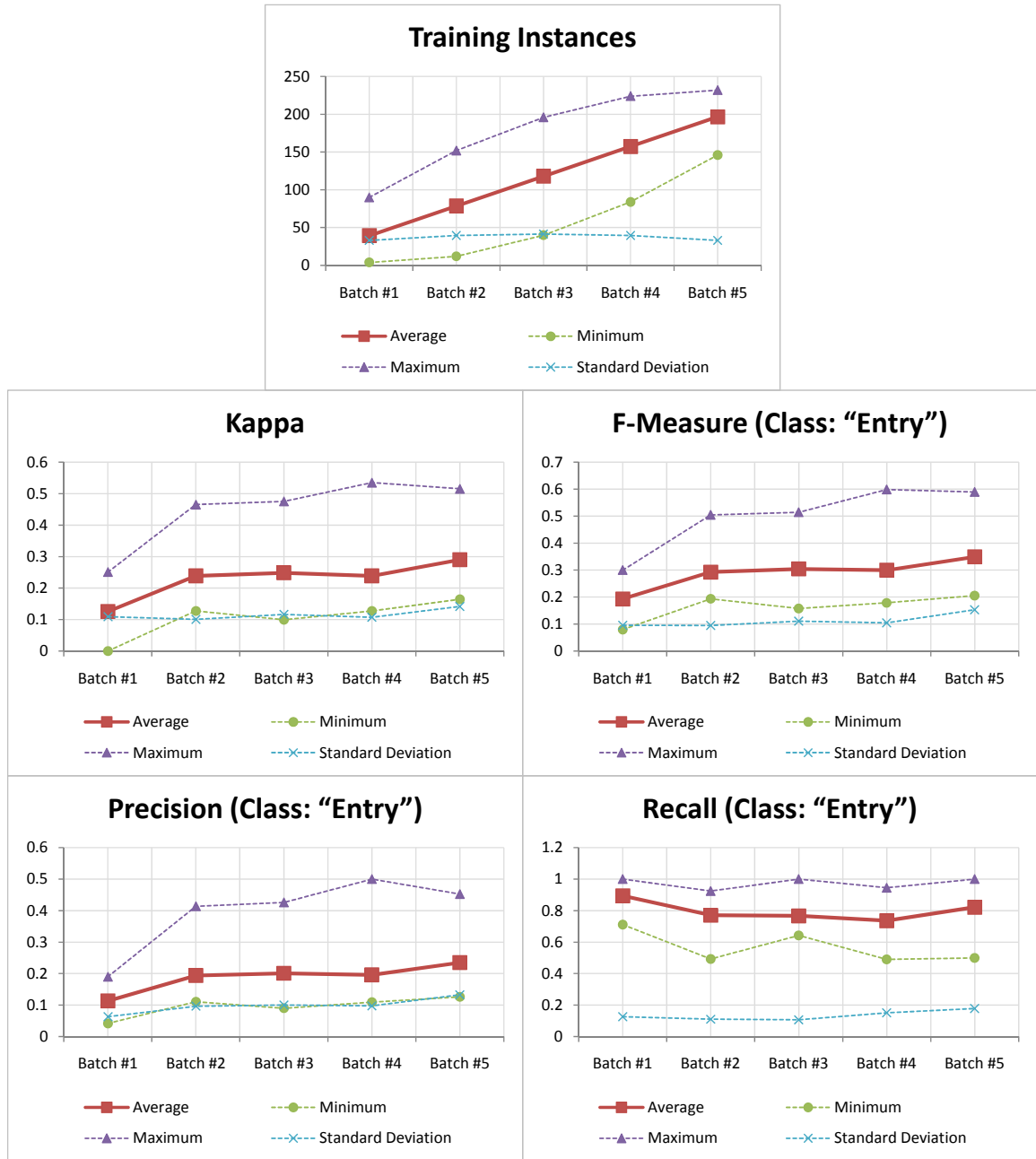


Figure 40: Results of Running Batch Tests for Incremental Learning Over the Entry Classifier



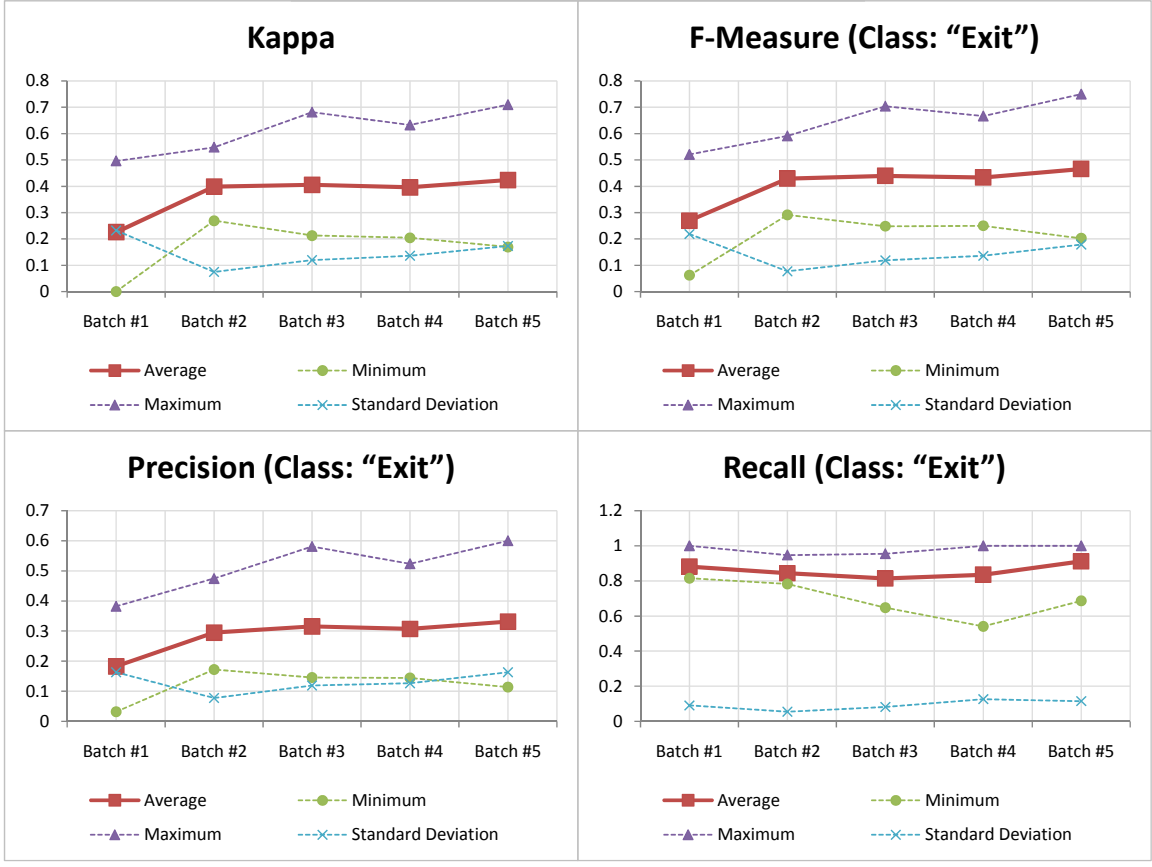
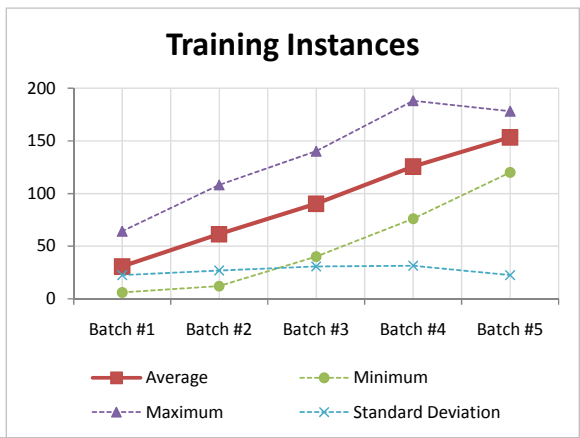


Figure 41: Results of Running Batch Tests for Incremental Learning Over the Exit Classifier

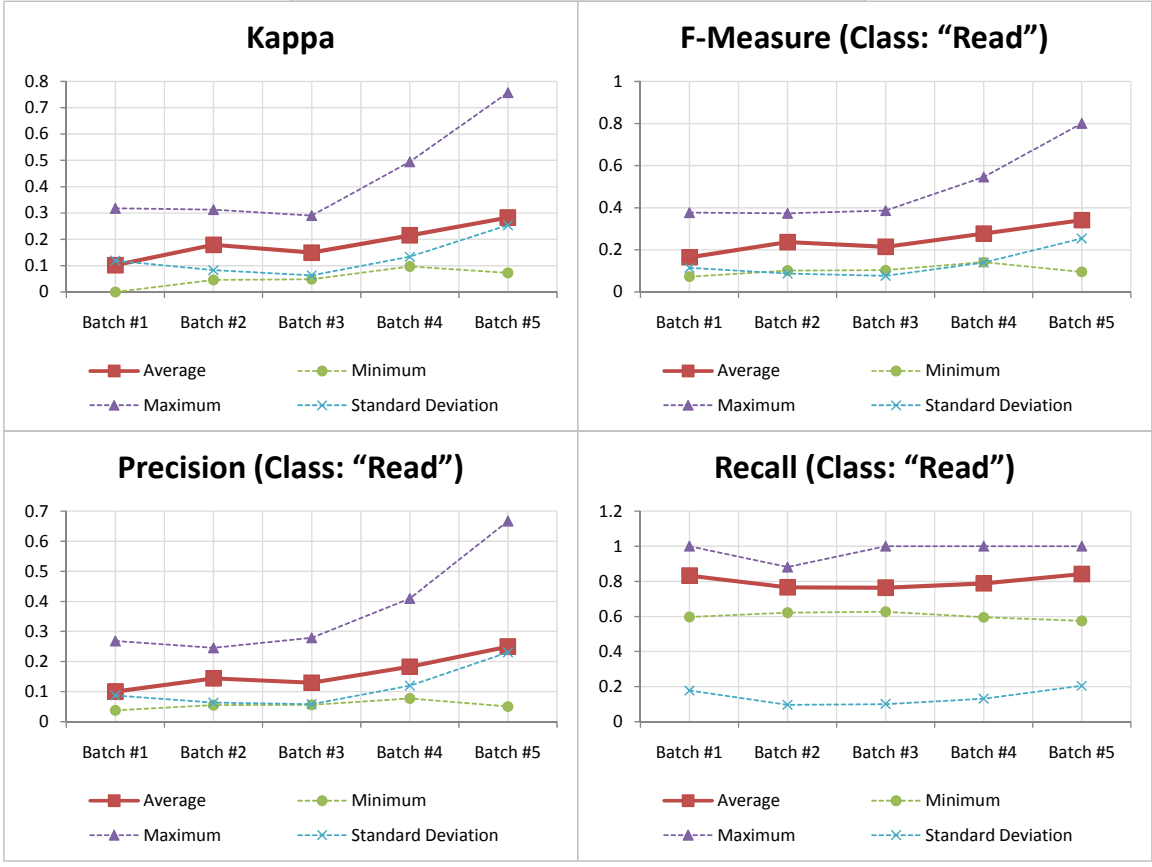
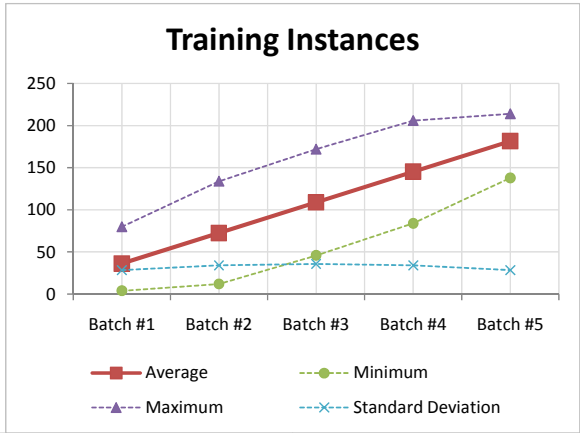


Figure 42: Results of Running Batch Tests for Incremental Learning Over the Read Classifier

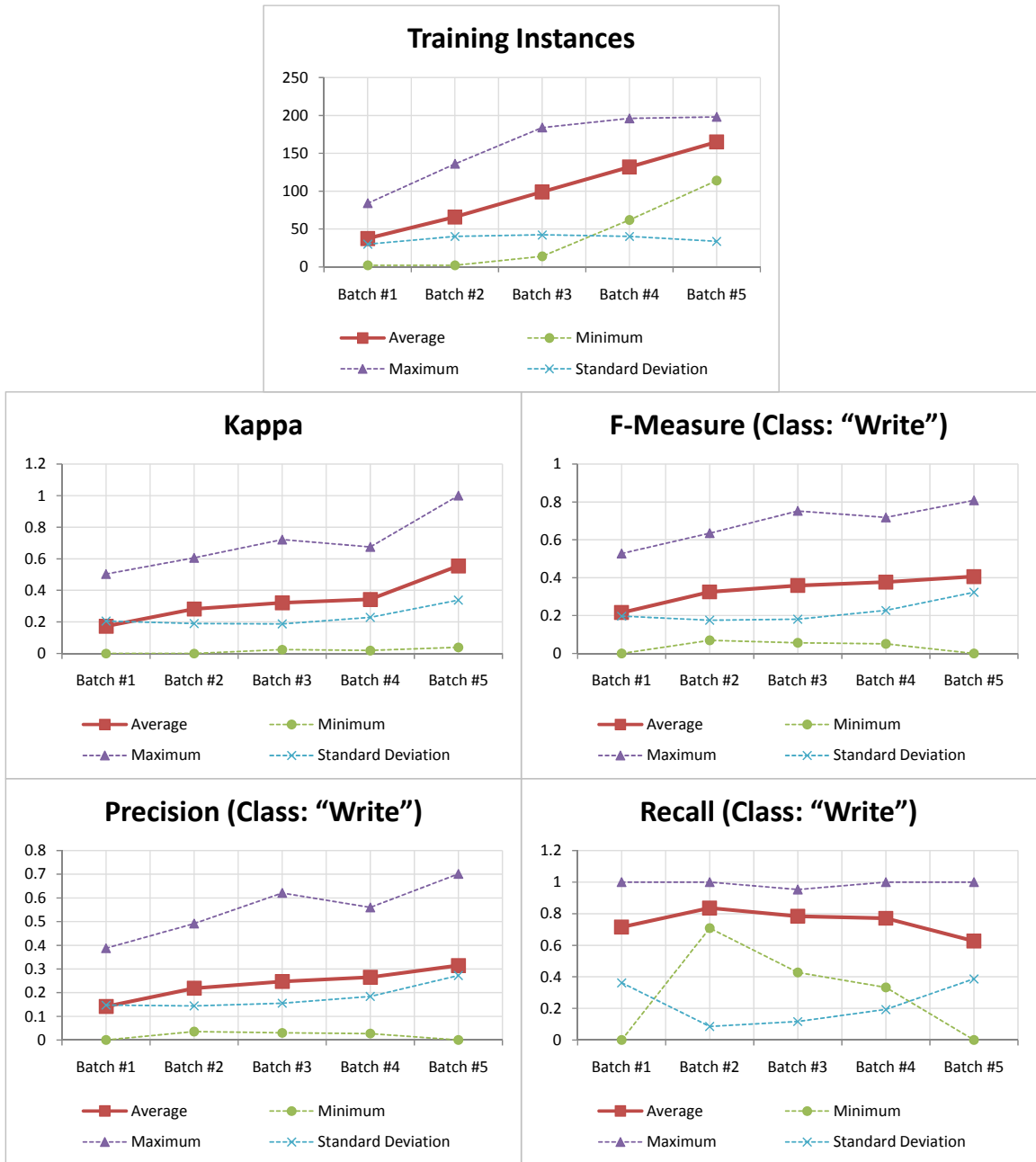


Figure 43: Results of Running Batch Tests for Incremental Learning Over the Write Classifier

The results shown in Figures 40, 41, 42 and 43 demonstrate that, along with the steady increase of the average number of training instances in each batch, the most of the accuracy statistics of all of our classifiers also improved steadily. However, the overall accuracy often stayed low in average, which is also understandable given the number of training instances

versus the testing instances, and also, each batch always had a few trials, which included one or more documents in the test set that belong to a problem domain or a source, which were never related to the documents used for training in those trials. Thus, there existed one or more trials in every batch where the accuracy statistics were very low, as reflected by the minimum curve on the charts of the figures, bringing the overall average down. However, it should be mentioned that there also exist trials in each batch where all the accuracy statistics were comparatively very high. This is an important success, as all of the trials in all batches had document(s) in the test set that were unseen in the training set. We also should mention that we reported in these figures only the critical statistical attributes over the positive instances, which are much “rare” in number in comparison to the negative instances, where the statistical measures (e.g. the precision, the recall and the f-measure) are relatively very high. We also did not report here the percentage of correctly classified instances, which also were about 90% or more on average for all the classifiers. Detailed results of our batch testing are included in Appendix B.2.

Finally, the charts in Figures 40, 41, 42 and 43 show that all of our classifiers can attain very promising results when used in practice the number of training instances are increased to a sufficient level (which may be determined by continuing to test in future with more training data for each problem domain). This statement is evident by the continual rise of the average of statistical measures across the batches, as demonstrated by these figures. Although the total number of trails per batch went down over subsequent batches after Batch #3, the standard deviation still often went down for some statistical measures, showing that the results tending to converge at high and more stable values. All these results indicate that the application of our supervised-learning based is feasible only with the availability of sufficient training data.

### 7.11.2 Results of 10-fold Cross-Validation

We discuss in this section the results of running 10-fold cross-validation over each of the classifiers for our supervised learning-based approach. These results would indicate by simulating how much accuracy we can realistically achieve by using these classifier with the availability of sufficient training data.

Here, we first extracted the feature values to build four different datasets from all the documents of our four different corpora, as described in Section 7.9.1. We then randomly removed portions of the negative instances so that we have an equal number of the positive and negative instances in each of the datasets. Then, we ran the 10-fold cross-validation tests individually over each of our data-movement classifiers, described in Section 7.6, using their respective datasets. All these tests were run using Weka’s 10-fold cross-validation testing module. The detailed outputs produced by Weka our 10-fold cross-validation experiments are included in Appendix B.1.

Each of our binary corpora that we used in our experiments is highly imbalanced with only about 104 instances (or 3.7% of all the instances) on average belonging to the positive classes and about 2708 instances (or 96.3% of all the instances) on average belonging to the negatives, as represented by the distribution of our corpora presented in Section 7.9.1. This high difference between the classes and the relatively low number of positive instances could lead to biased learning for our classifiers where they would emphasize on the negative instances more than the positive ones, when selecting the most discriminative features or determining their thresholds (Estabrooks, Jo, & Japkowicz, 2004). We therefore down-sampled our extracted datasets by randomly removing about 2604 negative instances (or about 92.6% of all the instances) from each of the datasets. This, for example, led the dataset used in our 10-fold cross-validation experiment over our Entry classifier to have a total of 236 instances (i.e. 118 “Entry” and 118 “Not Entry” instances).

The results of our execution of 10-fold cross-validation on all four of our classifiers using their respective datasets are shown in Table 34 and Table 35.

|               |           | Classified As |           |
|---------------|-----------|---------------|-----------|
|               |           | Entry         | Not Entry |
| Gold-Standard | Entry     | 97            | 21        |
|               | Not Entry | 15            | 103       |

|               |          | Classified As |          |
|---------------|----------|---------------|----------|
|               |          | Read          | Not Read |
| Gold-Standard | Read     | 93            | 16       |
|               | Not Read | 21            | 88       |

|               |          | Classified As |          |
|---------------|----------|---------------|----------|
|               |          | Exit          | Not Exit |
| Gold-Standard | Exit     | 77            | 15       |
|               | Not Exit | 10            | 82       |

|               |           | Classified As |           |
|---------------|-----------|---------------|-----------|
|               |           | Write         | Not Write |
| Gold-Standard | Write     | 83            | 16        |
|               | Not Write | 2             | 97        |

Table 34: Confusion Matrices for 10-fold Cross Validation on Supervised Learning-based Data-Movement Classification

Using the confusion matrices in Table 34, we can compute the accuracy results for this experiment, as shown in Table 35.

| (For Positive Class of) Data-Movement Type | Accuracy (Ratio of Correct) | Kappa        | Precision    | Recall       | F-Measure    |
|--------------------------------------------|-----------------------------|--------------|--------------|--------------|--------------|
| Entry                                      | 0.847                       | 0.695        | 0.866        | 0.822        | 0.843        |
| Exit                                       | 0.864                       | 0.728        | 0.885        | 0.837        | 0.860        |
| Read                                       | 0.830                       | 0.661        | 0.816        | 0.853        | 0.834        |
| Write                                      | 0.909                       | 0.818        | 0.976        | 0.838        | 0.902        |
| <b>Average =</b>                           | <b>0.862</b>                | <b>0.725</b> | <b>0.886</b> | <b>0.837</b> | <b>0.860</b> |

Table 35: Results of 10-fold Cross Validation on Supervised Learning-based Data-Movement Classification

The 10-fold cross-validation results, as presented in Table 35, show that our supervised learning-based classification approach can almost be as good as our heuristics-based classification approach, where the average agreement of its classifications with the true gold-standard labels set by the expert [shown in terms of Cohen’s Kappa (Cohen, 1960) in Table 33] is also better than the average pair-wise agreement of fully-trained human annotators with the same expert, as recorded by the results of our experiments, presented in Section 5.4.2.

## 7.12 CFP Range Measurement Results

We finally analyse the accuracy of the CFP range measurements for all the requirements documents of our corpus that are computed using the formulas presented in Section 7.8 over the automatically classified instances of noun-phrases and sentences following one of our classification approaches. In this experiment, we used the classification labels produced by our heuristic-based classification approach over our corpus. Our analysis involves comparing the CFP measurement accuracy of this approach to that of our trained human annotators in measuring the CFP manually for the same set of documents. We measured the accuracies in terms of the magnitudes of relative error (MRE) and the mean magnitude of relative error (MMRE). Table 36 presents the MRE and MMRE results of measuring CFP ranges for the all documents of our corpus, based on the classification labels of the noun-phrases and sentences of the documents, where the frequencies of the identified data-movement labels are aggregated by data-groups and functional processes following the formulas in Section 7.8.

| Doc. ID       | Total CFP                           |                                                |                    |                | MRE                                            |                    |                |
|---------------|-------------------------------------|------------------------------------------------|--------------------|----------------|------------------------------------------------|--------------------|----------------|
|               | Based on Expert's Annotation Labels | Based on Heuristic-based Classification Labels |                    |                | Based on Heuristic-based Classification Labels |                    |                |
|               |                                     | <i>Minimum</i>                                 | <i>Most-Likely</i> | <i>Maximum</i> | <i>Minimum</i>                                 | <i>Most-Likely</i> | <i>Maximum</i> |
| C1            | 65                                  | 51                                             | 69                 | 75             | 0.216                                          | 0.061              | 0.154          |
| C2            | 16                                  | 14                                             | 18                 | 23             | 0.125                                          | 0.125              | 0.437          |
| C3            | 80                                  | 66                                             | 84                 | 87             | 0.175                                          | 0.050              | 0.087          |
| C4            | 133                                 | 83                                             | 141                | 147            | 0.376                                          | 0.060              | 0.105          |
| C5            | 10                                  | 8                                              | 13                 | 15             | 0.200                                          | 0.300              | 0.500          |
| C6            | 49                                  | 44                                             | 58                 | 62             | 0.102                                          | 0.184              | 0.265          |
| <b>MMRE =</b> |                                     |                                                |                    |                | 0.199                                          | 0.130              | 0.258          |

Table 36: CFP Range Measurements Results Based on the Heuristic-based Classification Labels

The results presented in Table 36 show that the most-likely total CFP's of all the documents of our corpus, as calculated from the classification labels that are automatically produced by our heuristic-based classification approach, are nearly accurate (i.e. MMRE is only 0.13, and the MRE's are less than 20% for about 83.33% of all the documents), when matched with the total CFP's calculated from the true gold standard annotation labels chosen by the expert. Here, we find that the MMRE result of the most-likely total CFP's, calculated from our automatically produced classification labels, is still not as good as what the fully-trained

human annotators achieved manually (as presented in Table 14 of Section 5.6), but only when their collective gold-standard annotation labels, which were settled by multiple adjudication sessions, were considered. The most-likely total CFP's for our automated approach still achieved better MMRE results than what our four trained measurers on average could achieve individually (that is, their average individual MMRE = 0.165). Finally, we also find in Table 36 that the CFP ranges (i.e. from the minimum to the maximum total CFP) of all the documents of our corpus, as calculated from the our automated approach always included their actual CFP's within the limits of the calculated ranges. Thus, we can affirm that these CFP ranges produced from our automated approaches can successfully be used in practice, wherever COSMIC functional size measurement process is applied. Defining the correct limits of the CFP ranges ensure that any further estimation performed using the CFP values, e.g. the early estimation of the development effort, would also result in a range of estimation, and thus provide safety by allowing a concise margin of error.

## 7.13 Conclusion

In this chapter, we presented in details our approaches of automating the process of COSMIC functional size measurement. We introduced a heuristic-based and a supervised learning-based text classification approaches to extract the modeling artifacts of COSMIC FSM, e.g. the data-attributes, the data-groups and the data-movement. We then extended our formalization to COSMIC's method of counting the numeric value of CFP, by presenting the idea of measuring CFP in ranges. We finally analyzed the results of our experiments to validate each of our approaches. The results of the experiments showed that our automated approaches achieved good enough results to be applied in practice.

The next chapter ends the thesis with a summary of the contributions we made with this research in the fields of functional size measurement and requirements engineering and presents avenues of future work.



## Chapter 8

# Conclusions and Future Work

“The best way to predict the future is to create it.”  
— *Peter Drucker*

This final chapter summarizes the major contributions of this thesis. We then end with a review of our conclusive remarks and a discussion on future research.

### 8.1 Major Contributions

Our research investigated different techniques of linguistic analyses that can be utilized to measure functional size early from textual requirements, without depending on human experts. New approaches were developed and validated through experiments. All of these approaches were adapted to the COSMIC method (COSMIC, 2014), as the preferred standard of functional size measurement (FSM). The major contributions made by this thesis are discussed in the following sections.

#### 8.1.1 Methodology for Practical Application of FSM

The thesis proposed a unique comprehensive methodology, called LISMA, that integrates multiple novel approaches for performing functional size measurement (FSM) in different practical scenarios (as presented in Section 4.4). To validate the methodology, we conducted several controlled experiments with real project documents, experts and well-trained measurers. The experimental results show that our approaches can (i) attain high quality manual measurements of functional size in absence of an expert or a well-trained measurer or a costly adjudication process, (ii) approximate functional size with minimal errors from

informally written textual requirements, and (iii) extract the conceptual artifacts of an FSM standard from textual requirements and quantify their functional size with a good enough accuracy in comparison to that of well-trained human measurers. The results presented in this thesis also showed that the approaches used in LISMA can successfully automate the execution of functional size measurement process by applying different natural language processing and machine learning technologies. The details of these results were presented in Chapters 5, 6 and 7. We thus accomplished the overall aim of this research, presented in Section 1.3: which was “*to determine an objective procedure that does not depend on human expertise to be applied for effectively measuring functional size from textual requirement*”.

### **8.1.2 Formalization of an FSM Model for Traceability**

In Chapter 4, we formalized the FSM process by modeling a new ontology that relates the conceptual artifacts of FSM to the specific textual segments of the software requirements document to build traceability links over process of FSM. Thus, every step of the approaches in LISMA is devised towards measuring a traceable output of functional size, where all the traceability links resulting from the population of the ontology justify the output measurements by referring them to their originating segments of textual requirements. This satisfied our research objective #5 (see Section 1.3), which was “*to identify how experts deduce the relationship between the linguistic elements of unrestricted textual requirements and the objects of interest in a functional size measurement model*”.

### **8.1.3 Extension to FSM Quantification**

In Chapter 4, we also devised novel formulas to formalize the conventional quantification process of FSM. These formulas cannot only be applied over the instances of this ontology to calculate the numerical value of the functional size, but also fully complies with the standard process of FSM, described in (ISO/IEC 14143-1, 2007). Thus, this formalization can be implemented algorithmically to automate the computation of a numerical value of the functional size of a software. Then, in Chapter 7, we extended this formalization by deriving new additional formulas that quantify the upper and lower limits of the functional size as well. Our results show that the upper and lower limits provide added safety in FSM, as the outputted error margins in our experiments always included the correct result within its range.

### **8.1.4 Syntactic Features for Requirements Classification**

In Section 4.4.2, we presented an overview of our work to determine the most discriminating syntactic features of textual requirements for classifying requirements into the two classes of functional and non-functional requirements. The full details of the work and the experimental results, as published in (Hussain, Kosseim, & Ormandjieva, 2008), verifies the feasibility of using these features for requirements classification. Thus, we fulfilled our research objective #4: *“to explore the most discriminating syntactic features of textual requirements for classifying them into functional and non-functional requirements”*.

### **8.1.5 Improving the Annotation Quality for Non-Experts**

In Chapter 5, we presented the details of our requirements annotation tool, LASR, that implements a novel feature to improve the quality of the computed gold-standard annotations for multiple annotators. The feature automatically applies the formulas that we developed to compute the gold-standard annotations using the annotators’ levels of confidence and their levels of skill. The analyses of the results of our experiments showed that this feature improved the quality of the gold-standard annotations for a controlled group of non-expert annotators with minimal training. The experiments also showed that these higher quality gold-standards helped in measuring the functional size more accurately. LASR’s other features, such as its graphical interface, allowed the annotators to finish their functional size measurement tasks faster than the other group working manually. Thus, we fulfilled our research objective #1: *“to investigate if the process of functional size measurement (FSM) can be executed effectively with non-expert”*, and objective #2, which was *“to improve the overall process of FSM-related requirements annotation by attaining accurate annotations with non-experts having minimal training”*.

### **8.1.6 Automatic and Traceable Approximation of FSM**

In Chapter 6, we presented a novel approach to approximate functional size automatically from textual requirements using text classification techniques. The approach first classifies the software instances of a historical dataset by the quartiles of their functional size, and uses that information to build a corpus of textual requirements, which are annotated by approximated size classes. It then applies a supervised learning-based text classifier to

approximate the functional size and achieved moderate results in our experiments with a very small dataset of 61 instances. Since our approach uses a decision tree-based classification model, the reasoning of the output approximations are also traceable, back to its originating textual requirements. Our approach also identifies the most discriminating linguistic features that correlate with the approximated functional size, and thus addressed our research objective #3: *“to determine the most discriminating linguistic features of informally written textual requirements for approximating functional size”*.

### **8.1.7 Linguistic Features of COSMIC FSM**

In Chapter 7 (and also, in Appendix C), we presented an original list of lexical, syntactic and combined sets of features that can discriminate base noun-phrases on whether they express a certain type of COSMIC data-movement, which is a conceptual artifact of the COSMIC FSM model. We identified these features by analyzing the experts’ process of identifying these artifacts from textual requirements. We used these features in data-movement classification experiments, and our supervised learning-based approach automatically ranked these features by building decision trees using our supplied dataset. The analysis of our results showed that these features were discriminating enough for our supervised learning-based approach to attain promising results.

### **8.1.8 Heuristics for COSMIC FSM**

In Chapter 7, we also presented a detailed heuristic-based approach, which is unique in this field of research, to identify various conceptual artifacts of the COSMIC FSM model. The results of our experiments showed that this heuristic-based approach attains good results overall, and the accuracy of the functional size measurements that used the automatically generated outputs of this heuristic-based approach attains even better results than individual manual annotations of the fully-trained annotators. Thus, we addressed our research objective #6 (see Section 1.3): *“to evaluate the feasibility of automating functional size measurement from textual requirements”*.

### **8.1.9 Annotated Corpora for Data-Movement Classification**

As described in Chapters 5 and 7, we generated a set of fairly large annotated corpora, each containing a total of 2812 instances of base noun-phrases. All these instances were annotated by the expert as the true gold-standard annotation labels, and thus can be used to train or test any supervised learning-based approach in the field of COSMIC data-movement classification, although we have only small portions of it annotated with positive class-labels. The details of the corpora were described in Section 7.9.1.

## **8.2 Future Research Directions**

The functional size measurement (FSM) method provides an effective means of assessing the software size at the early stages of the software development lifecycle. However, its application over textual requirements poses many costly challenges thus is not widely adopted by the industry.

In this thesis, we addressed these challenges by devising a comprehensive methodology that not only facilitates a collaborative annotation environment to improve the quality of manual measurement of functional size, but also utilizes different natural language processing techniques to either automatically approximate the functional size from informally-written textual requirements, or automatically extract the FSM model from well-decomposed textual requirements and thus, measure the functional size objectively by using a formalized model of quantifying the functional size. This novel modeling approach for FSM also creates traceability links between the originating parts of the textual requirements and the resulting FSM modeling artifacts that are counted to quantify the functional size.

The approaches that are integrated in our methodology were all validated through controlled experiments employing industrial and academic projects' data and fully-trained human measurers. The promising results of these experiments thus confirm that our methodology can successfully be applied in practice.

For our future research work, we intend to continue collecting more datasets for training and/or testing our automated approaches. Having more datasets would not only help our supervised learning-based classifier to utilize more discriminating features, but also allow us to experiment with enough varieties of problem domains and study the effects of varying the problem domains. Acquiring more datasets would also help us investigate how the quality of textual requirements affects our functional size measurement approaches.

In relation to the prototype of LASR, the requirements annotation tool that we developed during this research, we also intend to release its complete version, further experiment on its usability and study if it can help minimize the annotators' effort for different requirements annotation tasks.

Finally, we plan to integrate our automated approaches of functional size measurement with our implementation of estimating the development effort, which is presented in (Abdukalykov, Hussain, Kassab, & Ormandjieva, 2011) to deploy a comprehensive workbench that can estimate the development effort directly from textual requirements.

# Bibliography

- Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7 (1), 39-59.
- Abbott, R. J. (1983). Program design by informal English descriptions. *Communications of the ACM*, 26 (11), 882-894.
- Abdukalykov, R., Hussain, I., Kassab, M., & Ormandjieva, O. (2011). Quantifying the Impact of Different Non-functional Requirements and Problem Domains on Software Effort Estimation. *Proceedings of 9th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 158-165). Washington, DC: IEEE Computer Society.
- Aiello, G., Alessi, M., Cossentino, M., Urso, A., & Vella, G. (2007). RTDWD: Real-Time Distributed Wideband-Delphi for User Stories Estimation. *Proceedings of the 3rd International Conference on Rapid Integration of Software Engineering Techniques* (pp. 35-50). Springer-Verlag.
- Albrecht, A. J. (1979). Measuring Application Development Productivity. *Proceedings of IBM Application Development Symp.* (pp. 83-92). Monterey, Calif.: Press I.B.M.
- Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: *A software science validation*. *IEEE Transactions on Software Engineering*, 9, 639-648.
- Amazon.com Inc. (2012). Amazon Mechanical Turk. Amazon Mechanical Turk: <https://www.mturk.com/mturk/welcome>

- Ambriola, V., & Gervasi, V. (2006). On the Systematic Analysis of Natural Language Requirements with CIRCE. *Automated Software Engineering, Automated Software Engineering*, 13 (1), 107-167.
- Ambriola, V., & Gervasi, V. (1997). Processing natural language requirements. *Proceedings of Automated Software Engineering (ASE'97): 12th IEEE International Conference*, November 1–5 (pp. 36-45). IEEE Computer Society.
- Angelis, L., & Stamelos, I. (2000). A simulation tool for efficient analogy based cost estimation. *Empirical Software Engineering*, 5, 35-68.
- Azzeh, M., Neagu, D., & Cowling, P. (2008). Improving analogy software effort estimation using fuzzy feature subset selection algorithm. *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering* (pp. 71-78). Leipzig, Germany: ACM.
- Beck, K., & Fowler, M. (2000). *Planning Extreme Programming*. Addison-Wesley.
- Bertran, M., Borrega, O., Recasens, M., & Soriano, B. (2008). AnCoraPipe: A tool for multilevel annotation. *Procesamiento del Lenguaje Natural*, 41, 291-292.
- Bevo, V. (2005). *Analyse et formalisation ontologique des procédures de mesure associées aux méthodes de mesure de la taille fonctionnelle des logiciels: de nouvelles perspectives pour la mesure*. Doctoral thesis, Montréal: Université du Québec à Montréal - UQAM.
- Boehm, B. (2000). Safe and Simple Software Cost Analysis. *IEEE Software*, 17 (5), 14-17.
- Boehm, B. (1981). *Software engineering economics*. Prentice-Hall.
- Boehm, B. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*, 10, 1, 4-21.
- Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R. & Reifer, D.. (2000). *Software cost estimation with Cocomo II*. Prentice-Hall.



- Bontcheva, K., Cunningham, H., Roberts, I., Roberts, A., Tablan, V., Aswani, N., & Gorrell, G. (2013). Teamware: A Web-based, Collaborative Text Annotation Framework. *Language Resources and Evaluation*, 47 (4), 1007-1029.
- Braga, P. L., Oliveira, A. L., & Meira, S. R. (2008). A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation. *Proceedings of the 2008 ACM Symposium on Applied Computing* (pp. 1788-1792). Fortaleza, Ceara, Brazil: ACM.
- Brill, E. (1992). A Simple Rule-Based Part of Speech Tagger. *Proceedings of the Third Conference on Applied Natural Language Processing* (pp. 152-155). Trento, Italy: Association for Computational Linguistics.
- Burgess, C. J., & Lefley, M. (2001). Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, 43, 863-873.
- Cake Software Foundation. (2014, May 1). CakePHP Cookbook Documentation: Release 2.x. Retrieved May 3, 2014, from CakePHP:  
[http://book.cakephp.org/2.0/\\_downloads/en/CakePHPCookbook.pdf](http://book.cakephp.org/2.0/_downloads/en/CakePHPCookbook.pdf)
- Carletta, J. (1996). Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22, 249-255.
- Casamayor, A., Godoy, D., & Campo, M. (2009). Semi-Supervised Classification of Non-Functional Requirements: An Empirical Analysis. *Inteligencia Artificial*, 44, 35-45.
- Cer, D., de Marneffe, M.-C., Jurafsky, D., & Manning, C. D. (2010). Parsing to Stanford Dependencies: Trade-offs between speed and accuracy. *Proceedings of 7th International Conference on Language Resources and Evaluation (LREC 2010)* (pp. 1628–1632). European Language Resources Association (ELRA).
- Chang, C. K., Christensen, M. J., & Tao, Z. (2001). Genetic algorithms for project management. *Annals of Software Engineering*, 11, 107-139.

- Chomsky, N. (1965). *Aspects of the Theory of Syntax*. Cambridge, MA: MIT Press.
- Chulani, S., Boehm, B., & Steece, B. (1999). Bayesian analysis of empirical software engineering cost models. *IEEE Transactions on Software Engineering*, 25, 573-583.
- Chung, L., & Sapakkul, S. (2006). Capturing and Reusing Functional and Non-functional Requirements Knowledge: A Goal-Object Pattern Approach. *Proceedings of the 2006 IEEE International Conference on Information Reuse and Integration*, September (pp. 539-544). Waikoloa, Hawaii, USA: IEEE Press.
- Cicchetti, D. V., & Feinstein, A. R. (1990). High agreement but low kappa: II. Resolving the paradoxes. *Journal of Clinical Epidemiology*, 43 (6), 551-558.
- Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2006). The Detection and Classification of Non-Functional Requirements with Application to Early Aspects. *Proceedings of the 14th IEEE International Requirements Engineering Conference 2006 (RE'06)*, September 11-15 (pp. 36-45). Minneapolis, MN: IEEE Press.
- Cochran, W. G. (1977). *Sampling techniques* (3rd ed.). John Wiley & Sons.
- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Journal of Educational and Psychological Measurement*, 20, 37-46.
- Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall.
- Condori-Fernández, N., Abrahão, S., & Pastor, O. (2007). On the estimation of the functional size of software from requirements specifications. *Journal of Computer Science and Technology*, 22 (3), 358-370.
- Conte, S. D., Dunsmore, H. E., & Shen, V. Y. (1986). *Software engineering metrics and models*. Redwood City, CA: Benjamin Cummings Publishing.
- COSMIC. (2014). The COSMIC Functional Size Measurement Method Version 4.0: Measurement Manual. Retrieved May 05, 2014, from COSMIC: [http://www.cosmicon.com/dl\\_manager4.asp?id=464](http://www.cosmicon.com/dl_manager4.asp?id=464)

- COSMIC. (2011). Why COSMIC is the best method for measuring Agile 'User Stories'. *COSMIC News*, 7 (1), 3.
- Cunningham, H., Maynard, D., Bontcheva, K., & Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)* (pp. 168-175). PA, USA: Association for Computational Linguistics.
- Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., *et al.* (2011). *Text Processing with GATE (Version 6)*. Sheffield, UK: Department of Computer Science, University of Sheffield.
- Cyre, W. (1995). A Requirements Sublanguage for Automated Analysis. *International Journal of Intelligent Systems*, 10 (7), 665-689.
- Cysneiros, L. M., & Leite, J. C. (2002). Non-functional requirements: from elicitation to modelling languages. *Proceedings of the International Conference on Software Engineering, 2002 (ICSE 2002)*, May (pp. 699-700). Orlando, Florida, USA: IEEE Press.
- Dawson, C. W. (1996). A neural network approach to software project effort estimation. In R. A. Adey, G. Rzevski, & A. K. Sunol (Ed.), *Proceedings of International Conference on Artificial Intelligence in Engineering* (p. 37). School of Mathematics & Computing, Derby University, UK.
- Delobel, C. (1978). Normalization and Hierarchical Dependencies in the Relational Data Model. *ACM Trans. Database Syst.*, 3 (3), 201-222.
- Demirors, O., & Gencel, C. (2004). A Comparison of Size Estimation Techniques Applied Early in the Life Cycle. *Software Process Improvement, Lecture Notes in Computer Science, vol. 3281* (pp. 184-194), Berlin, Heidelberg: Springer -Verlag.

- Denger, C., Berry, D. M., & Kamsties, E. (2003). Higher Quality Requirements Specifications through Natural Language Patterns. *Proceedings of the IEEE International Conference on Software Science, Technology and Engineering* (p. 80). Washington, DC: IEEE Computer Society.
- Diab, H., Koukane, F., Frappier, M., & St-Denis, R. (2005).  $\mu$ ROSE: Automated Measurement of COSMIC-FFP for Rational Rose Real Time. *Information and Software Technology*, 47 (3), 151-166.
- Drazan, J., & Mencl, V. (2007). Improved processing of textual use cases: Deriving behavior specifications. Lecture Notes in Computer Science-SOFSEM 2007: Theory and Practice of Computer Science, *Proceedings of 33rd Conference on Current Trends in Theory and Practice of Computer Science*, 4362, 856–868.
- Eriksson, M., Börstlerb, J., & Borga, K. (2009). Managing requirements specifications for productlines – An approach and industry case study. *Journal of Systems and Software*, 82 (3), 435-447.
- Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A Multiple Resampling Method for Learning from Imbalanced Data Sets. *Computational Intelligence*, 20 (1), 18-39.
- Eye, A. v., & Eye, M. v. (2008). On the Marginal Dependency of Cohen's  $\kappa$ . *European Psychologist*, 13 (4), 305-315.
- Fabbrini, F., Fusani, M., Gnesi, S., & Lami, G. (2001). An Automatic Quality Evaluation for Natural Language Requirements. *Proceedings of the Seventh International Workshop on RE: Foundation for Software Quality (REFSQ'2001)*, June 4-5. Interlaken, Switzerland.
- Fantechi, A., Gnesi, S., Ristori, G., Carenini, M., Vanocchi, M., & Moreschini, P. (1994). Assisting requirement formalization by means of natural language translation. *Form. Methods Syst. Des.*, 4 (3), 243-263.

- Flitman, A. M. (2000). A neural network DEA meta-model to facilitate software development time and cost estimation. *Proceedings of Artificial Neural Networks in Engineering Conference*. 10, (pp. 941-946). New York, NY, USA: ASME.
- Foss, T., Stensrud, E., Kitchenham, B., & Myrtveit, I. (2003). A simulation study of the model evaluation criterion MMRE. *IEEE Transactions on Software Engineering*, 29, 985-995.
- Fraser, S., Boehm, B., Erdogmus, H., Jorgensen, M., Rifkin, S., & Ross, M. (2009). The role of judgment in software estimation. *Proceedings of 31st International Conference on Software Engineering, ICSE 2009*, May 16-24, (pp. 13-17). Vancouver, Canada.
- Gaffney, J. E., & Werling, R. (1991). Estimating Software Size from Counts of Externals, A Generalization of Function Points. *Analytical Methods in Software Engineering Economics* (pp. 193-203). Berlin, Heidelberg: Springer.
- Galesic, M., & Bosnjak, M. (2009). Effects of Questionnaire Length on Participation and Indicators of Response Quality in a Web Survey. *Public Opinion Quarterly*, 73 (2), 349-360.
- Galorath. (2008). *SEER for Software Development: Estimating Software Projects*. Retrieved November 6, 2007, from *SEER by Galorath*:  
<http://www.galorath.com/index.php/products/software/C5>
- Gelhausen, T., & Tichy, W. F. (2007). Thematic Role Based Generation of UML Models from Real World Requirements. *Proceeding of The First International Conference on Semantic Computing (ICSC 2007)* (pp. 282-289). Los Alamitos: IEEE Computer Society.
- Gencel, C., & Demirors, O. (2008). Functional size measurement revisited. *Transactions on Software Engineering and Methodology (TOSEM)*, 17 (3), 15:1-15:36.

- Gencel, C., Demirors, O., & Yuceer, E. (2005). A Case Study on Using Functional Size Measurement Methods for Real Time Systems. *Proceedings of the 15th. International Workshop on Software Measurement (IWSM)* (pp. 159-178). Shaker-Verlag.
- Gencel, C., Demirors, O., & Yuceer, E. (2005). Utilizing Functional Size Measurement Methods for Real Time Software Systems. *11th IEEE International Software Metrics Symposium (METRICS 2005)*. Como, Italy. Retrieved May 05, 2014, from: [http://metrics2005.di.uniba.it/IndustryTrack/Gencel\\_Utilizingms.pdf](http://metrics2005.di.uniba.it/IndustryTrack/Gencel_Utilizingms.pdf)
- Gnesi, S., Lami, G., & Trentanni, G. (2005). An automatic tool for the analysis of natural language requirements. *International Journal of Computer Systems Science and Engineering, Special issue on Automated Tools for Requirements Engineering*, 20, 53-62.
- Grimstad, S., & Jorgensen, M. (2007). The Impact of Irrelevant Information on Estimates of Software Development Effort. *Proceedings of the 2007 Australian Software Engineering Conference, ASWEC '07* (pp. 359-368). IEEE Computer Society.
- Habela, P., Głowacki, E., Serafiński, T., & Subieta, K. (2005). Adapting Use Case Model for COSMIC-FFP Based Measurement. *Proceedings of 15th International Workshop on Software Measurement (IWSM-2005)*, (pp. 195-207). Montreal.
- Hakkarainen, J., Laamanen, P., & Rask, R. (1993). Neural networks in specification level software size estimation. *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, 4, (pp. 626-634). IEEE Press.
- Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11 (1), 10-18.
- Harmain, H., & Gaizauskas, R. (2000). CM-Builder: An automated NL-based CASE tool. *Proceedings of the Fifteenth IEEE International Conference on Automated Software Engineering*, September 11-15 (pp. 45-53). Grenoble, France: IEEE Press.

- Heiat, A. (2002). Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology*, 44, 911-922.
- Heinrich, E., Kemp, E., & Patrick, J. (1999). A Natural Language Like Description Language. *Proceedings of the 10th Australasian Conference on Information Systems (ACIS)*, (pp. 375–386). Wellington, New Zealand.
- Hill, R., Wang, J., & Nahrstedt, K. (2004). Quantifying Non-functional Requirements: A Process Oriented Approach. *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04)*, September (pp. 352-353). Kyoto, Japan: IEEE Press.
- Hoehler, F. K. (2000). Bias and prevalence effects on kappa viewed in terms of sensitivity and specificity. *Journal of Clinical Epidemiology*, 53 (5), 499-503.
- Hoogendoorn, S. (2009). Measuring agile progress in smart use case points. Retrieved May 05, 2014, from Sander Hoogendoorn:  
<http://sanderhoogendoorn.com/blog/index.php/measuring-agile-progress-in-smart-use-case-points/>
- Huang, S.-J., & Chiu, N.-H. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and Software Technology*, 48, 1034-1045.
- Huang, X., Ho, D., Ren, J., & Capretz, L. (2004). A neuro-fuzzy tool for software estimation. *Proceedings of the 20th IEEE International Conference on Software Maintenance*, (p. 520). IEEE Press.
- Hussain, I. (2007). *Automated Ambiguity Detection in Natural Language Software Requirements*. Master's Thesis, Department of Computer Science and Software Engineering, Concordia University.

- Hussain, I., Kosseim, L., & Ormandjieva, O. (2013). Approximation of COSMIC functional size to support early effort estimation in Agile. *Data & Knowledge Engineering*, 85, 2-14.
- Hussain, I., Kosseim, L., & Ormandjieva, O. (2008). Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents. In *LNCS: Natural Language and Information Systems*, vol. 5039/2008 (pp. 287-298). Germany: Springer-Verlag.
- Hussain, I., Ormandjieva, O., & Kosseim, L. (2007). Automatic Quality Assessment of SRS Text by Means of a Decision-Tree-Based Text Classifier. *Proceedings of the Seventh International Conference on Quality Software (QSIC 2007)* (pp. 209-218). Portland, USA: IEEE Computer Society.
- Hussain, I., Ormandjieva, O., & Kosseim, L. (2012). LASR: A Tool For Large Scale Annotation of Software Requirements. *Proceedings of EmpiRE 2012, the International Workshop on Empirical Requirements Engineering*, (pp. 57-60). Chicago, IL: IEEE.
- Hussain, I., Ormandjieva, O., & Kosseim, L. (2009). Mining and Clustering Textual Requirements to Measure Functional Size of Software with COSMIC. *Proceedings of the International Conference on Software Engineering Research and Practice (SERP 2009)* (pp. 599-605). CSREA Press.
- Hussain, I., Ormandjieva, O., & Kosseim, L. (2010). Towards Approximating COSMIC Functional Size from User Requirements in Agile Development Processes Using Text Mining. In *LNCS: Natural Language Processing and Information Systems*, vol. 6177/2010 (pp. 80-91).
- Idri, A., Abran, A., & Khoshgoftaar, T. M. (2002). Estimating software project effort by analogy based on linguistic values. *Proceedings of International Software Metrics Symposium* (pp. 21-30). Ottawa, Canada: IEEE Press.



- Idri, A., Khoshgoftaar, T. M., & Abran, A. (2002). Can neural networks be easily interpreted in software cost estimation? *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems*, 2 (pp. 1162-1167). IEEE Press.
- IEEE. (2004). A. Abran, & P. Bourque (Eds.) *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society Press.
- IEEE. (1998). *IEEE recommended practice for software requirements specifications*. IEEE Std 830-1998.
- IFPUG. (2013). *Software Non-functional Assessment Process (SNAP): Assessment Practices Manual - Release 2.1*. International Function Point Users Group (IFPUG).
- ISO/IEC 14143-1. (1998). *Functional Size Measurement - Definition of Concepts*. International Organization for Standardization.
- ISO/IEC 14143-1. (2007). *Functional Size Measurement - Definition of Concepts*. International Organization for Standardization.
- ISO/IEC 19761. (2003). *COSMIC Full Function Points Measurement Manual v2.2*. International Organization for Standardization.
- ISO/IEC 19761. (2011). *COSMIC Full Function Points Measurement Method*. International Organization for Standardization.
- ISO/IEC 20926. (2003). *Software engineering — IFPUG 4.1 Unadjusted functional size measurement method — Counting Practices Manual*. International Organization for Standardization.
- ISO/IEC 20968. (2002). *Software Engineering — Mk II Function Point Analysis — Counting Practices Manual*. International Organization for Standardization.
- ISO/IEC 24570. (2005). *Software engineering — NESMA functional size measurement method version 2.1 — Definitions and counting guidelines for the application of Function Points Analysis*. International Organization for Standardization.

- ISO/IEC 29881. (2010). *Information technology — Systems and software engineering — FiSMA 1.1 functional size measurement method*. International Organization for Standardization.
- Jedlitschka, A., Ciolkowski, M., & Pfahl, D. (2008). Reporting Experiments in Software Engineering. In F. Shull, J. Singer, & D. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering* (pp. 201-228). London, UK: Springer.
- Jenner, M. S. (2011). Automation of Counting of Functional Size Using COSMIC FFP in UML. In R. R. Dumke, & A. Abran (Eds.), *COSMIC Functional Points: Theory and Advanced Practices* (pp. 276-283). Boca Raton, FL: Auerbach Publications.
- John, G. H., & Langley, P. (1995). Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence* (pp. 338-345). Morgan Kaufmann.
- Jones, C. (1997). *Applied Software Measurement: Assuring Productivity and Quality* (2nd ed.). New York, NY: McGraw-Hill.
- Jongmoon, B., Boehm, B., & Steece, B. M. (2002). Disaggregating and calibrating the CASE tool variable in COCOMO II. *IEEE Transactions on Software Engineering*, 28, 1009-1022.
- Jørgensen, M. (2004a). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70, 37-60.
- Jørgensen, M. (2004b). Regression Models of Software Development Effort Estimation Accuracy and Bias. *Empirical Software Engineering*, 9, 297-314.
- Jørgensen, M., & Molokken-Ostfold, K. (2004). Reasons for software effort estimation error: impact of respondent role, information collection approach, and data analysis method. *IEEE Transactions on Software Engineering*, 30, 993-1007.
- Jørgensen, M., & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering*, 33, 33-53.

- Kamsties, E., Berry, D. M., & Paech, B. (2001). Detecting Ambiguities in Requirements Documents Using Inspections. *Proceedings of the First Workshop on Inspection in Software Engineering (WISE'01)*, (pp. 68-80).
- Karttunen, L. (1971). Implicative Verbs. *Language*, 47 (2), 340-358.
- Kassab, M. (2009). *Non-Functional Requirements: Modeling and Assessment*. VDM Verlag.
- Kassab, M., Ormandjieva, O., Daneva, M., & Abran, A. (2008). Non-Functional Requirements: Size Measurement and Testing with COSMIC-FFP. In J. J. Cuadrado-Gallego, R. Braungarten, R. R. Dumke, & A. Abran (Eds.), *Software Process and Product Measurement* (pp. 168-182). Berlin, Heidelberg: Springer-Verlag.
- Kemerer, C. F. (1987). An empirical validation of software cost estimation models. *Communications of the ACM*, 30, 416-429.
- Kitchenham, B. A., & Taylor, N. R. (1984). Software cost models. *ICL Technical Journal*, 4, 73-102.
- Kitchenham, B., Pickard, L., MacDonell, S., & Shepperd, M. (2001). What accuracy statistics really measure [software estimation]. *Proceedings of IEEE Software*, 148 (2), 81-85.
- Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (pp. 423-430). Stroudsburg, PA: Association for Computational Linguistics.
- Ko, Y., Park, S., Seo, J., & Choi, S. (2007). Using classification techniques for informal requirements in the requirements analysis-supporting system. *Information and Software Technology*, 49, 1128-1140.
- Körner, S. J., & Landhäußer, M. (2010). Semantic Enriching of Natural Language Texts with Automatic Thematic Role Annotation. *Natural Language Processing and Information Systems*, vol. 6177 (pp. 92-99). Berlin, Heidelberg: Springer Verlag.

- Kraemer, H. C., Periyakoil, V. S., & Noda, A. (2004). Kappa coefficients in medical research. *Tutorials in Biostatistics Volume 1: Statistical Methods in Clinical Studies*. John Wiley & Sons, Ltd.
- Krenn, B., Evert, S., & Zinsmeister, H. (2004). Determining Intercoder Agreement for a Collocation Identification Task. *Proceedings of Konvens'04*, September, (pp. 89–96). Vienna, Austria.
- Landhäußer, M., Körner, S. J., & Tichy, W. F. (2014). From requirements to UML models and back: how automatic processing of text can support requirements engineering. *Software Quality Journal*, 22 (1), 121-149.
- Landis, R. J., & Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33, 159-174.
- Larman, C. (2003). *Agile and Iterative Development: A Manager's Guide*. Pearson Education.
- le Cessie, S., & van Houwelingen, J. (1992). Ridge Estimators in Logistic Regression. *Applied Statistics*, 41 (1), 191-201.
- Lederer, A. L., & Prasad, J. (1992). Nine management guidelines for better cost estimating. *Communications of the ACM*, 35, 51-59.
- Lederer, A. L., & Prasad, J. (1991). The validation of a political model of information systems development cost estimating. *ACM Transactions on Computer Personnel*, 13, 47-57.
- Leffingwell, D., & Widrig, D. (2003). *Managing Software Requirements: A Use Case Approach*. Pearson Education.
- Letier, E., Kramer, J., Magee, J., & Uchitel, S. (2005). Monitoring and control in scenario-based requirements analysis. *Proceedings of the 27th International Conference on Software Engineering* (pp. 382-391). Louis, Missouri, USA: ACM Press.

- Levine, R. L., & Hunter, J. E. (1983). Regression methodology: Correlation, meta-analysis, confidence intervals, and reliability. *Journal of Leisure Research*, 15, 323-343.
- Lewis, J. P. (2001). Limits to software estimation. *Software Engineering Notes*, 26, 54-59.
- Lin, D. (2003). Dependency-Based Evaluation of Minipar. In *Text, Speech and Language Technology: Treebanks*, 20 (pp. 317-329), Netherlands: Springer.
- Liu, D. (2003). *Automating Transition from Use Cases to Class Model*, Master's Thesis, Dept. of Electr. & Comput. Eng., University of Calgary. Calgary, AB.
- Marín, B., Pastor, O., & Giachetti, G. (2008). Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment. *Proceedings of the 9th international conference on Product-Focused Software Process Improvement* (pp. 215-229). Berlin, Heidelberg: Springer-Verlag.
- Martin, R. C. (2003). *Agile Software Development: Principles, Patterns and Practices*. Prentice Hall.
- McConnell, S. (2006). *Software Estimation—Demystifying the Black Art*. Microsoft Press.
- Meli, R. (1997). Early and Extended Function Point: a new method for Function Points estimation. *Proceedings of IFPUG - Fall Conference*, September 15-19. Scottsdale, AZ.
- Mendes, E., & Mosley, N. (2008). Bayesian Network Models for Web Effort Prediction: A Comparative Study. *IEEE Transactions on Software Engineering*, 34, 723-737.
- Mendes, E., Martino, S. D., Ferrucci, F., & Gravino, C. (2007). Effort estimation: how valuable is it for a web company to use a cross-company data set, compared to using its own single-company data set? *Proceedings of the 16th international conference on World Wide Web* (pp. 963-972). ACM.

- Menzies, T., Chen, Z., Port, D., & Hihn, J. (2005). Simple Software Cost Estimation: Safe or Unsafe? *Proceedings of PROMISE 2005, International Workshop on Predictor Models in Software Engineering* (pp. 1-6) New York, NY: ACM.
- Menzies, T., Zhihao, C., Hihn, J., & Lum, K. (2006). Selecting Best Practices for Effort Estimation. *IEEE Transactions on Software Engineering*, 32, 883-895.
- Meyer, B. (1985). On Formalism in Specifications. *IEEE Software*, 2, 6-26.
- Mich, L., & Garigliano, R. (2002). NL-OOPS: A Requirements Analysis tool based on Natural Language Processing. *Proceedings of the 3rd International Conference On Data Mining*, September 25-27 (pp. 321-330). Bologna.
- Microsoft Corp. (2011). Windows Intune: Software Categories. Retrieved June 02, 2012, from Windows Intune:  
<http://onlinehelp.microsoft.com/en-us/windowsintune/ff399004.aspx>
- Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, 38 (11), 39-41.
- Minfang, Z., Woye, L., & Jiansi, C. (2012). Cost Estimation of Equipment Software in Earlier Period Based on Wideband Delphi and Function Resolve. In *LNEE: Information Engineering and Application*, vol. 154 (pp. 1304-1309). Springer.
- Miranda, E., Bourque, P., & Abran, A. (2009). Sizing user stories using paired comparisons. *Information and Software Technology*, 51, 1327-1337.
- Miyazaki, Y., Terakado, M., Ozaki, K., & Nozaki, H. (1994). Robust regression for developing software estimation models. *Journal of Systems and Software*, 27, 42430.
- Molkken, K., & Jørgensen, M. (2003). A Review of Surveys on Software Effort Estimation. *Proceedings of the 2003 International Symposium on Empirical Software Engineering* (p. 223). Washington, DC: IEEE Computer Society.

- Moradi-Seresht, S., Ormandjieva, O., & Sabra, S. (2008). Automatic Conceptual Analysis of User Requirements with the Requirements Engineering Assistance Diagnostic (READ) Tool. *Proceedings of 6th International Conference on Software Engineering Research, Management and Applications (SERA 2008)* (pp. 133-142). Prague: IEEE.
- Moreira, A., Araujo, J., & Brito, I. (2002). Crosscutting Quality Attributes for Requirements Engineering. *Proceedings of 14th International Conference on Software Engineering and Knowledge Engineering*, (pp. 167–174). Ischia, Italy.
- Musilek, P., Pedrycs, W., Succi, G., & Reformat, M. (2000). Software cost estimation with fuzzy models. *Applied Computing Review*, 8, 24-29.
- Mylopoulos, J., Chung, L., & Nixon, B. (1992). Representing and Using Nonfunctional Requirements: A process Oriented Approach. *IEEE Trans. S.E.*, 18, 483-497.
- Myrtveit, I., Stensrud, E., & Shepperd, M. (2005). Reliability and validity in comparative studies of software prediction models. *IEEE Transactions on Software Engineering*, 31, 380-391.
- Ogren, P. V. (2006). Knowtator: A Protégé plug-in for annotated corpus construction. *Proceedings of the 2006 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology* (pp. 273-275). Morristown, NJ: Association for Computational Linguistics.
- Ormandjieva, O., Hussain, I., & Kosseim, L. (2007). Toward a text classification system for the quality assessment of software requirements written in natural language. *Proceedings of SOQUA '07: Fourth international workshop on Software quality assurance* (pp. 39-45). ACM.
- Osborne, M., & MacNish, C. K. (1996). Processing Natural Language Software Requirement Specifications. *Proceedings of ICRE '96, the 2nd International Conference on Requirements Engineering* (p. 229). IEEE Computer Society.

- Park, H., & Baek, S. (2008). An empirical validation of a neural network model for software effort estimation. *Expert Systems with Applications*, 35, 929-937.
- Pendharkar, P. C., Subramanian, G. H., & Rodger, J. A. (2005). A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering*, 31, 615-624.
- Pfleeger, S. L., Wu, F., & Lewis, R. (2005). *Software Cost Estimation and Sizing Methods, Issues and Guidelines*. RAND Corporation.
- Port, D., & Korte, M. (2008). Comparative studies of the model evaluation criterions MMRE and PRED in software cost estimation research. *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement* (pp. 51-60). ACM.
- Princeton University. (2010). About WordNet. Retrieved May 05, 2014, from WordNet: <http://wordnet.princeton.edu>
- Putnam, L. H. (1981). SLIM: a quantitative tool for software cost and schedule estimation. *Proceedings of NBS/IEEE/ACM Software Tool Fair* (pp. 49-57). McLean, VA: Quantitative Software Management Inc.
- Putnam, L. H., & Meyers, W. (1992). *Measures for Excellence: Reliable Software, on Time, Within Budget*. Prentice Hall.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning* (Morgan Kaufmann Series in Machine Learning) (1 ed.). Morgan Kaufmann.
- Rashwan, A., Ormandjieva, O., & Witte, R. (2013). Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. *Proceedings of COMPSAC 2013, the 37th Annual International Computer Software & Applications Conference* (pp. 381-386). Kyoto, Japan: IEEE.
- Regolin, E., Souza, G. d., Pozo, A., & Vergilio, S. (2003). Exploring machine learning techniques for software size estimation. *Proceedings of SCCC 2003, the 23rd*



*International Conference of the Chilean Computer Science Society* (pp. 130-136).  
Los Alamitos, CA, USA.

Rolland, C., & Proix, C. (1992). A Natural Language Approach For Requirements Engineering. *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, vol. 593 of *Lecture Notes in Computer Science* (pp. 257–277). Manchester, UK.

Rubin, H. A. (1983). Macroestimation of software development parameters: The Estimacs System. *Proceedings of SOFTFAIR Conference on Software Development Tools, Techniques and Alternatives* (pp. 109-118). Arlington, Va: IEEE Press.

Rule, S., & Rule, P. G. (2011). Everything You Always Wanted to Know About Software Measurement. *Methods and Tools*, 19 (2), 13-24.

Saadaoui, S., Majchrowski, A., & Ponsard, C. (2009). Experiment with COSMIC V3.0: Case Studies in Business Applications. *Proceedings of the 16th European Systems & Software Process Improvement and Innovation (EuroSPI'09) Conference*,. Spain: University of Alcala. Retrieved May 20, 2014, from:  
<https://www.cetic.be/IMG/pdf/cetic-eurospi09-applying-cosmic-final.pdf>

Samarasinghe, N., & S., S. (2005). Generating a Domain Model from a Use Case Model. *Proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering* (pp. 23-29). Toronto, Canada.

Sang, E. F., Daelemans, W., Déjean, H., Koeling, R., Krymolowski, Y., Punyakanok, V., et al. (2000). Applying System Combination to Base Noun Phrase Identification. *Proceedings of the 18th Conference on Computational Linguistics*. vol. 2 (pp. 857-863). Stroudsburg, PA: Association for Computational Linguistics.

Santillo, L., Conte, M., & Meli, R. (2005). E&Q: An Early & Quick Approach to Functional Size. *Proceedings of IEEE International Symposium on Software Metrics* (p. 41). Los Alamitos, CA, USA: IEEE Computer Society.

- Sayyad Shirabad, J., & Menzies, T. (2005). The PROMISE Repository of Software Engineering Databases. (School of Information Technology and Engineering, University of Ottawa, Canada) Retrieved November 6, 2009, from:  
<http://promise.site.uottawa.ca/SERepository>
- Shan, Y., McKay, R. I., Lokan, C. J., & Essam, D. L. (2002). Software project effort estimation using genetic programming. *Proceedings of International Conference on Communications* (pp. 1108-1112). Canberra, ACT, Australia: IEEE Computer Society.
- Shepperd, M., & Cartwright, M. (2001). Predicting with sparse data. *IEEE Transactions on Software Engineering*, 27, 987-998.
- Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23, 736-743.
- Shukla, K. K. (2000). Neuro-genetic prediction of software development effort. *Information and Software Technology*, 42, 701-713.
- Sneed, H. M. (2001). Extraction of function points from source-code. *Proceedings of New Approaches in Software Measurement, 10th International Workshop, IWSM* (pp. 135-146). Berlin, Germany: Springer-Verlag.
- Some, S. (2006). Supporting use case based requirements engineering. *Journal of Information and Software Technology*, 48, 43-58.
- Srinivasan, L., & Stefan, S. S. (2010). *Patent No. US 7,743,369 B1*. United States of America.
- Stamelos, I., Angelis, L., Dimou, P., & Sakellaris, E. (2003). On the use of Bayesian belief networks for the prediction of software productivity. *Information and Software Technology*, 45, 51-60.
- Stanford Center for Biomedical Informatics Research. (2014). The Protégé Ontology Editor and Knowledge Acquisition System. Retrieved May 20, 2014, from:  
<http://protege.stanford.edu>

- SWEBOK. (2004). *Guide to the Software Engineering Body of Knowledge (2004 Version)*. (A. Abran, J. W. Moore, P. Bourque, & R. Dupuis, Eds.) CA, US: IEEE Computer Society.
- The Standish Group. (2013). *CHAOS Manifesto 2013: Think Big, Act Small*. Boston, MA: The Standish Group International Inc.
- Tjong, S. F., Hallam, N., & Hartley, M. (2006). Improving the Quality of Natural Language Requirements Specifications through Natural Language Requirements Patterns. *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology* (pp. 199-204). IEEE Press.
- Total Metrics. (2007, August). *Methods for Software Sizing: How To Decide Which Method To Use*. Retrieved May 19, 2014, from totalmetrics.com:  
[http://www.totalmetrics.com/function-point-resources/downloads/R185\\_Why-use-Function-Points.pdf](http://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf)
- Trudel, S., & Abran, A. (2009). Functional Size Measurement Quality Challenges for Inexperienced Measurers. *Proceedings of the International Conferences on Software Process and Product Measurement* (pp. 157-169). Berlin, Heidelberg: Springer-Verlag.
- Trudel, S., & Abran, A. (2008). Improving quality of functional requirements by measuring their functional size. *Proceedings of the International Conferences on Software Process and Product Measurement* (pp. 287-301). Berlin, Heidelberg: Springer-Verlag.
- Vogelezang, F., Symons, C., Lesterhuis, A., Meli, R., & Daneva, M. (2013). Approximate COSMIC Functional Size — Guideline for Approximate COSMIC Functional Size Measurement. *Proceedings of The Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA)* (pp. 27-32). IEEE Press.

- Weiss, S. M., Indurkha, N., & Zhang, T. (2004). Text Mining. Predictive Methods for Analyzing Unstructured Information. Springer, Berlin.
- Widlöcher, A., & Mathet, Y. (2009). La plate-forme Glozz: Environnement d'annotation et d'exploration de corpus. Retrieved May 19, 2014, from:  
[http://lipn.univ-paris13.fr/taln09/pdf/TALN\\_120.pdf](http://lipn.univ-paris13.fr/taln09/pdf/TALN_120.pdf)
- Wiebe, J., Wilson, T., Bruce, R., Bell, M., & Martin, M. (2004). Learning Subjective Language. *Computational Linguistics*, 30 (3), 277-308.
- Wilson, W. M., Rosenberg, L. H., & Hyatt, L. E. (1996). Automated quality analysis of Natural Language requirement specifications. *Proceedings of 14th Annual Pacific Northwest Software Quality Conference*, (pp. 140-151).
- Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques (2nd ed.)*. Morgan Kaufmann.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer-Verlag.
- Yue, T., Briand, L. C., & Labiche, Y. (2011). A systematic review of transformation approaches between user requirements and analysis models. *Requirements Engineering*, 16 (2), 75-99.
- Zhang, G., Patuwo, E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14, 35-62.

# Appendix A

## Lexical Databases

In our observation of our corpus, we deduced that the semantics of the COSMIC data-movements can be realized from textual requirements by using both the syntactical and the lexical information embedded in the requirements. We, therefore, used a set of vocabularies (or dictionaries, or gazetteers) to build our source of lexical knowledge. We applied these vocabularies in all of our classification tasks to enrich our approaches with the necessary lexical knowledge. We describe these vocabularies below:

### A.1 Data-movement Verbs

These verbs commonly appear to express the sense of different types of data-movements and are not specific to any problem domain. We build different vocabularies of data-movement verbs by the different types of possible data-movements. For each of these vocabularies that is related to a particular type of data-movement, we first seed it with some of the commonly used verbs to express the action of that specific type of data-movements. We then expand the vocabulary with the synsets of WordNet (Miller, 1995; Princeton University, 2010), that is by adding new synonyms that are common to each pair of verbs in our vocabulary until no new synonym exists for all possible pairs. Thus, the following sections list the words in the vocabularies that we ended up building, each related to a particular type of data-movement.

#### A.1.1 Entry Verbs

These verbs commonly appear to express the action of Entry data-movements and are not specific to any problem domain. The following are the list of Entry verbs that we ended up building for our experiments:

|              |        |        |         |          |         |            |        |
|--------------|--------|--------|---------|----------|---------|------------|--------|
| assign       | click  | Enter  | input   | post     | select  | substitute | type   |
| authenticate | create | fill   | mention | provide  | set     | tap        | update |
| change       | define | give   | modify  | re-enter | specify | tell       |        |
| choose       | edit   | inform | point   | search   | submit  | touch      |        |

### A.1.2 Exit Verbs

These verbs commonly appear to express the action of Exit data-movements and are not specific to any problem domain. The following are the list of Exit verbs that we ended up building for our experiments:

|         |      |         |          |        |        |
|---------|------|---------|----------|--------|--------|
| default | list | output  | preset   | return | Show   |
| display | mail | post    | print    | search | update |
| edit    | mark | present | retrieve | send   | view   |

### A.1.3 Read Verbs

These verbs commonly appear to express the action of Read data-movements and are not specific to any problem domain. The following are the list of Read verbs that we ended up building for our experiments:

|              |      |        |           |        |          |
|--------------|------|--------|-----------|--------|----------|
| authenticate | edit | obtain | read      | return | Validate |
| check        | find | post   | recognize | search | Verify   |
| display      | get  | preset | retrieve  | update |          |

### A.1.4 Write Verbs

These verbs commonly appear to express the action of Write data-movements and are not specific to any problem domain. The following are the list of Write verbs that we ended up building for our experiments:

|         |        |        |        |          |        |        |
|---------|--------|--------|--------|----------|--------|--------|
| add     | change | define | erase  | register | set    | Update |
| archive | copy   | delete | insert | remove   | store  |        |
| book    | create | edit   | record | save     | submit |        |

### A.1.5 Triggering-Entry Verbs

When these verbs appear within the sentences, especially in presence of an actor name as a subject, they often implicitly indicate the sense of Entry data-movement of triggering events of functional processes. The following are the list of Triggering-Entry verbs that we ended up building for our experiments:

|          |          |         |       |      |
|----------|----------|---------|-------|------|
| check    | click    | request | begin | want |
| navigate | initiate | ask     | start |      |

### A.1.6 System-Message-Exit Verbs

When these verbs appear within the sentences, they often implicitly indicate the sense of Exit data-movement of a special data-group, called “System Message”. The following are the list of System-Message-Exit verbs that we ended up building for our experiments:

|          |         |        |        |             |
|----------|---------|--------|--------|-------------|
| check    | confirm | inform | refuse | notify      |
| validate | prompt  | signal | show   | acknowledge |

## A.2 Stative Verbs

We develop the vocabulary of stative verbs that are mostly used to describe the states of objects instead of describing actions over them. The usage of these verbs is also not specific to any problem domain. These verbs in most cases describe the state being or having or spatial relations amongst objects. We again used WordNet (Miller, 1995; Princeton University, 2010) to manually extract our vocabulary of stative verbs. Thus, we end up building the following vocabulary of stative verbs:

|         |         |          |         |         |         |           |            |
|---------|---------|----------|---------|---------|---------|-----------|------------|
| be      | belong  | deserve  | hate    | know    | mind    | realize   | surprise   |
| have    | agree   | disagree | hear    | like    | need    | recognise | understand |
| consist | appear  | dislike  | imagine | love    | owe     | recognize | want       |
| own     | believe | doubt    | impress | matter  | prefer  | remember  | weigh      |
| possess | concern | feel     | include | mean    | promise | seem      | wish       |
| contain | depend  | fit      | involve | measure | realise | suppose   |            |

### A.3 Attribute Names

We developed this vocabulary of attributes containing words that can be used to represent members or properties or measurable attributes of any entity (i.e. data-group, for our work). We manually selected a slice of noun-phrases from WordNet (Miller, 1995; Princeton University, 2010) that belong to the super classes Property, Relation and Communication via their hypernym paths. We then manually modified some of the words to create new forms that may appear in textual requirements. We list the base of forms of these words that generated our vocabulary below:

|              |               |             |              |             |            |                |               |
|--------------|---------------|-------------|--------------|-------------|------------|----------------|---------------|
| #            | choice        | definition  | footnote     | inductance  | operand    | race           | size          |
| acceleration | circumference | degree      | force        | intensity   | option     | radius         | specification |
| address      | citation      | depth       | frequency    | interval    | p.i.n.     | radiation      | speed         |
| advantage    | city          | density     | function     | language    | payment    | raise          | spot          |
| age          | clarity       | description | gender       | lastname    | payoff     | range          | status        |
| aid          | class         | diameter    | genre        | latitude    | parameter  | rank           | stipend       |
| allowance    | code          | dimension   | grade        | length      | path       | rate           | strength      |
| altitude     | color         | discount    | gravity      | level       | password   | ratio          | string        |
| amount       | colour        | distance    | group        | limit       | percentage | rebate         | telephone     |
| angle        | comment       | duration    | growth       | limitation  | period     | reference      | temperature   |
| aperture     | commission    | duty        | head         | location    | ph#        | remuneration   | text          |
| area         | compensation  | earning     | header       | longitude   | phone      | reply          | title         |
| attribute    | concentration | elasticity  | heading      | loss        | photo      | resilience     | time          |
| badge        | condition     | email       | headline     | luminance   | photograph | resistance     | torque        |
| badge#       | conductivity  | ethnicity   | heartrate    | magnitude   | picture    | resistivity    | transparency  |
| batch        | conductance   | expanse     | height       | mass        | pin        | response       | type          |
| batch#       | consistency   | expenditure | hour         | message     | population | responsibility | username      |
| benefit      | constraint    | expense     | humidity     | measure     | portion    | restriction    | value         |
| birthdate    | consumption   | expertise   | hue          | measurement | position   | resolution     | velocity      |
| birthday     | contrast      | experience  | i.d.         | mileage     | post       | revenue        | viscosity     |
| bitrate      | cost          | exposure    | id           | minute      | power      | reward         | voltage       |
| body         | count         | extent      | id#          | month       | pressure   | role           | volume        |
| bonus        | country       | extension   | id's         | momentum    | premium    | salary         | wage          |
| brightness   | criterion     | factor      | ident        | money       | profit     | saturation     | watt          |
| capacity     | date          | fee         | illumination | name        | proportion | second         | wavelength    |
| caption      | datum         | filename    | image        | no.         | price      | selection      | weight        |
| carat        | day           | firstname   | income       | number      | pulse      | serial#        | width         |
| category     | debt          | flux        | info         | occupation  | quantity   | sex            | year          |
| charge       | deduction     | footer      | information  | opacity     | quota      | situation      | zip           |

### A.4 Data-group Names

According to our approach, as discussed in Section 4.4.3, the names of the data-groups belonging to the problem domains of our corpus were already available to us. For our corpus, these names formed out a vocabulary of data-group names. These are listed below:

|         |        |            |             |           |         |              |
|---------|--------|------------|-------------|-----------|---------|--------------|
| budget  | course | fund       | opportunity | product   | role    | subscription |
| chart   | event  | invitation | page        | professor | search  | user         |
| content | filter | news       | performance | revenue   | student |              |



## A.5 Actor Names

The Entry data-movements in most cases are caused by primary actors who are human. The mentions of these actors appear in software requirements documents as base noun-phrases that indicate names of people or their roles or their professions or their positions in an organization. Thus, we used a dictionary to identify the candidates for such noun-phrases. We suggest including more names of human actors to this dictionary based on our knowledge of the problem domains and the human actors they involve. It should be mentioned that the mentions of non-human actors, e.g. the systems, subsystems, modules, features, functions etc., that are named with “helper”, “assistant”, “expert”, “client”, “publisher”, “presenter”, “holder”, “sender”, “reader”, “collector”, “master”, “resident”, “viewer”, “receiver”, “worker”, “operator”, “builder”, “manager”, “learner”, “trainer” etc., may be mistakenly be identified as candidates for human actors because of the use of this dictionary. We, therefore, consider the identified noun-phrases as candidates only, a subset of which can then finally indicate human actors only when they are combined with other syntactic features leading to the deduction. The vocabulary that we used in our tests is presented below:

|                |             |              |                 |              |              |                |              |
|----------------|-------------|--------------|-----------------|--------------|--------------|----------------|--------------|
| acc-holder     | biologist   | coach        | employee        | landlady     | participant  | receptionist   | spouse       |
| accountant     | bloke       | collector    | employer        | landlord     | partner      | recipient      | staff        |
| account-holder | bookkeeper  | colonel      | engineer        | lawyer       | patient      | recruit        | stenographer |
| actor          | borrower    | commander    | examiner        | leader       | patron       | referee        | student      |
| actress        | boss        | commissioner | expert          | learner      | person       | registrar      | subscriber   |
| adjunct        | botanist    | competitor   | father          | lender       | personnel    | rep            | supporter    |
| administrator  | boy         | consumer     | fellow          | lessee       | physician    | representative | taker        |
| advisor        | brigadier   | contender    | female          | lessor       | physicist    | resident       | teacher      |
| advocate       | broker      | contestant   | follower        | lieutenant   | pianist      | resider        | technician   |
| agent          | brother     | contributor  | friend          | magistrate   | pilot        | scholar        | teller       |
| ally           | buddy       | coo          | gamer           | male         | player       | scientist      | tenant       |
| amateur        | builder     | cook         | general         | man          | poet         | secretary      | trader       |
| ambassador     | buyer       | councilor    | girl            | manager      | poetess      | seller         | trainee      |
| anthropologist | c.e.o.      | counsellor   | governor        | marketer     | policeman    | senator        | trainer      |
| apprentice     | c.o.o.      | counselor    | graduate        | master       | politician   | sender         | trainer      |
| archeologist   | campaigner  | critic       | grandfather     | mate         | practitioner | senior         | tutor        |
| artisan        | captain     | customer     | grandmother     | mayor        | presenter    | servant        | uncle        |
| artist         | cardholder  | defendant    | guy             | mechanic     | president    | shareholder    | user         |
| assistant      | card-holder | delegate     | helper          | member       | principal    | shopper        | v.i.p.       |
| associate      | cashier     | demonstrator | historian       | mentor       | proctor      | shopper        | viewer       |
| athlete        | ceo         | dentist      | holder          | merchandiser | professional | sister         | violinist    |
| attendant      | chairman    | deputy       | householder     | mineworker   | professor    | soldier        | vip          |
| attorney       | challenger  | designer     | husband         | minister     | programer    | solicitor      | voter        |
| auditor        | chap        | detective    | individual      | mother       | programmer   | somebody       | waiter       |
| aunt           | chef        | digger       | inspector       | musician     | promoter     | someone        | wife         |
| author         | chemist     | director     | instructor      | nurse        | psychologist | sophomore      | witness      |
| banker         | chief       | disciple     | instrumentalist | officer      | publisher    | speaker        | woman        |
| barrister      | classmate   | doctor       | judgmental      | official     | purchaser    | specialist     | worker       |
| bartender      | clerk       | dude         | jury            | operator     | reader       | spokesperson   | writer       |
| beginner       | client      | editor       | laborer         | opponent     | receiver     | sportsman      | zoologist    |

## Appendix B

# Raw Test Results: Outputs from WEKA

In this appendix, we present the detailed raw outputs generated by Weka (Witten & Frank, 2005) during the 10-fold cross-validation tests and the batch tests with incremental learning, all of which were with our supervised learning-based data-movement classifiers.

## B.1 10-fold Cross-Validation Results

Weka's outputs for our 10-fold cross validation tests are presented in the following sections.

### B.1.1 For *Entry* Classifier

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -U -M 5
Relation: Entry_nounPhrase_Classification-
weka.filters.supervised.instance.SpreadSubsample-M1.0-X0.0-S1
Instances: 236
Attributes: 19
 isDirectObject
 isInObjectLikePosition
 isPartOfDependantClause
 isPartOfDependantClauseType
 isObjectOfDataMovementVerb
 isObjectOfVerbWithSubjectType
 mainVerbIsDataMovementVerb
 mainVerbHasSubjectType
 isSubjectOfStativeVerb
 isPartOfNegativeSense
 isAttribute
 ownsAttribute
 isDataGroup
 belongsToDataGroup
 isRelatedToAttribute
 isRelatedToDataGroup
 isObjectOfEntryVerb
 mainVerbIsEntryVerb
 class
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 unpruned tree

```

```

isObjectOfEntryVerb = true
| isObjectOfVerbWithSubjectType = actor: Entry (84.0/4.0)
| isObjectOfVerbWithSubjectType = nonactor: Entry (0.0)
| isObjectOfVerbWithSubjectType = nosubject
| | isDataGroup = true: Entry (13.0/4.0)
| | isDataGroup = false
| | | isDirectObject = true: Entry (7.0/3.0)
| | | isDirectObject = false: NotEntry (5.0)
isObjectOfEntryVerb = false
| isDataGroup = true
| | isPartOfDependantClause = true: Entry (5.0/1.0)
| | isPartOfDependantClause = false
| | | isAttribute = true: Entry (10.0/3.0)
| | | isAttribute = false
| | | | isObjectOfVerbWithSubjectType = actor: Entry (5.0/2.0)
| | | | isObjectOfVerbWithSubjectType = nonactor: NotEntry (0.0)
| | | | isObjectOfVerbWithSubjectType = nosubject: NotEntry (41.0/11.0)
| isDataGroup = false: NotEntry (66.0)

```

Number of Leaves : 11

Size of the tree : 19

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===  
 === Summary ===

|                                  |           |           |
|----------------------------------|-----------|-----------|
| Correctly Classified Instances   | 200       | 84.7458 % |
| Incorrectly Classified Instances | 36        | 15.2542 % |
| Kappa statistic                  | 0.6949    |           |
| Mean absolute error              | 0.2014    |           |
| Root mean squared error          | 0.3303    |           |
| Relative absolute error          | 40.2736 % |           |
| Root relative squared error      | 66.0592 % |           |
| Total Number of Instances        | 236       |           |

=== Detailed Accuracy By Class ===

|               | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class    |
|---------------|---------|---------|-----------|--------|-----------|----------|----------|
|               | 0.822   | 0.127   | 0.866     | 0.822  | 0.843     | 0.904    | Entry    |
|               | 0.873   | 0.178   | 0.831     | 0.873  | 0.851     | 0.904    | NotEntry |
| Weighted Avg. | 0.847   | 0.153   | 0.848     | 0.847  | 0.847     | 0.904    |          |

=== Confusion Matrix ===

```

 a b <-- classified as
97 21 | a = Entry
15 103 | b = NotEntry

```

## B.1.2 For *Exit* Classifier

=== Run information ===

```

Scheme:weka.classifiers.trees.J48 -U -M 5
Relation: Exit_nounPhrase_Classification-
weka.filters.supervised.instance.SpreadSubsample-M1.0-X0.0-S1
Instances: 184
Attributes: 19
 isDirectObject
 isInObjectLikePosition
 isPartOfDependantClause
 isPartOfDependantClauseType
 isObjectOfDataMovementVerb

```

```

isObjectOfVerbWithSubjectType
mainVerbIsDataMovementVerb
mainVerbHasSubjectType
isSubjectOfStativeVerb
isPartOfNegativeSense
isAttribute
ownsAttribute
isDataGroup
belongsToDataGroup
isRelatedToAttribute
isRelatedToDataGroup
isObjectOfExitVerb
mainVerbIsExitVerb
class
Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 unpruned tree

isObjectOfExitVerb = true: Exit (84.0/7.0)
isObjectOfExitVerb = false: NotExit (100.0/15.0)

Number of Leaves : 2
Size of the tree : 3

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances 159 86.413 %
Incorrectly Classified Instances 25 13.587 %
Kappa statistic 0.7283
Mean absolute error 0.2142
Root mean squared error 0.3344
Relative absolute error 42.838 %
Root relative squared error 66.8727 %
Total Number of Instances 184

=== Detailed Accuracy By Class ===

 TP Rate FP Rate Precision Recall F-Measure ROC Area Class
 0.837 0.109 0.885 0.837 0.86 0.835 Exit
 0.891 0.163 0.845 0.891 0.868 0.835 NotExit
Weighted Avg. 0.864 0.136 0.865 0.864 0.864 0.835

=== Confusion Matrix ===

 a b <-- classified as
77 15 | a = Exit
10 82 | b = NotExit

```

### B.1.3 For Read Classifier

```

=== Run information ===

Scheme:weka.classifiers.trees.J48 -U -M 5
Relation: Read_nounPhrase_Classification-
weka.filters.supervised.instance.SpreadSubsample-M1.0-X0.0-S1
Instances: 218
Attributes: 19
 isDirectObject
 isInObjectLikePosition

```

```

isPartOfDependantClause
isPartOfDependantClauseType
isObjectOfDataMovementVerb
isObjectOfVerbWithSubjectType
mainVerbIsDataMovementVerb
mainVerbHasSubjectType
isSubjectOfStativeVerb
isPartOfNegativeSense
isAttribute
ownsAttribute
isDataGroup
belongsToDataGroup
isRelatedToAttribute
isRelatedToDataGroup
isObjectOfReadVerb
mainVerbIsReadVerb
class

```

Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

J48 unpruned tree

-----

```

isObjectOfReadVerb = true
| isDirectObject = true: Read (53.0/3.0)
| isDirectObject = false
| | isAttribute = true: NotRead (5.0/2.0)
| | isAttribute = false: Read (21.0/3.0)
isObjectOfReadVerb = false
| isAttribute = true
| | isObjectOfVerbWithSubjectType = actor: NotRead (6.0)
| | isObjectOfVerbWithSubjectType = nonactor: Read (0.0)
| | isObjectOfVerbWithSubjectType = nosubject: Read (28.0/7.0)
| isAttribute = false
| | isObjectOfDataMovementVerb = true
| | | isPartOfDependantClause = true: Read (5.0/1.0)
| | | isPartOfDependantClause = false
| | | | isDataGroup = true: NotRead (6.0/1.0)
| | | | isDataGroup = false: Read (7.0/3.0)
| | | isObjectOfDataMovementVerb = false: NotRead (87.0/9.0)

```

Number of Leaves : 10

Size of the tree : 18

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

|                                  |           |           |
|----------------------------------|-----------|-----------|
| Correctly Classified Instances   | 181       | 83.0275 % |
| Incorrectly Classified Instances | 37        | 16.9725 % |
| Kappa statistic                  | 0.6606    |           |
| Mean absolute error              | 0.2427    |           |
| Root mean squared error          | 0.3698    |           |
| Relative absolute error          | 48.5411 % |           |
| Root relative squared error      | 73.9563 % |           |
| Total Number of Instances        | 218       |           |

=== Detailed Accuracy By Class ===

|               | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class   |
|---------------|---------|---------|-----------|--------|-----------|----------|---------|
|               | 0.853   | 0.193   | 0.816     | 0.853  | 0.834     | 0.845    | Read    |
|               | 0.807   | 0.147   | 0.846     | 0.807  | 0.826     | 0.845    | NotRead |
| Weighted Avg. | 0.83    | 0.17    | 0.831     | 0.83   | 0.83      | 0.845    |         |

=== Confusion Matrix ===

```
 a b <-- classified as
93 16 | a = Read
21 88 | b = NotRead
```

## B.1.4 For *Write* Classifier

=== Run information ===

```
Scheme:weka.classifiers.trees.J48 -U -M 5
Relation: Write_nounPhrase_Classification-
weka.filters.supervised.instance.SpreadSubsample-M1.0-X0.0-S1
Instances: 198
Attributes: 19
 isDirectObject
 isInObjectLikePosition
 isPartOfDependantClause
 isPartOfDependantClauseType
 isObjectOfDataMovementVerb
 isObjectOfVerbWithSubjectType
 mainVerbIsDataMovementVerb
 mainVerbHasSubjectType
 isSubjectOfStativeVerb
 isPartOfNegativeSense
 isAttribute
 ownsAttribute
 isDataGroup
 belongsToDataGroup
 isRelatedToAttribute
 isRelatedToDataGroup
 isObjectOfWriteVerb
 mainVerbIsWriteVerb
 class
Test mode:10-fold cross-validation
```

=== Classifier model (full training set) ===

J48 unpruned tree

-----

```
isObjectOfWriteVerb = true: Write (82.0/1.0)
isObjectOfWriteVerb = false
| isPartOfDependantClauseType = advcl: NotWrite (1.0)
| isPartOfDependantClauseType = ccomp: NotWrite (6.0/2.0)
| isPartOfDependantClauseType = xcomp: NotWrite (3.0)
| isPartOfDependantClauseType = none
| | isAttribute = true
| | | isInObjectLikePosition = true: NotWrite (22.0/3.0)
| | | isInObjectLikePosition = false: Write (5.0)
| | isAttribute = false: NotWrite (79.0/8.0)
```

Number of Leaves : 7

Size of the tree : 11

Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===

=== Summary ===

|                                  |        |           |
|----------------------------------|--------|-----------|
| Correctly Classified Instances   | 180    | 90.9091 % |
| Incorrectly Classified Instances | 18     | 9.0909 %  |
| Kappa statistic                  | 0.8182 |           |

```

Mean absolute error 0.1467
Root mean squared error 0.2891
Relative absolute error 29.3285 %
Root relative squared error 57.8204 %
Total Number of Instances 198

```

```
=== Detailed Accuracy By Class ===
```

|               | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class    |
|---------------|---------|---------|-----------|--------|-----------|----------|----------|
|               | 0.838   | 0.02    | 0.976     | 0.838  | 0.902     | 0.877    | Write    |
|               | 0.98    | 0.162   | 0.858     | 0.98   | 0.915     | 0.877    | NotWrite |
| Weighted Avg. | 0.909   | 0.091   | 0.917     | 0.909  | 0.909     | 0.877    |          |

```
=== Confusion Matrix ===
```

```

a b <-- classified as
83 16 | a = Write
 2 97 | b = NotWrite

```

## B.2 Batch Test Results for Incremental Learning

The output of the Weka wrapper that we coded using Java for batch testing with incremental learning are Comma-Separated Value (CSV) files containing different performance results. The contents of these files are shown as tables in the following sections.

## B.2.1 For Entry Classifier

| Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances | Total Correct | Correct(%) | Kappa    | Mean Absolute Error | RMS      | FP Rate (+) | Precision (+) | Recall (+) | F-Measure (+) |
|----------------------|---------------------|--------------------------|-------------------------|---------------|------------|----------|---------------------|----------|-------------|---------------|------------|---------------|
| C1                   | C2,C3,C4,C5,C6      | 28                       | 2498                    | 2152          | 86.14892   | 0.250313 | 0.188668            | 0.316892 | 0.131997    | 0.189744      | 0.711538   | 0.299595      |
| C2                   | C1,C3,C4,C5,C6      | 8                        | 2753                    | 114           | 4.140937   | 0        | 0.5                 | 0.5      | 1           | 0.041409      | 1          | 0.079526      |
| C3                   | C1,C2,C4,C5,C6      | 44                       | 2101                    | 1232          | 58.63874   | 0.106715 | 0.297001            | 0.463037 | 0.433416    | 0.099482      | 1          | 0.180961      |
| C4                   | C1,C2,C3,C5,C6      | 62                       | 1494                    | 1037          | 69.41098   | 0.166637 | 0.334147            | 0.478422 | 0.316986    | 0.145594      | 0.873563   | 0.249589      |
| C5                   | C1,C2,C3,C4,C6      | 4                        | 2785                    | 116           | 4.165171   | 0        | 0.5                 | 0.5      | 1           | 0.041652      | 1          | 0.079972      |
| C6                   | C1,C2,C3,C4,C5      | 90                       | 2429                    | 2117          | 87.15521   | 0.229215 | 0.142681            | 0.319004 | 0.125637    | 0.161473      | 0.780822   | 0.267606      |
| C1,C2                | C3,C4,C5,C6         | 36                       | 2439                    | 2105          | 86.30586   | 0.249721 | 0.217621            | 0.344754 | 0.130398    | 0.18883       | 0.71       | 0.298319      |
| C1,C3                | C2,C4,C5,C6         | 72                       | 1787                    | 1258          | 70.39731   | 0.132331 | 0.250831            | 0.413288 | 0.301466    | 0.115318      | 0.817073   | 0.202112      |
| C1,C4                | C2,C3,C5,C6         | 90                       | 1180                    | 1092          | 92.54237   | 0.410328 | 0.196221            | 0.282449 | 0.04607     | 0.413793      | 0.493151   | 0.45          |
| C1,C5                | C2,C3,C4,C6         | 32                       | 2471                    | 2129          | 86.15945   | 0.24719  | 0.181059            | 0.308664 | 0.131701    | 0.1875        | 0.705882   | 0.296296      |
| C1,C6                | C2,C3,C4,C5         | 118                      | 2115                    | 1856          | 87.75414   | 0.217362 | 0.127849            | 0.273874 | 0.118677    | 0.152778      | 0.745763   | 0.253602      |
| C2,C3                | C1,C4,C5,C6         | 52                       | 2042                    | 1351          | 66.16063   | 0.127203 | 0.289809            | 0.4717   | 0.350769    | 0.110533      | 0.923913   | 0.197445      |
| C2,C4                | C1,C3,C5,C6         | 70                       | 1435                    | 1009          | 70.31359   | 0.170664 | 0.299274            | 0.463125 | 0.306953    | 0.147844      | 0.86747    | 0.252632      |
| C2,C5                | C1,C3,C4,C6         | 12                       | 2726                    | 2387          | 87.5642    | 0.293822 | 0.112462            | 0.307326 | 0.120122    | 0.216958      | 0.776786   | 0.339181      |
| C2,C6                | C1,C3,C4,C5         | 98                       | 2370                    | 2016          | 85.06329   | 0.183327 | 0.129942            | 0.323904 | 0.146023    | 0.131783      | 0.73913    | 0.223684      |
| C3,C4                | C1,C2,C5,C6         | 106                      | 783                     | 597           | 76.24521   | 0.263785 | 0.368602            | 0.506207 | 0.240947    | 0.231111      | 0.8        | 0.358621      |
| C3,C5                | C1,C2,C4,C6         | 48                       | 2074                    | 1494          | 72.03472   | 0.157855 | 0.252813            | 0.437747 | 0.287879    | 0.12844       | 0.893617   | 0.224599      |
| C3,C6                | C1,C2,C4,C5         | 134                      | 1718                    | 1394          | 81.14086   | 0.182253 | 0.202416            | 0.362153 | 0.191962    | 0.128065      | 0.921569   | 0.22488       |
| C4,C5                | C1,C2,C3,C6         | 66                       | 1467                    | 1351          | 92.09271   | 0.46478  | 0.200885            | 0.301215 | 0.065123    | 0.395973      | 0.694118   | 0.504274      |
| C4,C6                | C1,C2,C3,C5         | 152                      | 1111                    | 1007          | 90.63906   | 0.328061 | 0.168406            | 0.308701 | 0.086062    | 0.245902      | 0.714286   | 0.365854      |
| C5,C6                | C1,C2,C3,C4         | 94                       | 2402                    | 1961          | 81.6403    | 0.15006  | 0.157831            | 0.339211 | 0.181467    | 0.111345      | 0.746479   | 0.193784      |
| C1,C2,C3             | C4,C5,C6            | 80                       | 1728                    | 1428          | 82.63889   | 0.239704 | 0.231679            | 0.407152 | 0.172727    | 0.181034      | 0.807692   | 0.295775      |
| C1,C2,C4             | C3,C5,C6            | 98                       | 1121                    | 905           | 80.73149   | 0.26621  | 0.202954            | 0.342397 | 0.192015    | 0.214008      | 0.797101   | 0.337423      |
| C1,C2,C5             | C3,C4,C6            | 40                       | 2412                    | 2116          | 87.72803   | 0.294497 | 0.231848            | 0.367267 | 0.11841     | 0.217143      | 0.77551    | 0.339286      |
| C1,C2,C6             | C3,C4,C5            | 126                      | 2056                    | 1719          | 83.60895   | 0.144336 | 0.190317            | 0.339276 | 0.15992     | 0.106145      | 0.690909   | 0.184019      |
| C1,C3,C4             | C2,C5,C6            | 134                      | 469                     | 394           | 84.00853   | 0.419736 | 0.267584            | 0.363564 | 0.148325    | 0.38          | 0.745098   | 0.503311      |
| C1,C3,C5             | C2,C4,C6            | 76                       | 1760                    | 1490          | 84.65909   | 0.296209 | 0.223971            | 0.365727 | 0.155952    | 0.215569      | 0.9        | 0.347826      |
| C1,C3,C6             | C2,C4,C5            | 162                      | 1404                    | 1148          | 81.76638   | 0.11702  | 0.266789            | 0.399497 | 0.177762    | 0.089888      | 0.648649   | 0.157895      |
| C1,C4,C5             | C2,C3,C6            | 94                       | 1153                    | 1066          | 92.45447   | 0.474952 | 0.18298             | 0.279302 | 0.057301    | 0.425926      | 0.647887   | 0.513966      |
| C1,C4,C6             | C2,C3,C5            | 180                      | 797                     | 641           | 80.4266    | 0.135623 | 0.184088            | 0.331635 | 0.189857    | 0.109756      | 0.642857   | 0.1875        |
| C1,C5,C6             | C2,C3,C4            | 122                      | 2088                    | 1851          | 88.64943   | 0.212399 | 0.203732            | 0.307297 | 0.107829    | 0.151163      | 0.684211   | 0.247619      |
| C2,C3,C4             | C1,C5,C6            | 114                      | 724                     | 340           | 46.96133   | 0.109077 | 0.37483             | 0.511995 | 0.579186    | 0.137079      | 1          | 0.241107      |
| C2,C3,C5             | C1,C4,C6            | 56                       | 2015                    | 1190          | 59.05707   | 0.099323 | 0.30366             | 0.478169 | 0.426494    | 0.094818      | 0.955556   | 0.172518      |
| C2,C3,C6             | C1,C4,C5            | 142                      | 1659                    | 1510          | 91.01869   | 0.287948 | 0.258177            | 0.326622 | 0.084988    | 0.203488      | 0.744681   | 0.319635      |
| C2,C4,C5             | C1,C3,C6            | 74                       | 1408                    | 1294          | 91.90341   | 0.455371 | 0.213207            | 0.325116 | 0.067069    | 0.386207      | 0.691358   | 0.495575      |
| C2,C4,C6             | C1,C3,C5            | 160                      | 1052                    | 941           | 89.44867   | 0.279078 | 0.197705            | 0.346925 | 0.097633    | 0.208         | 0.684211   | 0.319018      |
| C2,C5,C6             | C1,C3,C4            | 102                      | 2343                    | 1989          | 84.89117   | 0.183866 | 0.117798            | 0.310267 | 0.148506    | 0.131105      | 0.761194   | 0.223684      |
| C3,C4,C5             | C1,C2,C6            | 110                      | 756                     | 562           | 74.33862   | 0.220362 | 0.290643            | 0.423834 | 0.255411    | 0.206278      | 0.730159   | 0.321678      |
| C3,C4,C6             | C1,C2,C5            | 196                      | 400                     | 273           | 68.25      | 0.156146 | 0.316705            | 0.472695 | 0.331579    | 0.131034      | 0.95       | 0.230303      |
| C3,C5,C6             | C1,C2,C4            | 138                      | 1691                    | 1591          | 94.08634   | 0.400321 | 0.233975            | 0.313693 | 0.053593    | 0.296         | 0.755102   | 0.425287      |
| C4,C5,C6             | C1,C2,C3            | 156                      | 1084                    | 880           | 81.18081   | 0.169436 | 0.225094            | 0.394238 | 0.184866    | 0.130631      | 0.725      | 0.221374      |
| C1,C2,C3,C4          | C5,C6               | 142                      | 410                     | 363           | 88.53659   | 0.534429 | 0.26696             | 0.386172 | 0.096419    | 0.5           | 0.744681   | 0.598291      |
| C1,C2,C3,C5          | C4,C6               | 84                       | 1701                    | 1406          | 82.65726   | 0.236796 | 0.238055            | 0.407226 | 0.172308    | 0.178886      | 0.802632   | 0.292566      |
| C1,C2,C3,C6          | C4,C5               | 170                      | 1345                    | 1211          | 90.03717   | 0.249267 | 0.161093            | 0.319831 | 0.096799    | 0.169935      | 0.787879   | 0.27957       |
| C1,C2,C4,C5          | C3,C6               | 102                      | 1094                    | 838           | 76.59963   | 0.214402 | 0.254604            | 0.395759 | 0.235638    | 0.179661      | 0.791045   | 0.292818      |
| C1,C2,C4,C6          | C3,C5               | 188                      | 738                     | 667           | 90.3794    | 0.230369 | 0.1857              | 0.31891  | 0.084034    | 0.178082      | 0.541667   | 0.268041      |
| C1,C2,C5,C6          | C3,C4               | 130                      | 2029                    | 1790          | 88.2208    | 0.142035 | 0.168699            | 0.318826 | 0.107287    | 0.109244      | 0.490566   | 0.178694      |
| C1,C3,C4,C5          | C2,C6               | 138                      | 442                     | 321           | 72.62443   | 0.248785 | 0.288158            | 0.427459 | 0.274809    | 0.25          | 0.734694   | 0.373057      |
| C1,C3,C4,C6          | C2,C5               | 224                      | 86                      | 65            | 75.5814    | 0.127536 | 0.268521            | 0.38969  | 0.225       | 0.142857      | 0.5        | 0.222222      |
| C1,C3,C5,C6          | C2,C4               | 166                      | 1377                    | 1239          | 89.97821   | 0.293788 | 0.153998            | 0.294249 | 0.101341    | 0.195266      | 0.942857   | 0.323529      |
| C1,C4,C5,C6          | C2,C3               | 184                      | 770                     | 673           | 87.4026    | 0.192872 | 0.17654             | 0.326352 | 0.115591    | 0.148515      | 0.576923   | 0.23622       |
| C2,C3,C4,C5          | C1,C6               | 118                      | 697                     | 487           | 69.87088   | 0.189813 | 0.304709            | 0.43355  | 0.30721     | 0.186722      | 0.762712   | 0.3           |
| C2,C3,C4,C6          | C1,C5               | 204                      | 341                     | 244           | 71.55425   | 0.167979 | 0.291944            | 0.42484  | 0.295385    | 0.135135      | 0.9375     | 0.23622       |
| C2,C3,C5,C6          | C1,C4               | 146                      | 1632                    | 1540          | 94.36275   | 0.408542 | 0.245898            | 0.294319 | 0.05167     | 0.299145      | 0.777778   | 0.432099      |
| C2,C4,C5,C6          | C1,C3               | 164                      | 1025                    | 831           | 81.07317   | 0.154276 | 0.236017            | 0.399141 | 0.185035    | 0.120192      | 0.694444   | 0.204918      |
| C3,C4,C5,C6          | C1,C2               | 200                      | 373                     | 273           | 73.19035   | 0.185696 | 0.310324            | 0.447603 | 0.278873    | 0.146552      | 0.944444   | 0.253731      |
| C1,C2,C3,C4,C5       | C6                  | 146                      | 383                     | 330           | 86.16188   | 0.514923 | 0.263387            | 0.378492 | 0.136095    | 0.452381      | 0.844444   | 0.589147      |
| C1,C2,C3,C4,C6       | C5                  | 232                      | 27                      | 20            | 74.07407   | 0.275862 | 0.294698            | 0.391835 | 0.28        | 0.222222      | 1          | 0.363636      |
| C1,C2,C3,C5,C6       | C4                  | 174                      | 1318                    | 1121          | 85.05311   | 0.187819 | 0.17848             | 0.351135 | 0.150738    | 0.126126      | 0.903226   | 0.221344      |
| C1,C2,C4,C5,C6       | C3                  | 192                      | 711                     | 626           | 88.04501   | 0.164535 | 0.179171            | 0.321497 | 0.107402    | 0.129412      | 0.5        | 0.205607      |
| C1,C3,C4,C5,C6       | C2                  | 228                      | 59                      | 52            | 88.13559   | 0.405755 | 0.259129            | 0.330261 | 0.109091    | 0.333333      | 0.75       | 0.461538      |
| C2,C3,C4,C5,C6       | C1                  | 208                      | 314                     | 237           | 75.47771   | 0.190017 | 0.267623            | 0.400004 | 0.253333    | 0.146067      | 0.928571   | 0.252427      |



## B.2.2 For Exit Classifier

| Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances | Total Correct | Correct(%) | Kappa    | Mean Absolute Error | RMS      | FP Rate (+) | Precision (+) | Recall (+) | F-Measure (+) |
|----------------------|---------------------|--------------------------|-------------------------|---------------|------------|----------|---------------------|----------|-------------|---------------|------------|---------------|
| C1                   | C2,C3,C4,C5,C6      | 64                       | 2498                    | 1535          | 61.44916   | 0.051061 | 0.346505            | 0.511907 | 0.390894    | 0.04985       | 0.833333   | 0.094073      |
| C2                   | C1,C3,C4,C5,C6      | 6                        | 2753                    | 89            | 3.232837   | 0        | 0.5                 | 0.5      | 1           | 0.032328      | 1          | 0.062632      |
| C3                   | C1,C2,C4,C5,C6      | 32                       | 2101                    | 1987          | 94.57401   | 0.496199 | 0.116215            | 0.210459 | 4.94E-02    | 0.382716      | 0.815789   | 0.521008      |
| C4                   | C1,C2,C3,C5,C6      | 32                       | 1494                    | 1356          | 90.76305   | 0.432277 | 0.143323            | 0.291083 | 0.087447    | 0.333333      | 0.815789   | 0.473282      |
| C5                   | C1,C2,C3,C4,C6      | 6                        | 2785                    | 89            | 3.195691   | 0        | 0.5                 | 0.5      | 1           | 0.031957      | 1          | 0.061935      |
| C6                   | C1,C2,C3,C4,C5      | 44                       | 2429                    | 2257          | 92.9189    | 0.375534 | 0.184251            | 0.270695 | 0.067825    | 0.266055      | 0.828571   | 0.402778      |
| C1,C2                | C3,C4,C5,C6         | 70                       | 2439                    | 2247          | 92.12792   | 0.325911 | 0.249637            | 0.316696 | 0.078505    | 0.217573      | 0.912281   | 0.351351      |
| C1,C3                | C2,C4,C5,C6         | 96                       | 1787                    | 1660          | 92.89312   | 0.361723 | 0.179005            | 0.26333  | 0.070568    | 0.245399      | 0.909091   | 0.386473      |
| C1,C4                | C2,C3,C5,C6         | 96                       | 1180                    | 1070          | 90.67797   | 0.385859 | 0.250795            | 0.378421 | 0.09331     | 0.273973      | 0.909091   | 0.421053      |
| C1,C5                | C2,C3,C4,C6         | 70                       | 2471                    | 2263          | 91.58236   | 0.307211 | 0.185939            | 0.291888 | 0.084093    | 0.203922      | 0.912281   | 0.333333      |
| C1,C6                | C2,C3,C4,C5         | 108                      | 2115                    | 1940          | 91.72577   | 0.26928  | 0.201342            | 0.321202 | 0.083293    | 0.172249      | 0.947368   | 0.291498      |
| C2,C3                | C1,C4,C5,C6         | 38                       | 2042                    | 1932          | 94.61312   | 0.492895 | 0.104832            | 0.210658 | 0.048756    | 0.380645      | 0.808219   | 0.517544      |
| C2,C4                | C1,C3,C5,C6         | 38                       | 1435                    | 1301          | 90.66202   | 0.426832 | 0.136182            | 0.293803 | 0.088106    | 0.329609      | 0.808219   | 0.468254      |
| C2,C5                | C1,C3,C4,C6         | 12                       | 2726                    | 2545          | 93.36023   | 0.412426 | 0.061419            | 0.225329 | 0.062879    | 0.299578      | 0.825581   | 0.439628      |
| C2,C6                | C1,C3,C4,C5         | 50                       | 2370                    | 2202          | 92.91139   | 0.368587 | 0.169127            | 0.274685 | 0.067738    | 0.260664      | 0.820896   | 0.395683      |
| C3,C4                | C1,C2,C5,C6         | 64                       | 783                     | 718           | 91.6986    | 0.54807  | 0.138287            | 0.266095 | 0.071923    | 0.474747      | 0.783333   | 0.591195      |
| C3,C5                | C1,C2,C4,C6         | 38                       | 2074                    | 1962          | 94.59981   | 0.488463 | 0.102722            | 0.217741 | 0.048976    | 0.375796      | 0.808219   | 0.513043      |
| C3,C6                | C1,C2,C4,C5         | 76                       | 1718                    | 1619          | 94.23749   | 0.439934 | 0.148931            | 0.241296 | 0.052885    | 0.328244      | 0.796296   | 0.464865      |
| C4,C5                | C1,C2,C3,C6         | 38                       | 1467                    | 1331          | 90.72938   | 0.423696 | 0.135579            | 0.292776 | 0.087518    | 0.325967      | 0.808219   | 0.464567      |
| C4,C6                | C1,C2,C3,C5         | 76                       | 1111                    | 988           | 88.92889   | 0.365759 | 0.190776            | 0.341511 | 0.10596     | 0.277419      | 0.796296   | 0.411483      |
| C5,C6                | C1,C2,C3,C4         | 50                       | 2402                    | 2232          | 92.92256   | 0.365949 | 0.169058            | 0.274523 | 0.067666    | 0.258216      | 0.820896   | 0.392857      |
| C1,C2,C3             | C4,C5,C6            | 102                      | 1728                    | 1671          | 96.70139   | 0.505632 | 0.193589            | 0.228418 | 0.02786     | 0.397436      | 0.756098   | 0.521008      |
| C1,C2,C4             | C3,C5,C6            | 102                      | 1121                    | 1018          | 90.81178   | 0.383425 | 0.234058            | 0.352665 | 0.091667    | 0.272059      | 0.902439   | 0.418079      |
| C1,C2,C5             | C3,C4,C6            | 40                       | 2412                    | 2116          | 87.72803   | 0.294497 | 0.231848            | 0.367267 | 0.11841     | 0.217143      | 0.77551    | 0.339286      |
| C1,C2,C6             | C3,C4,C5            | 114                      | 2056                    | 1871          | 91.00195   | 0.240701 | 0.219381            | 0.294435 | 0.090549    | 0.152778      | 0.942857   | 0.262948      |
| C1,C3,C4             | C2,C5,C6            | 128                      | 469                     | 448           | 95.52239   | 0.681169 | 0.190549            | 0.252725 | 0.040816    | 0.581395      | 0.892857   | 0.704225      |
| C1,C3,C5             | C2,C4,C6            | 102                      | 1760                    | 1609          | 91.42045   | 0.302306 | 0.151236            | 0.247754 | 0.085515    | 0.201087      | 0.902439   | 0.328889      |
| C1,C3,C6             | C2,C4,C5            | 140                      | 1404                    | 1349          | 96.08262   | 0.418909 | 0.206543            | 0.249938 | 0.039074    | 0.28          | 0.954545   | 0.43299       |
| C1,C4,C5             | C2,C3,C6            | 94                       | 1153                    | 1066          | 92.45447   | 0.474952 | 0.18298             | 0.279302 | 0.057301    | 0.425926      | 0.647887   | 0.513966      |
| C1,C4,C6             | C2,C3,C5            | 140                      | 797                     | 673           | 84.44166   | 0.215439 | 0.24803             | 0.352431 | 0.15871     | 0.145833      | 0.954545   | 0.253012      |
| C1,C5,C6             | C2,C3,C4            | 122                      | 2088                    | 1834          | 87.83525   | 0.212789 | 0.140532            | 0.297882 | 0.117676    | 0.149466      | 0.736842   | 0.248521      |
| C2,C3,C4             | C1,C5,C6            | 70                       | 724                     | 663           | 91.57459   | 0.546515 | 0.131768            | 0.279089 | 0.071964    | 0.478261      | 0.77193    | 0.590604      |
| C2,C3,C5             | C1,C4,C6            | 44                       | 2015                    | 1907          | 94.6402    | 0.484679 | 0.09418             | 0.22537  | 0.048329    | 0.373333      | 0.8        | 0.509091      |
| C2,C3,C6             | C1,C4,C5            | 82                       | 1659                    | 1564          | 94.27366   | 0.432416 | 0.139474            | 0.247167 | 0.052239    | 0.322581      | 0.784314   | 0.457143      |
| C2,C4,C5             | C1,C3,C6            | 44                       | 1408                    | 1276          | 90.625     | 0.411773 | 0.130991            | 0.307477 | 0.088191    | 0.321839      | 0.8        | 0.459016      |
| C2,C4,C6             | C1,C3,C5            | 82                       | 1052                    | 933           | 88.68821   | 0.355538 | 0.188607            | 0.317119 | 0.107892    | 0.27027       | 0.784314   | 0.40201       |
| C2,C5,C6             | C1,C3,C4            | 56                       | 2343                    | 2177          | 92.91507   | 0.358444 | 0.158124            | 0.28     | 0.067573    | 0.252427      | 0.8125     | 0.385185      |
| C3,C4,C5             | C1,C2,C6            | 70                       | 756                     | 693           | 91.66667   | 0.539559 | 0.132158            | 0.272362 | 0.071531    | 0.468085      | 0.77193    | 0.582781      |
| C3,C4,C6             | C1,C2,C5            | 108                      | 400                     | 350           | 87.5       | 0.462828 | 0.191641            | 0.350608 | 0.110497    | 0.411765      | 0.736842   | 0.528302      |
| C3,C5,C6             | C1,C2,C4            | 82                       | 1691                    | 1594          | 94.26375   | 0.42739  | 0.142225            | 0.239392 | 0.052439    | 0.31746       | 0.784314   | 0.451977      |
| C4,C5,C6             | C1,C2,C3            | 82                       | 1084                    | 963           | 88.83764   | 0.352545 | 0.186409            | 0.334147 | 0.106486    | 0.266667      | 0.784314   | 0.39801       |
| C1,C2,C3,C4          | C5,C6               | 134                      | 410                     | 300           | 73.17073   | 0.204445 | 0.284518            | 0.377077 | 0.277922    | 0.170543      | 0.88       | 0.285714      |
| C1,C2,C3,C5          | C4,C6               | 108                      | 1701                    | 1609          | 94.59142   | 0.40472  | 0.184769            | 0.234941 | 0.052916    | 0.278689      | 0.894737   | 0.425         |
| C1,C2,C3,C6          | C4,C5               | 146                      | 1344                    | 1289          | 95.90774   | 0.394991 | 0.090238            | 0.200752 | 0.041509    | 0.256757      | 1          | 0.408602      |
| C1,C2,C4,C5          | C3,C6               | 108                      | 1094                    | 994           | 90.85923   | 0.370946 | 0.226542            | 0.356313 | 0.090909    | 0.261538      | 0.894737   | 0.404762      |
| C1,C2,C4,C6          | C3,C5               | 188                      | 738                     | 667           | 90.3794    | 0.230369 | 0.185961            | 0.319183 | 0.084034    | 0.178082      | 0.541667   | 0.268041      |
| C1,C2,C5,C6          | C3,C4               | 120                      | 2028                    | 1844          | 90.92702   | 0.230919 | 0.12995             | 0.27404  | 0.091683    | 0.14486       | 0.96875    | 0.252033      |
| C1,C3,C4,C5          | C2,C6               | 134                      | 442                     | 419           | 94.79638   | 0.630515 | 0.201252            | 0.279779 | 0.047962    | 0.52381       | 0.88       | 0.656716      |
| C1,C3,C4,C6          | C2,C5               | 172                      | 86                      | 80            | 93.02326   | 0.632479 | 0.183235            | 0.275961 | 0.075       | 0.5           | 1          | 0.666667      |
| C1,C3,C5,C6          | C2,C4               | 146                      | 1377                    | 1269          | 92.15686   | 0.231593 | 0.16136             | 0.257608 | 0.078792    | 0.144         | 0.947368   | 0.25          |
| C1,C4,C5,C6          | C2,C3               | 148                      | 770                     | 720           | 93.50649   | 0.417108 | 0.203101            | 0.282066 | 0.060403    | 0.307692      | 0.8        | 0.444444      |
| C2,C3,C4,C5          | C1,C6               | 76                       | 697                     | 638           | 91.53515   | 0.537325 | 0.126542            | 0.291498 | 0.07154     | 0.471264      | 0.759259   | 0.58156       |
| C2,C3,C4,C6          | C1,C5               | 114                      | 341                     | 295           | 86.51026   | 0.448957 | 0.197241            | 0.357963 | 0.117647    | 0.409836      | 0.714286   | 0.520833      |
| C2,C3,C5,C6          | C1,C4               | 88                       | 1632                    | 1539          | 94.30147   | 0.41875  | 0.140399            | 0.228568 | 0.051768    | 0.310924      | 0.770833   | 0.443114      |
| C2,C4,C5,C6          | C1,C3               | 88                       | 1025                    | 908           | 88.58537   | 0.341242 | 0.183911            | 0.337032 | 0.108495    | 0.258741      | 0.770833   | 0.387435      |
| C3,C4,C5,C6          | C1,C2               | 114                      | 373                     | 325           | 87.13137   | 0.443006 | 0.192673            | 0.345687 | 0.112426    | 0.396825      | 0.714286   | 0.510204      |
| C1,C2,C3,C4,C5       | C6                  | 140                      | 383                     | 343           | 89.55614   | 0.454571 | 0.220948            | 0.36266  | 0.105263    | 0.344828      | 0.909091   | 0.5           |
| C1,C2,C3,C4,C6       | C5                  | 178                      | 27                      | 25            | 92.59259   | 0.709677 | 0.203277            | 0.286599 | 0.083333    | 0.6           | 1          | 0.75          |
| C1,C2,C3,C5,C6       | C4                  | 152                      | 1318                    | 1269          | 96.28225   | 0.3675   | 0.167342            | 0.2222   | 0.036866    | 0.238095      | 0.9375     | 0.379747      |
| C1,C2,C4,C5,C6       | C3                  | 152                      | 711                     | 593           | 83.40366   | 0.16936  | 0.245811            | 0.361451 | 0.168345    | 0.113636      | 0.9375     | 0.202703      |
| C1,C3,C4,C5,C6       | C2                  | 178                      | 59                      | 52            | 88.13559   | 0.415842 | 0.246369            | 0.348376 | 0.125       | 0.3           | 1          | 0.461538      |
| C2,C3,C4,C5,C6       | C1                  | 120                      | 314                     | 270           | 85.98726   | 0.425482 | 0.200602            | 0.34793  | 0.120567    | 0.392857      | 0.6875     | 0.5           |

## B.2.3 For Read Classifier

| Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances | Total Correct | Correct(%) | Kappa    | Mean Absolute Error | RMS      | FP Rate (+) | Precision (+) | Recall (+) | F-Measure (+) |
|----------------------|---------------------|--------------------------|-------------------------|---------------|------------|----------|---------------------|----------|-------------|---------------|------------|---------------|
| C1                   | C2,C3,C4,C5,C6      | 54                       | 2498                    | 1677          | 67.13371   | 0.049822 | 0.433803            | 0.489753 | 0.326159    | 0.058542      | 0.597561   | 0.106638      |
| C2                   | C1,C3,C4,C5,C6      | 4                        | 2753                    | 107           | 3.886669   | 0        | 0.5                 | 0.5      | 1           | 0.038867      | 1          | 0.074825      |
| C3                   | C1,C2,C4,C5,C6      | 38                       | 2101                    | 1383          | 65.8258    | 0.11611  | 0.25809             | 0.429888 | 0.352561    | 0.102532      | 0.9        | 0.184091      |
| C4                   | C1,C2,C3,C5,C6      | 34                       | 1494                    | 1302          | 87.14859   | 0.317691 | 0.312584            | 0.402933 | 0.112896    | 0.268519      | 0.630435   | 0.376623      |
| C5                   | C1,C2,C3,C4,C6      | 8                        | 2785                    | 105           | 3.770197   | 0        | 0.5                 | 0.5      | 1           | 0.037702      | 1          | 0.072664      |
| C6                   | C1,C2,C3,C4,C5      | 80                       | 2429                    | 1846          | 75.99835   | 0.125911 | 0.248198            | 0.396659 | 0.24322     | 0.094637      | 0.869565   | 0.170697      |
| C1,C2                | C3,C4,C5,C6         | 58                       | 2439                    | 1607          | 65.88766   | 0.050346 | 0.414613            | 0.514019 | 0.339975    | 0.058685      | 0.625      | 0.107296      |
| C1,C3                | C2,C4,C5,C6         | 92                       | 1787                    | 1474          | 82.48461   | 0.201612 | 0.256977            | 0.342263 | 1.75E-01    | 0.146893      | 0.825397   | 0.2494        |
| C1,C4                | C2,C3,C5,C6         | 88                       | 1180                    | 991           | 83.98305   | 0.312727 | 0.303005            | 0.392628 | 0.161435    | 0.237288      | 0.861538   | 0.372093      |
| C1,C5                | C2,C3,C4,C6         | 62                       | 2471                    | 1603          | 64.87252   | 0.046089 | 0.321873            | 0.458689 | 0.350606    | 0.05518       | 0.628205   | 0.101449      |
| C1,C6                | C2,C3,C4,C5         | 134                      | 2115                    | 1851          | 87.51773   | 0.19079  | 0.269209            | 0.349857 | 0.12494     | 0.125         | 0.880952   | 0.218935      |
| C2,C3                | C1,C4,C5,C6         | 42                       | 2042                    | 1617          | 79.18707   | 0.188757 | 0.182664            | 0.399534 | 0.208291    | 0.146751      | 0.795455   | 0.247788      |
| C2,C4                | C1,C3,C5,C6         | 38                       | 1435                    | 975           | 67.94425   | 0.136198 | 0.258676            | 0.440862 | 0.325651    | 0.134387      | 0.755556   | 0.228188      |
| C2,C5                | C1,C3,C4,C6         | 12                       | 2726                    | 2484          | 91.12252   | 0.308222 | 0.08149             | 0.263623 | 0.077392    | 0.2397        | 0.621359   | 0.345946      |
| C2,C6                | C1,C3,C4,C5         | 84                       | 2370                    | 1811          | 76.4135    | 0.122242 | 0.249035            | 0.433212 | 0.23795     | 0.092715      | 0.835821   | 0.166915      |
| C3,C4                | C1,C2,C5,C6         | 72                       | 783                     | 592           | 75.60664   | 0.27027  | 0.242537            | 0.370877 | 0.246479    | 0.24569       | 0.780822   | 0.37377       |
| C3,C5                | C1,C2,C4,C6         | 46                       | 2074                    | 1642          | 79.17068   | 0.18186  | 0.162212            | 0.350291 | 0.208249    | 0.141079      | 0.790698   | 0.239437      |
| C3,C6                | C1,C2,C4,C5         | 118                      | 1718                    | 1400          | 81.4901    | 0.132196 | 0.236199            | 0.385342 | 0.181055    | 0.10119       | 0.68       | 0.176166      |
| C4,C5                | C1,C2,C3,C6         | 42                       | 1467                    | 1218          | 83.02658   | 0.253435 | 0.213848            | 0.40931  | 0.159536    | 0.21147       | 0.670455   | 0.321526      |
| C4,C6                | C1,C2,C3,C5         | 114                      | 1111                    | 835           | 75.15752   | 0.180052 | 0.317487            | 0.458843 | 0.254013    | 0.143312      | 0.865385   | 0.245902      |
| C5,C6                | C1,C2,C3,C4         | 88                       | 2402                    | 1758          | 73.18901   | 0.106169 | 0.292619            | 0.401908 | 0.272144    | 0.082251      | 0.876923   | 0.150396      |
| C1,C2,C3             | C4,C5,C6            | 96                       | 1728                    | 1454          | 84.14352   | 0.177951 | 0.25578             | 0.315936 | 1.52E-01    | 0.136519      | 0.655738   | 0.225989      |
| C1,C2,C4             | C3,C5,C6            | 92                       | 1121                    | 535           | 47.72525   | 0.051397 | 0.395926            | 0.530219 | 0.542533    | 0.0816        | 0.809524   | 0.148256      |
| C1,C2,C5             | C3,C4,C6            | 66                       | 2412                    | 1560          | 64.67662   | 0.047962 | 0.306146            | 0.43215  | 0.353168    | 0.056064      | 0.644737   | 0.103158      |
| C1,C2,C6             | C3,C4,C5            | 138                      | 2056                    | 1580          | 76.84825   | 0.105394 | 0.309269            | 0.402549 | 0.235119    | 0.074219      | 0.95       | 0.137681      |
| C1,C3,C4             | C2,C5,C6            | 126                      | 469                     | 377           | 80.3838    | 0.290119 | 0.232493            | 0.377532 | 0.177305    | 0.278846      | 0.630435   | 0.386667      |
| C1,C3,C5             | C2,C4,C6            | 100                      | 1760                    | 1578          | 89.65909   | 0.250387 | 0.26135             | 0.331814 | 0.094062    | 0.187817      | 0.627119   | 0.289063      |
| C1,C3,C6             | C2,C4,C5            | 172                      | 1404                    | 1176          | 83.76068   | 0.109737 | 0.253689            | 0.336385 | 0.161477    | 0.074689      | 0.782609   | 0.136364      |
| C1,C4,C5             | C2,C3,C6            | 96                       | 1153                    | 796           | 69.03729   | 0.124883 | 0.360423            | 0.474503 | 0.313187    | 0.118557      | 0.754098   | 0.2049        |
| C1,C4,C6             | C2,C3,C5            | 168                      | 797                     | 573           | 71.8946    | 0.133056 | 0.352034            | 0.479897 | 0.290155    | 0.100402      | 1          | 0.182482      |
| C1,C5,C6             | C2,C3,C4            | 142                      | 2088                    | 1799          | 86.159     | 0.153819 | 0.292178            | 0.373428 | 0.138049    | 0.101587      | 0.842105   | 0.181303      |
| C2,C3,C4             | C1,C5,C6            | 76                       | 724                     | 461           | 63.67403   | 0.148715 | 0.379737            | 0.545948 | 0.37366     | 0.175676      | 0.732394   | 0.283379      |
| C2,C3,C5             | C1,C4,C6            | 50                       | 2015                    | 1592          | 79.00744   | 0.179804 | 0.162117            | 0.346616 | 0.209736    | 0.140127      | 0.785714   | 0.237838      |
| C2,C3,C6             | C1,C4,C5            | 122                      | 1659                    | 1291          | 77.81796   | 0.12996  | 0.255898            | 0.394623 | 0.222843    | 0.09799       | 0.8125     | 0.174888      |
| C2,C4,C5             | C1,C3,C6            | 46                       | 1408                    | 954           | 67.75568   | 0.129283 | 0.268368            | 0.458872 | 0.326778    | 0.129032      | 0.744186   | 0.219931      |
| C2,C4,C6             | C1,C3,C5            | 118                      | 1052                    | 802           | 76.23574   | 0.180849 | 0.31068             | 0.461305 | 0.240519    | 0.14539       | 0.82       | 0.246988      |
| C2,C5,C6             | C1,C3,C4            | 92                       | 2343                    | 1703          | 72.68459   | 0.09525  | 0.297852            | 0.423045 | 0.275877    | 0.076358      | 0.825397   | 0.139785      |
| C3,C4,C5             | C1,C2,C6            | 80                       | 756                     | 567           | 75         | 0.255161 | 0.247232            | 0.417931 | 0.25182     | 0.234513      | 0.768116   | 0.359322      |
| C3,C4,C6             | C1,C2,C5            | 152                      | 400                     | 294           | 73.5       | 0.188951 | 0.334865            | 0.454507 | 0.258856    | 0.188034      | 0.666667   | 0.293333      |
| C3,C5,C6             | C1,C2,C4            | 126                      | 1691                    | 1378          | 81.49024   | 0.118829 | 0.210055            | 0.329522 | 0.180547    | 0.091743      | 0.652174   | 0.160858      |
| C4,C5,C6             | C1,C2,C3            | 122                      | 1084                    | 746           | 68.81919   | 0.109973 | 0.363498            | 0.494935 | 0.315637    | 0.101648      | 0.770833   | 0.179612      |
| C1,C2,C3,C4          | C5,C6               | 130                      | 410                     | 349           | 85.12195   | 0.456942 | 0.245249            | 0.339832 | 0.142077    | 0.402299      | 0.795455   | 0.534351      |
| C1,C2,C3,C5          | C4,C6               | 104                      | 1701                    | 1399          | 82.24574   | 0.186892 | 0.242964            | 0.328195 | 0.177007    | 0.136499      | 0.807018   | 0.233503      |
| C1,C2,C3,C6          | C4,C5               | 176                      | 1345                    | 1181          | 87.80669   | 0.139267 | 0.267191            | 0.321964 | 0.120091    | 0.091429      | 0.761905   | 0.163265      |
| C1,C2,C4,C5          | C3,C6               | 100                      | 1094                    | 761           | 69.56124   | 0.128168 | 0.367166            | 0.488641 | 0.307246    | 0.121547      | 0.745763   | 0.209026      |
| C1,C2,C4,C6          | C3,C5               | 172                      | 738                     | 533           | 72.22222   | 0.134249 | 0.330773            | 0.439355 | 0.286713    | 0.100877      | 1          | 0.183267      |
| C1,C2,C5,C6          | C3,C4               | 146                      | 2029                    | 1737          | 85.60867   | 0.143159 | 0.275457            | 0.361235 | 0.143502    | 0.094937      | 0.833333   | 0.170455      |
| C1,C3,C4,C5          | C2,C6               | 134                      | 442                     | 389           | 88.00905   | 0.420177 | 0.223467            | 0.351403 | 0.09        | 0.409836      | 0.595238   | 0.485437      |
| C1,C3,C4,C6          | C2,C5               | 206                      | 86                      | 76            | 88.37209   | 0.494118 | 0.228553            | 0.323817 | 0.125       | 0.375         | 1          | 0.545455      |
| C1,C3,C5,C6          | C2,C4               | 180                      | 1377                    | 1284          | 93.24619   | 0.213099 | 0.272433            | 0.338716 | 0.064801    | 0.137255      | 0.736842   | 0.231405      |
| C1,C4,C5,C6          | C2,C3               | 176                      | 770                     | 548           | 71.16883   | 0.11464  | 0.336848            | 0.448786 | 0.296395    | 0.08642       | 1          | 0.159091      |
| C2,C3,C4,C5          | C1,C6               | 84                       | 697                     | 559           | 80.20086   | 0.275915 | 0.277366            | 0.392459 | 0.177778    | 0.267974      | 0.61194    | 0.372727      |
| C2,C3,C4,C6          | C1,C5               | 156                      | 341                     | 226           | 66.27566   | 0.172662 | 0.351864            | 0.500787 | 0.348387    | 0.181818      | 0.774194   | 0.294479      |
| C2,C3,C5,C6          | C1,C4               | 130                      | 1632                    | 1206          | 73.89706   | 0.096745 | 0.224066            | 0.381905 | 0.262594    | 0.077434      | 0.795455   | 0.141129      |
| C2,C4,C5,C6          | C1,C3               | 126                      | 1025                    | 776           | 75.70732   | 0.14923  | 0.307657            | 0.459037 | 0.242084    | 0.125461      | 0.73913    | 0.214511      |
| C3,C4,C5,C6          | C1,C2               | 160                      | 373                     | 242           | 64.87936   | 0.099913 | 0.36697             | 0.500627 | 0.348837    | 0.130435      | 0.62069    | 0.215569      |
| C1,C2,C3,C4,C5       | C6                  | 138                      | 383                     | 325           | 84.8564    | 0.360048 | 0.321738            | 0.409762 | 0.119534    | 0.359375      | 0.575      | 0.442308      |
| C1,C2,C3,C4,C6       | C5                  | 210                      | 27                      | 25            | 92.59259   | 0.756757 | 0.195489            | 0.274415 | 0.086957    | 0.666667      | 1          | 0.8           |
| C1,C2,C3,C5,C6       | C4                  | 184                      | 1318                    | 1033          | 78.37633   | 0.072606 | 0.223317            | 0.323427 | 0.217525    | 0.050336      | 0.882353   | 0.095238      |
| C1,C2,C4,C5,C6       | C3                  | 180                      | 711                     | 508           | 71.44866   | 0.114059 | 0.325144            | 0.441657 | 0.293353    | 0.085586      | 1          | 0.157676      |
| C1,C3,C4,C5,C6       | C2                  | 214                      | 59                      | 49            | 83.05085   | 0.241645 | 0.257313            | 0.347377 | 0.175439    | 0.166667      | 1          | 0.285714      |
| C2,C3,C4,C5,C6       | C1                  | 164                      | 314                     | 225           | 71.65605   | 0.151042 | 0.351281            | 0.49272  | 0.271777    | 0.170213      | 0.592593   | 0.264463      |

## B.2.4 For Write Classifier

| Training Document(s) | Testing Document(s) | Total Training Instances | Total Testing Instances | Total Correct | Correct(%) | Kappa    | Mean Absolute Error | RMS      | FP Rate (+) | Precision (+) | Recall (+) | F-Measure (+) |
|----------------------|---------------------|--------------------------|-------------------------|---------------|------------|----------|---------------------|----------|-------------|---------------|------------|---------------|
| C1                   | C2,C3,C4,C5,C6      | 12                       | 2498                    | 1728          | 69.17534   | 0.112573 | 0.40175             | 0.565977 | 0.314761    | 0.095579      | 0.860215   | 0.172043      |
| C2                   | C1,C3,C4,C5,C6      | 2                        | 2753                    | 98            | 3.559753   | 0        | 0.5                 | 0.5      | 1           | 0.035598      | 1          | 0.068748      |
| C3                   | C1,C2,C4,C5,C6      | 48                       | 2101                    | 1990          | 94.7168    | 0.503526 | 0.209982            | 0.279573 | 0.048371    | 0.3875        | 0.826667   | 0.52766       |
| C4                   | C1,C2,C3,C5,C6      | 52                       | 1494                    | 797           | 53.34672   | 0.074757 | 0.352843            | 0.482178 | 0.48487     | 0.086207      | 0.890411   | 0.157195      |
| C5                   | C1,C2,C3,C4,C6      | 27                       | 2785                    | 2686          | 96.44524   | 0        | 0.035548            | 0.188541 | 0           | 0             | 0          | 0             |
| C6                   | C1,C2,C3,C4,C5      | 84                       | 2429                    | 2288          | 94.19514   | 0.344823 | 0.100139            | 0.234068 | 0.052698    | 0.246988      | 0.719298   | 0.367713      |
| C1,C2                | C3,C4,C5,C6         | 14                       | 2439                    | 1831          | 75.07175   | 0.154206 | 0.392271            | 0.467209 | 0.254367    | 0.119469      | 0.880435   | 0.21039       |
| C1,C3                | C2,C4,C5,C6         | 60                       | 1787                    | 1702          | 95.24342   | 0.562976 | 0.274902            | 0.305529 | 0.044237    | 0.441176      | 0.869565   | 0.585366      |
| C1,C4                | C2,C3,C5,C6         | 64                       | 1180                    | 1111          | 94.15254   | 0.606042 | 0.290117            | 0.335122 | 0.055705    | 0.491803      | 0.895522   | 0.634921      |
| C1,C5                | C2,C3,C4,C6         | 12                       | 2471                    | 1707          | 69.08134   | 0.113121 | 0.402498            | 0.56677  | 0.315812    | 0.09627       | 0.860215   | 0.17316       |
| C1,C6                | C2,C3,C4,C5         | 96                       | 2115                    | 2000          | 94.56265   | 0.382225 | 0.157992            | 0.25063  | 0.049903    | 0.274648      | 0.764706   | 0.404145      |
| C2,C3                | C1,C4,C5,C6         | 50                       | 2042                    | 1932          | 94.61312   | 0.501243 | 0.214747            | 0.270219 | 0.049289    | 0.386076      | 0.824324   | 0.525862      |
| C2,C4                | C1,C3,C5,C6         | 54                       | 1435                    | 989           | 68.91986   | 0.154059 | 0.433941            | 0.556556 | 0.322817    | 0.130435      | 0.916667   | 0.228374      |
| C2,C5                | C1,C3,C4,C6         | 2                        | 2726                    | 98            | 3.595011   | 0        | 0.5                 | 0.5      | 1           | 0.03595       | 1          | 0.069405      |
| C2,C6                | C1,C3,C4,C5         | 86                       | 2370                    | 2230          | 94.09283   | 0.3404   | 0.100088            | 0.236093 | 0.053587    | 0.243902      | 0.714286   | 0.363636      |
| C3,C4                | C1,C2,C5,C6         | 100                      | 783                     | 700           | 89.39974   | 0.47295  | 0.22926             | 0.316097 | 0.107629    | 0.362903      | 0.918367   | 0.520231      |
| C3,C5                | C1,C2,C4,C6         | 48                       | 2074                    | 1382          | 66.63452   | 0.098928 | 0.267676            | 0.422104 | 0.341171    | 0.087015      | 0.866667   | 0.158151      |
| C3,C6                | C1,C2,C4,C5         | 132                      | 1718                    | 1467          | 85.38999   | 0.13658  | 0.157017            | 0.294044 | 0.144214    | 0.093284      | 0.757576   | 0.166113      |
| C4,C5                | C1,C2,C3,C6         | 52                       | 1467                    | 1064          | 72.52897   | 0.163582 | 0.258197            | 0.425879 | 0.281205    | 0.136564      | 0.849315   | 0.235294      |
| C4,C6                | C1,C2,C3,C5         | 136                      | 1111                    | 965           | 86.85869   | 0.193935 | 0.165185            | 0.335186 | 0.126852    | 0.138365      | 0.709677   | 0.231579      |
| C5,C6                | C1,C2,C3,C4         | 84                       | 2402                    | 2261          | 94.12989   | 0.344557 | 0.100526            | 0.239996 | 0.053305    | 0.246988      | 0.719298   | 0.367713      |
| C1,C2,C3             | C4,C5,C6            | 62                       | 1728                    | 1435          | 83.04398   | 0.236988 | 0.218854            | 0.31135  | 0.171084    | 0.172012      | 0.867647   | 0.287105      |
| C1,C2,C4             | C3,C5,C6            | 66                       | 1121                    | 1053          | 93.93399   | 0.604352 | 0.285817            | 0.333724 | 0.05782     | 0.491667      | 0.893939   | 0.634409      |
| C1,C2,C5             | C3,C4,C6            | 14                       | 2412                    | 1666          | 69.07131   | 0.114074 | 0.363776            | 0.491251 | 0.315948    | 0.097291      | 0.858696   | 0.174779      |
| C1,C2,C6             | C3,C4,C5            | 98                       | 2056                    | 1942          | 94.45525   | 0.377698 | 0.156988            | 0.251864 | 0.050847    | 0.271429      | 0.76       | 0.4           |
| C1,C3,C4             | C2,C5,C6            | 112                      | 469                     | 442           | 94.24307   | 0.721355 | 0.241985            | 0.290693 | 0.058685    | 0.621212      | 0.953488   | 0.752294      |
| C1,C3,C5             | C2,C4,C6            | 60                       | 1760                    | 1234          | 70.11364   | 0.116306 | 0.346541            | 0.485183 | 0.303962    | 0.099825      | 0.826087   | 0.178125      |
| C1,C3,C6             | C2,C4,C5            | 144                      | 1404                    | 1338          | 95.29915   | 0.358984 | 0.174099            | 0.244262 | 0.042847    | 0.253165      | 0.740741   | 0.377358      |
| C1,C4,C5             | C2,C3,C6            | 64                       | 1153                    | 1084          | 94.01561   | 0.605311 | 0.290777            | 0.336645 | 0.05709     | 0.491803      | 0.895522   | 0.634921      |
| C1,C4,C6             | C2,C3,C5            | 148                      | 797                     | 705           | 88.45671   | 0.265688 | 0.203102            | 0.322404 | 0.112694    | 0.186916      | 0.8        | 0.30303       |
| C1,C5,C6             | C2,C3,C4            | 96                       | 2088                    | 1973          | 94.49234   | 0.381931 | 0.161327            | 0.24566  | 0.050565    | 0.274648      | 0.764706   | 0.404145      |
| C2,C3,C4             | C1,C5,C6            | 102                      | 724                     | 627           | 86.60221   | 0.406236 | 0.254318            | 0.341096 | 0.134615    | 0.315789      | 0.875      | 0.464088      |
| C2,C3,C5             | C1,C4,C6            | 50                       | 2015                    | 1905          | 94.54094   | 0.500896 | 0.21041             | 0.274508 | 0.049974    | 0.386076      | 0.824324   | 0.525862      |
| C2,C3,C6             | C1,C4,C5            | 134                      | 1659                    | 1376          | 82.94153   | 0.103351 | 0.16524             | 0.330901 | 0.167793    | 0.074576      | 0.6875     | 0.134557      |
| C2,C4,C5             | C1,C3,C6            | 54                       | 1408                    | 1020          | 72.44318   | 0.165958 | 0.239268            | 0.403365 | 0.282186    | 0.139269      | 0.847222   | 0.239216      |
| C2,C4,C6             | C1,C3,C5            | 138                      | 1052                    | 907           | 86.21673   | 0.185602 | 0.18881             | 0.371105 | 0.133072    | 0.133758      | 0.7        | 0.224599      |
| C2,C5,C6             | C1,C3,C4            | 86                       | 2343                    | 2203          | 94.02475   | 0.340122 | 0.100479            | 0.246946 | 0.05422     | 0.243902      | 0.714286   | 0.363636      |
| C3,C4,C5             | C1,C2,C6            | 100                      | 756                     | 656           | 86.77249   | 0.405614 | 0.243493            | 0.350778 | 0.132956    | 0.313869      | 0.877551   | 0.462366      |
| C3,C4,C6             | C1,C2,C5            | 184                      | 400                     | 300           | 75         | 0.024723 | 0.251991            | 0.393017 | 0.244275    | 0.030303      | 0.428571   | 0.056604      |
| C3,C5,C6             | C1,C2,C4            | 132                      | 1691                    | 1599          | 94.55943   | 0.302924 | 0.145325            | 0.240837 | 0.048854    | 0.213592      | 0.666667   | 0.323529      |
| C4,C5,C6             | C1,C2,C3            | 136                      | 1084                    | 938           | 86.53137   | 0.192951 | 0.185708            | 0.362672 | 0.130104    | 0.138365      | 0.709677   | 0.231579      |
| C1,C2,C3,C4          | C5,C6               | 114                      | 410                     | 377           | 91.95122   | 0.675306 | 0.221165            | 0.29616  | 0.089674    | 0.56          | 1          | 0.717949      |
| C1,C2,C3,C5          | C4,C6               | 62                       | 1701                    | 1599          | 94.00353   | 0.509255 | 0.260553            | 0.315943 | 0.05695     | 0.388158      | 0.867647   | 0.536364      |
| C1,C2,C3,C6          | C4,C5               | 146                      | 1345                    | 1166          | 86.69145   | 0.146056 | 0.19464             | 0.297622 | 0.130402    | 0.099476      | 0.730769   | 0.175115      |
| C1,C2,C4,C5          | C3,C6               | 66                       | 1094                    | 1026          | 93.78428   | 0.603547 | 0.281536            | 0.33853  | 0.059339    | 0.491667      | 0.893939   | 0.634409      |
| C1,C2,C4,C6          | C3,C5               | 150                      | 738                     | 645           | 87.39837   | 0.250246 | 0.204112            | 0.310483 | 0.123249    | 0.17757       | 0.791667   | 0.290076      |
| C1,C2,C5,C6          | C3,C4               | 98                       | 2029                    | 1915          | 94.38147   | 0.37739  | 0.158841            | 0.249891 | 0.051541    | 0.271429      | 0.76       | 0.4           |
| C1,C3,C4,C5          | C2,C6               | 112                      | 442                     | 400           | 90.49774   | 0.623188 | 0.219168            | 0.292351 | 0.105263    | 0.505882      | 1          | 0.671875      |
| C1,C3,C4,C6          | C2,C5               | 196                      | 86                      | 85            | 98.83721   | 0.661417 | 0.13194             | 0.159723 | 0.011765    | 0.5           | 1          | 0.666667      |
| C1,C3,C5,C6          | C2,C4               | 144                      | 1377                    | 1112          | 80.75527   | 0.09894  | 0.212521            | 0.320099 | 0.191111    | 0.071942      | 0.740741   | 0.131148      |
| C1,C4,C5,C6          | C2,C3               | 148                      | 770                     | 720           | 93.50649   | 0.417108 | 0.203101            | 0.282066 | 0.060403    | 0.307692      | 0.8        | 0.444444      |
| C2,C3,C4,C5          | C1,C6               | 102                      | 697                     | 520           | 74.60545   | 0.235959 | 0.272459            | 0.379555 | 0.263482    | 0.197183      | 0.875      | 0.321839      |
| C2,C3,C4,C6          | C1,C5               | 186                      | 341                     | 266           | 78.00587   | 0.018724 | 0.27464             | 0.416357 | 0.21194     | 0.027397      | 0.333333   | 0.050633      |
| C2,C3,C5,C6          | C1,C4               | 134                      | 1632                    | 1541          | 94.42402   | 0.294788 | 0.147656            | 0.236457 | 0.05        | 0.207921      | 0.65625    | 0.315789      |
| C2,C4,C5,C6          | C1,C3               | 138                      | 1025                    | 880           | 85.85366   | 0.184522 | 0.186161            | 0.34857  | 0.136683    | 0.133758      | 0.7        | 0.224599      |
| C3,C4,C5,C6          | C1,C2               | 184                      | 373                     | 295           | 79.08847   | 0.038342 | 0.244969            | 0.368247 | 0.202186    | 0.038961      | 0.428571   | 0.071429      |
| C1,C2,C3,C4,C5       | C6                  | 114                      | 383                     | 364           | 95.03916   | 0.780344 | 0.249898            | 0.293666 | 0.049853    | 0.701754      | 0.952381   | 0.808081      |
| C1,C2,C3,C4,C6       | C5                  | 198                      | 27                      | 27            | 100        | 1        | 0.157895            | 0.157895 | 0           | 0             | 0          | 0             |
| C1,C2,C3,C5,C6       | C4                  | 146                      | 1318                    | 1253          | 95.06829   | 0.349753 | 0.172227            | 0.251128 | 0.044892    | 0.246753      | 0.730769   | 0.368932      |
| C1,C2,C4,C5,C6       | C3                  | 150                      | 711                     | 678           | 95.35865   | 0.499712 | 0.195552            | 0.259056 | 0.039301    | 0.4           | 0.75       | 0.521739      |
| C1,C3,C4,C5,C6       | C2                  | 196                      | 59                      | 58            | 98.30508   | 0.65896  | 0.138092            | 0.17199  | 0.017241    | 0.5           | 1          | 0.666667      |
| C2,C3,C4,C5,C6       | C1                  | 186                      | 314                     | 262           | 83.43949   | 0.038625 | 0.241409            | 0.359185 | 0.155844    | 0.04          | 0.333333   | 0.071429      |

## Appendix C

# Significance of Linguistic Features

In this appendix, we present some of the important key features of the noun phrases resulting from their syntactic relationships with other words in the software requirements sentences that often help to express data-movements in terms of the COSMIC FSM. Our automated data-movement classification approaches determine many of these features by using syntactic parsers and our lexical databases, as discussed in Section 7.4.

### C.1 “Noun Phrase is a Direct Object”

This is a binary feature indicating if the noun-phrase appears as a direct object to any verb. That is, if the head of a noun phrase (i.e. the head of a compound noun or a pronoun itself) appears as a direct object to any verb in the sentence, we record the value of this feature as “true”, or “false” otherwise. Moreover, if the head of a noun phrase appears as a syntactic nominal subject to a verb in passive form, we record the value of this feature for that noun phrase as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 44, the noun-phrase “course offering” is a direct object. Thus, the value of this feature here for the noun-phrase “course offering” would be “true”.

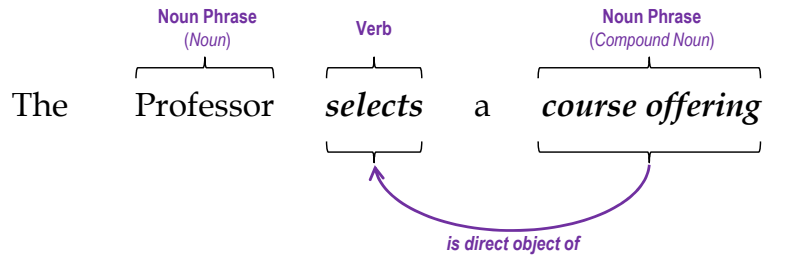


Figure 44: Example of the Feature: “Noun Phrase is a Direct Object”.

In relation to COSMIC data-movements, the noun-phrases that indicate moving data-attributes/data-groups often appear as direct object to the verb that expresses the action of the data-movement. In the above example, the verb “selects” expresses the data-movement of Entry type has occurred on the noun-phrase “course offering”, which appeared as a direct object to the verb “selects”.

## C.2 “Noun Phrase is in an Object-Like Position”

This is a binary feature indicating if the noun-phrase appears as a direct or indirect object to any verb, or as a prepositional object to a direct/indirect object of any verb, or in “a chain of prepositional objects” linked to a direct/indirect object of any verb.

In our work, we identify a noun-phrase to appear as a *prepositional object* to any verb or another noun phrase, when the first noun-phrase follows a preposition and that preposition serves to modify (or complement) the meaning of the verb or the second noun-phrase. For example, in Figure 45, the noun-phrase “Fund Usage” is a prepositional object to the noun-phrase “overview page”.

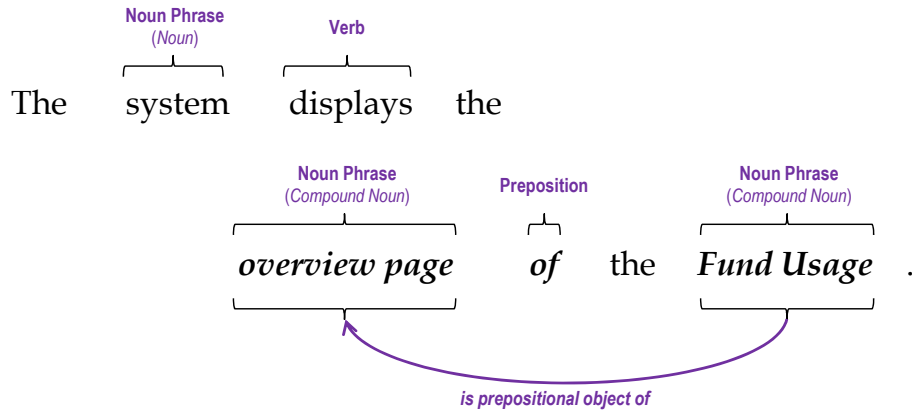


Figure 45: Example of a Prepositional Object.

Here, in Figure 45, the noun phrase “Fund Usage” follows the preposition “of” and modifies the meaning of the noun phrase “overview page”.

Also, when a noun-phrase appears as a prepositional object to another noun-phrase which may be a prepositional object to another noun phrase and so on, we then define the boundary, starting from the start of the first noun phrase and ending at the end of the last noun phrase, as a *chain of prepositional objects*. In such a chain, there are more than one prepositional objects linked to one another. For example, in the sentence shown in Figure 46, we call the segment “list of names of all students in the class” as a chain of prepositional objects.

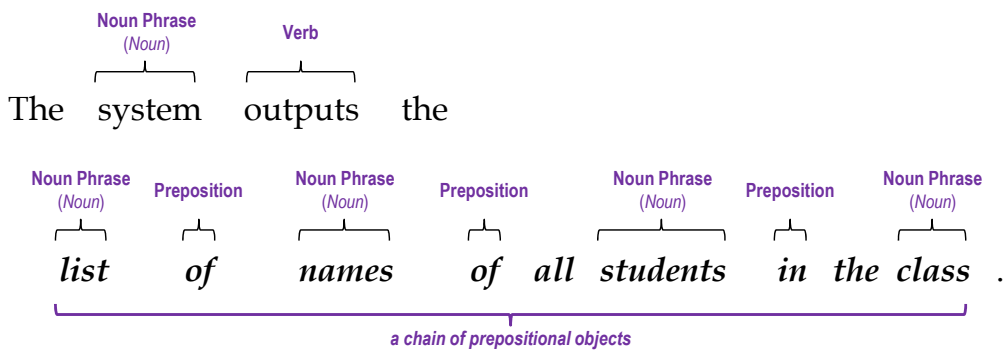


Figure 46: Example of a Chain of Prepositional Objects.

Thus, for our work, we identify a noun-phrase to appear in an *object like position*, if the head<sup>20</sup> of a noun phrase appears either:

- (i) As a direct or indirect object to any verb,
- (ii) Or, as a prepositional object to a direct/indirect object of any verb,
- (iii) Or, in a “chain of prepositional objects” linked to a direct/indirect object of any verb
- (iv) Or, a syntactic nominal subject to a verb in passive form
- (v) Or, as a prepositional object to a noun phrase that appears as a syntactic nominal subject to a verb in passive form
- (vi) Or, in a chain of prepositional objects linked to a noun phrase that appears as a syntactic nominal subject to a verb in passive form

Therefore, if the noun-phrase appears in an object like position we record the value of the feature, described in Appendix C.2, as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 47, the underlined noun-phrases “list”, “names”, “students” and “class” are all in object-like positions to the verb “outputs”. Here, the noun phrase “list” is the direct object to the verb “outputs”; then the noun phrase “names” is a prepositional object to the noun phrase “list”; and finally, the noun phrases “list”, “names”, “students” and “class” are all in a chain of prepositional objects that are linked to the noun phrase “list”. Thus, by the rules (i), (ii) and (iii) listed above, the value of this feature here for each of the noun phrases would be “true”.

---

<sup>20</sup> The head of a noun-phrase is selected as follows: (i) if the noun-phrase consists of only one noun, then the head is the noun itself; (ii) if the noun-phrase consists of a compound noun, then the head is the rightmost noun; and (iii) if the noun-phrase consists of only one pronoun, then the head is the pronoun itself.

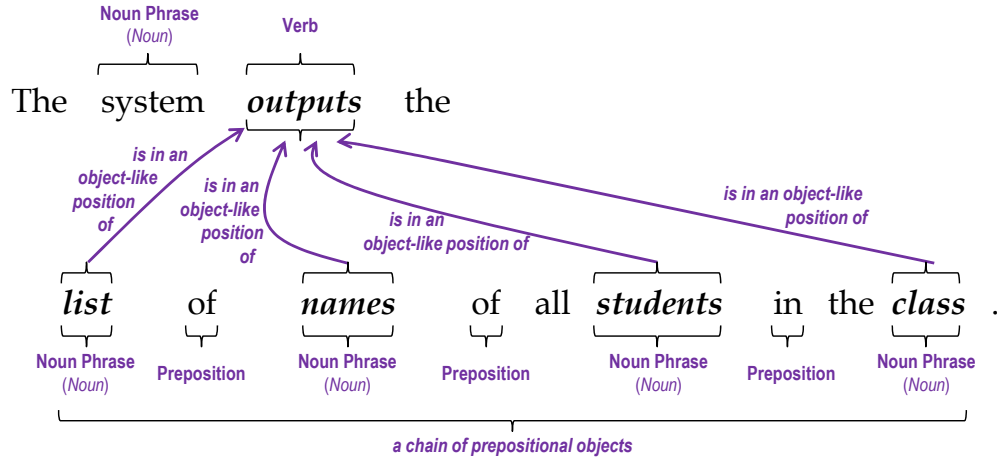


Figure 47: Example of the Feature: “Noun Phrase is in an Object-Like Position”.

In relation to the COSMIC data-movements, the noun-phrases that indicate moving data-attributes or data-groups cannot only appear as direct/indirect object to the verb that expresses the action of the data-movement, but also as a prepositional object or in a chain of prepositional objects linked to a direct/indirect object of such verb. In the second sentence of the above example, shown in Figure 47, the verb “outputs” expresses that the data-movement of *Exit* type has occurred on the noun-phrase “names”, which did not appear as its direct object, but as a prepositional object to the direct object of the verb.

### C.3 “Noun Phrase is a Subject”

This is a binary feature indicating if the noun-phrase appears as a subject to any verb. That is, if the head of a noun phrase (i.e. the head of a compound noun or a pronoun itself) appears as subject to any verb in the sentence, we record the value of this feature as “true”, or “false” otherwise. Moreover, if the head of a noun phrase appears as a prepositional object with “by” preposition to a verb in passive form, we record the value of this feature for that noun phrase as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 48, the noun-phrase “Professor” is a subject. Thus, the value of this feature here for the noun phrase “Professor” would be “true”.



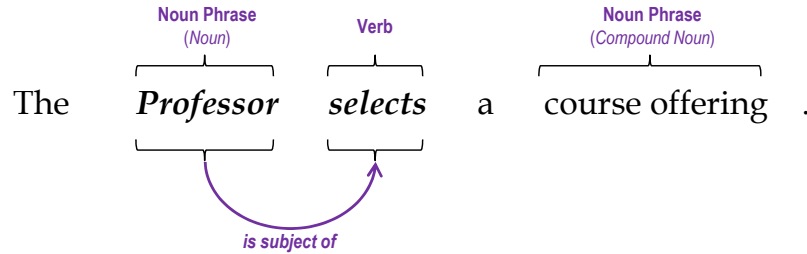


Figure 48: Example of the Feature: “Noun Phrase is a Subject”.

In relation to the COSMIC data-movements, the noun-phrases that indicate moving data-attributes/data-groups rarely appear as subjects to the verbs that express the action of the data-movement. In the above example, the verb “selects” expresses the data-movement of Entry type has occurred, but the moving data-attribute/data-group is not the noun-phrase “Professor”, which appeared as a subject to the verb “selects”.

#### C.4 “Noun Phrase is in an Object-Like Position to a Verb with No Subject”

This is a binary feature indicating if the noun-phrase appears as a direct or indirect object, or as a prepositional object to a direct/indirect object, or in a chain of prepositional objects linked to a direct/indirect object of any verb that has no subject. This feature is similar to the feature described in Appendix C.2. However, it adds an additional constraint that the associated verb cannot have a subject of its own.

Thus, if a noun phrase appears in an *object-like position* (Appendix C.2 describes the details on how we define the object-like position in our work) to any verb that has no subject, we record the value of this feature as “true”, or “false” otherwise.

Moreover, if a noun-phrase the head of a noun phrase appears in an object-like position to a verb in passive form and the verb does not have as a prepositional object with “by” preposition, we record the value of this feature as “true”, or “false” otherwise.

For example, in the sentence: “System logs the action of modifying the course information.”, the noun-phrase “course information” appears in an object-like position to the verb “modify” that has no subject. Hence, the value of this feature here would be “true”.

In relation to the COSMIC data-movements, this feature suggests insufficiency of information. That is, the noun-phrases that indicate moving data-attributes or data-groups should appear in object-like positions to verbs that have subjects clearly specified. This is because the subject in most cases acts as an agent to the action that a verb refers to and, thus, a clearly specified subject helps to deduce a specific kind of data-movement action that may be indicated by the verb. For example, the human actors often appear as subjects to Entry data-movements, while the non-human actors often appear as subjects to *Exit*, *Read* and *Write* data-movements. Thus, when a subject is absent for a verb, as in the case of the above example sentence, it usually implies the dependency on other features in identifying the kind of movement of the data-attribute or data-group that may be referred to by the noun-phrase in the object-like position of the verb.

### **C.5 “Noun Phrase is in an Object-Like Position to a Verb with an Actor Subject”**

This is a binary feature indicating if the noun-phrase appears as a direct or indirect object, or as a prepositional object, or in a chain of prepositional objects linked to a direct/indirect object of any verb that has a subject, which is identified as a mention of a potential *Actor*. The feature is similar to the feature described in Appendix C.2. However, it adds an additional constraint that the associated verb must have a subject that is identified as a mention of a potential *Actor*. This feature, therefore, requires the use of the lexical knowledge stored in our vocabulary of actors, as presented in Appendix A.5.

We identify a noun-phrase as an *Actor* subject, if the noun-phrase appears:

- (i) As a subject to a verb and the stemmed head of the noun-phrase contains a sub-string that exists in our vocabulary of actors.

- (ii) Or, as a prepositional object with “by” preposition to a verb in passive form and the stemmed head of the noun-phrase contains a sub-string that exists in our vocabulary of actors.

Therefore, if a noun phrase appears in an *object-like position* (please see Appendix C.2) for details on how we define the object-like position in our work) to any verb that has an Actor subject, we record the value of the feature, described in Appendix C.5, as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 49, the noun-phrase “course offering” appears in an object-like position to the verb “select” with an Actor subject “Professor”. Thus, the value of this feature here would be “true”.

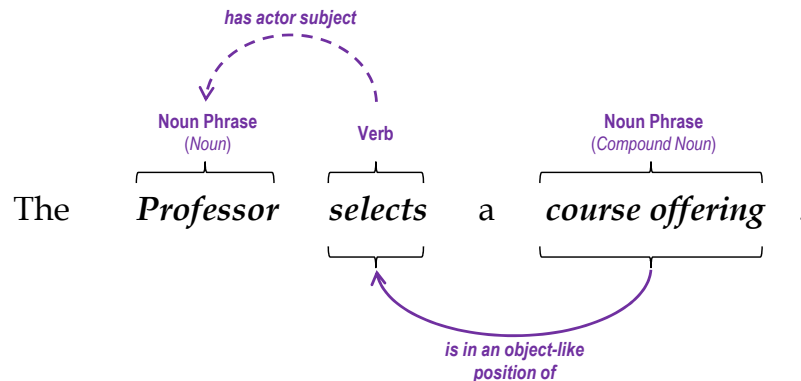


Figure 49: Example of the Feature: “Noun Phrase is in an Object-Like Position of a Verb with an Actor Subject”.

In relation to the COSMIC data-movements, the noun-phrases that indicate data-attributes/data-groups participating in Entry data-movements tend to appear in object-like positions to verbs that have actor subjects. Thus, when the subject is an actor, as in the case of the above example shown in Figure 49, some of the noun-phrases that appear in object-like positions, e.g. the noun phrase “course offering” in the above example, often indicate data-attributes or data-groups participating in Entry data-movements.

## C.6 “Noun Phrase is in an Object-Like Position to a Verb with a Non-Actor Subject”

This is a binary feature indicating if the noun-phrase appears as a direct or indirect object, or as a prepositional object, or in a chain of prepositional objects linked to a direct/indirect object of any verb that has a subject, which is identified as a mention of a potential *Non-Actor*. The feature is similar to the feature described in Appendix C.2. However, it adds an additional constraint that the associated verb must have a subject that is identified as a mention of a potential *Non-Actor*. This feature, therefore, requires the use of the lexical knowledge stored in our vocabulary of actors, as presented in Appendix A.5.

We identify a noun-phrase as a *Non-Actor* subject, if the noun-phrase appears:

- (i) As a subject to a verb and the stemmed head of the noun-phrase does not contain a sub-string that exists in our vocabulary of actors.
- (ii) Or, as a prepositional object with “by” preposition to a verb in passive form and the stemmed head of the noun-phrase does not contain a sub-string that exists in our vocabulary of actors.

Therefore, if a noun phrase appears in an *object-like position* (please see Appendix C.2 for details on how we define the object-like position in our work) to any verb that has a Non-Actor subject, we record the value of the feature, described in Appendix C.6, as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 50, the noun-phrases “list”, “names”, “students” and “class” appear in object-like positions to the verb “output” with a Non-Actor subject “system”. Thus, the value of this feature here for each of the noun-phrases would be “true”.

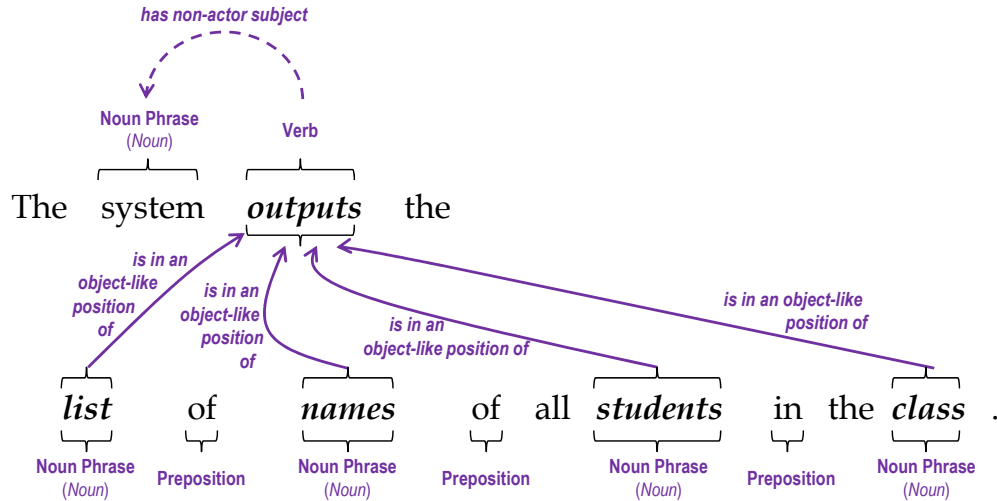


Figure 50: Example of the Feature: “Noun Phrase is in an Object-Like Position of a Verb with Non-Actor Subject”.

In relation to the COSMIC data-movements, the noun-phrases that indicate data-attributes/data-groups participating in Exit, Read or Write data-movements tend to appear in object-like positions to verbs that have non-actor subjects. Thus, when the subject is a non-actor, as in the case of the above example, shown in Figure 50, some of the noun-phrases that appear in object-like positions, e.g. the noun phrase “names”, “students” and “class” in the example, often indicate data-attributes or data-groups participating in Exit, Read or Write data-movements. In this example, it indicates an Exit data-movement.

## C.7 “Noun Phrase is a Subject of a Stative verb”

This is a binary feature indicating if the noun-phrase appears as a subject to any *Stative* verb. Thus, it requires the use of our vocabulary of *Stative* verbs, as presented in Appendix A.2.

Therefore, if the head of a noun phrase (i.e. the head of a compound noun or a pronoun itself) appears as subject to a verb, and the morphologically stemmed form of the verb exists in our vocabulary of *Stative* verbs, we record the value of this feature as “true”, or “false” otherwise.

For example, in the sentence: “Student checks if the credit card number is valid.”, the noun-phrase “credit card number” is a subject of the Stative verb “is” (i.e. “be”). Hence, the value of this feature here would be “true”.

In relation to the COSMIC data-movements, the noun-phrases that indicate moving data-attributes/data-groups may appear as subjects to the Stative verbs, if they are part of clausal complements to verbs that express the action of data-movements. Thus, in the case of the above example sentence, the noun-phrase “credit card number” appears as a subject of a Stative verb and is part of the clausal complement to a data-movement verb “check”. Therefore, it indicates the movement of a data-attribute/data-group.

## **C.8 “Noun Phrase is in an Object-Like Position to a Data-Movement Verb”**

This is a binary feature indicating if the noun-phrase appears as a direct or indirect object, or as a prepositional object, or in a chain of prepositional objects linked to a direct/indirect object of a *Data-Movement verb*. The feature is similar to the feature described in Appendix C.2. However, it adds an additional constraint that the associated verb must be a *Data-Movement verb*. This feature, therefore, requires the use of the lexical knowledge stored in our vocabulary of data-movement verbs, as presented in Appendix A.1.

Therefore, if a noun phrase appears in an *object-like position* (please see Appendix C.2 for details on how we define the object-like position in our work) to a verb, and the morphologically stemmed form of the verb exists in our vocabulary of data-movement verbs, we record the value of feature#8 as “true”, or “false” otherwise.

For example, in the sentence shown in Figure 51, the noun-phrase “course offering” appears in an object-like position to a data-movement verb “selects”. Thus, the value of this feature here would be “true”.

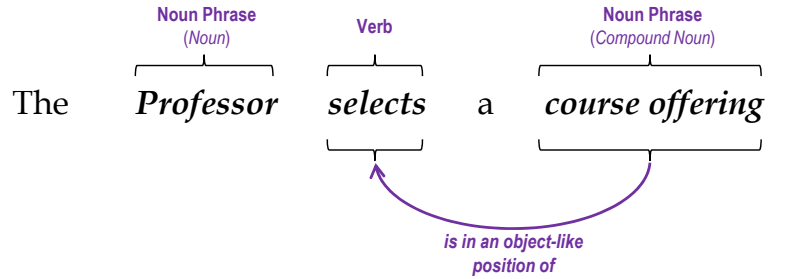


Figure 51: Example of the feature: "Noun Phrase is in an Object-Like Position of a Data-Movement Verb".

In relation to the COSMIC data-movements, the noun-phrases that indicate moving data-attributes/data-groups tend to appear in object-like positions to data-movement verbs. Thus, when the verb is a data-movement verb, as in the case of the above example, shown in Figure 51, the noun-phrases, e.g. "course offering" in the example, often indicate moving data-attributes/data-groups.

## C.9 "Noun Phrase Partly Contains an Attribute Name"

We identify a noun-phrase as a potential mention of an attribute, if the stemmed head of the noun-phrase (i.e. the head of the compound noun, in our case) exists in our vocabulary of data-attributes, as presented in Appendix A.3. Thus, if the noun-phrase is a potential mention of a data-attribute, we record this feature for the noun-phrase as "true", or "false" otherwise.

For example, in the sentence shown in Figure 52, the noun-phrase "names" refers to a mention of an attribute, as the lexicon "name", and its stemmed head exists in our vocabulary of data-attributes, as presented in Appendix A.3. Therefore, this feature for the noun phrase "names" will be "true".

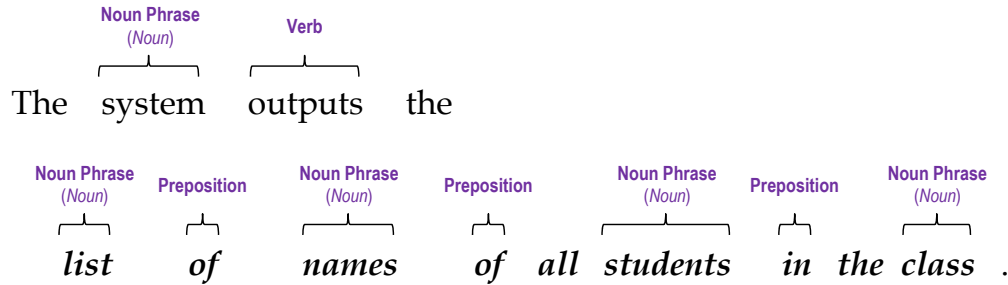


Figure 52: Example of the Feature: "Noun Phrase is a Mention of an Attribute".

In relation to the COSMIC data-movements, functional requirements, when they are well-decomposed, tend to describe the movements of the members, or the properties, or the attributes of the data-groups by using mentions of the data-attributes that belong to the moving data-groups. In requirement sentences, where mentions of both data-groups and data-attributes are present, the mentions of data-attributes are usually attached to the semantics of movement, while the mentions of data-groups help to convey the indication of the owners of the attributes. Thus, in the example shown in Figure 52, the noun-phrase “names” indicates the moving data-attribute, while the noun-phrase “students” is the data-group that contains the moving data-attribute.

### C.10 “Noun Phrase Partly Contains a Data-Group Name”

We identify a noun-phrase as a mention of a data-group, if the noun-phrase is not an attribute (see the feature description, presented in Appendix C.2) and the noun-phrase contains a substring that exists in our vocabulary of data-groups, as presented in Appendix A.4. Thus, if the noun-phrase is a mention of a data-group, we record this feature for the noun-phrase as “true”, or “false” otherwise.



For example, in the sentence shown in

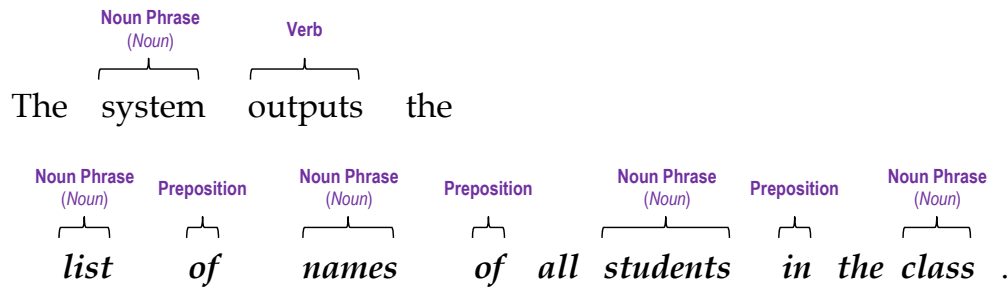


Figure 53, the noun-phrases “students” and “class” refer to the data-groups “Student” and “Class” respectively. Therefore, the values of this feature for both the noun-phrases will be “true”.

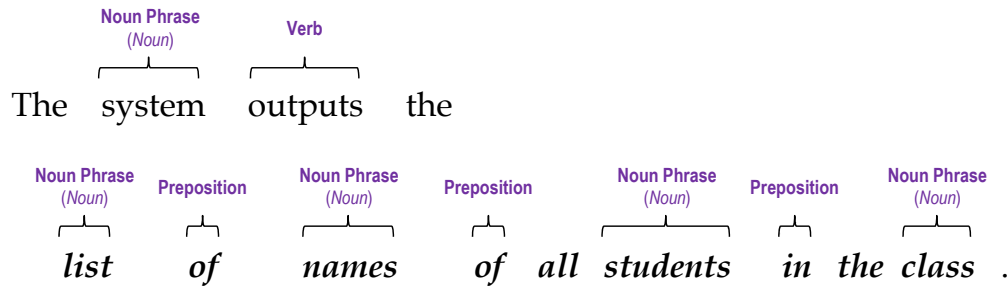


Figure 53: Example of the Feature: "Noun Phrase is a Mention of a Data-Group".

In relation to the COSMIC data-movements, functional requirements, usually when they are not well-decomposed, often describe the movements of the data-groups by using the mentions of the data-groups directly, without mentioning which specific attributes are to move. In such cases, for example, in the sentence shown in

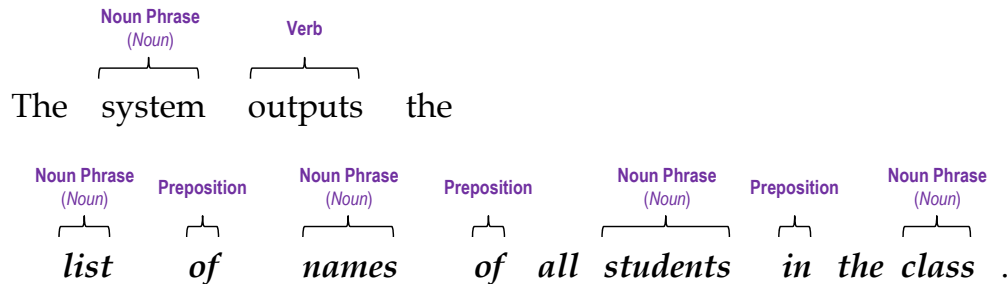


Figure 53, a noun-phrase, e.g. “class” in the example, can indicate a moving data-group without explicitly mentioning which attribute(s) belonging to the data-group participate in the data-movement.

## C.11 “Noun Phrase is Related to an Attribute Name by a Chain of Prepositional Objects”

Here, by our explanations presented in Appendices C.2 and C.9 about the chain of prepositional objects and the mentions of attributes respectively, this feature name is self-explanatory.

For example, in the sentence shown in

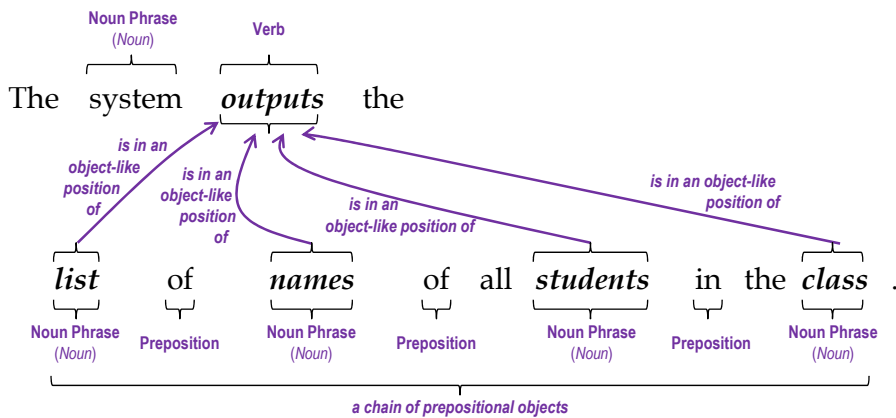


Figure 54, the noun-phrases “list”, “students”, and “class” are all related to the attribute “names” by a chain of prepositional objects. Therefore, the values of this feature for the noun-phrases “list”, “students”, and “class” will all be “true”.

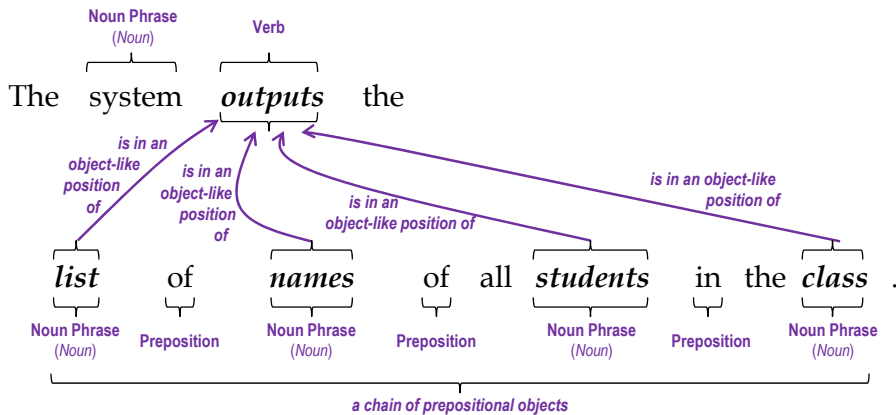
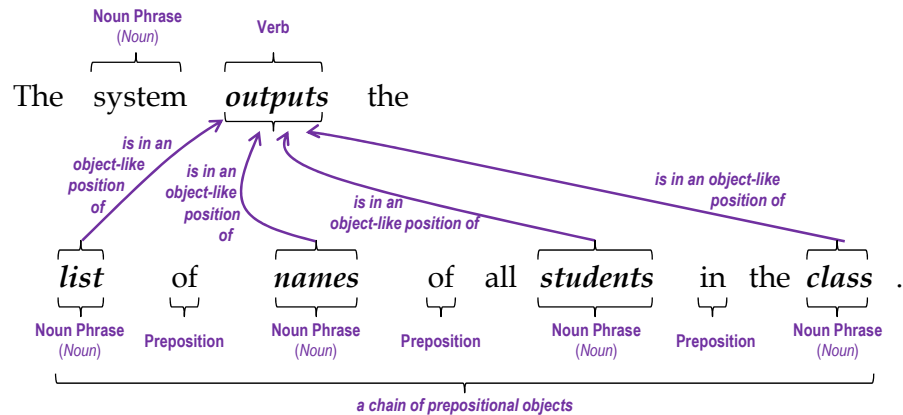


Figure 54: Example of the Feature: "Noun Phrase is Related to an Attribute Name by a Chain of Prepositional Objects".

In relation to the COSMIC data-movements, the mentions of data-attributes in a functional requirement sentence often carry the sense of the moving data-groups. In such cases, noun-phrases that are related to the data-attributes by chains of prepositional objects may not indicate moving data-groups, even though they contain other feature values leading to conclude otherwise. Thus, although the noun-phrases “list” and “students” in the example



shown in

Figure 54 are in object-like positions to the data-movement verb “outputs”, none of them are indicating any moving data-group as they are related to the mention of the data-attribute “names” by a chain of prepositional objects.

## C.12 “Noun Phrase is Related to a Data-Group Name by a Chain of Prepositional Objects”

The feature is similar to the feature, described in Appendix C.11, where we however consider data-groups instead of the attributes.

## C.13 “Noun Phrase Owns an Attribute”

In Appendix C.9, we described how we identify an attribute.

We record the feature “Noun Phrase Owns an Attribute” as “true”, if:

- (i) The attribute appears as a possessive determiner of the head of the noun-phrase. For example, in “Device User’s address”, the attribute “address” appears as a possessive determiner of the head “User” of the noun-phrase “Device User”. Therefore, this feature for the noun-phrase “Device User” will be “true”.
- (ii) The noun-phrase appears as a prepositional object of the attribute with an associated preposition “of”. For example, in “address of the Device User”, the noun-phrase “Device User” appears as a prepositional object of the attribute “address” with an associated preposition “of”. Therefore, this feature for the noun-phrase “Device User” will be “true”.

In relation to the COSMIC data-movement, the noun-phrases in functional requirements that owns attribute are often mentions of moving data-groups.

## **C.14 “Noun Phrase Belongs To A Data-Group”**

In Appendix C.10, we described how we identify a data-group.

We record the feature “Noun Phrase Belongs To Data-Group” as “true”, if:

- (i) The noun-phrase appears as a possessive determiner of the mention of the data-group. For example, in “Item’s price”, the noun-phrase “price” appears as a possessive determiner of the mention of the data-group “Item”. Therefore, this feature for the noun-phrase “price” will be “true”.
- (ii) The mention of the data-group appears as a prepositional object of the noun-phrase with an associated preposition “of”. For example, in “price of the Item”, the mention of the data-group “Item” appears as a prepositional object of the noun-phrase “price” with an associated preposition “of”. Therefore, this feature for the noun-phrase “price” will be “true”.

In relation to the COSMIC data-movement, the noun-phrases in functional requirement that belong to data-groups are often attributes.

## **C.15 “Noun Phrase is Part of a Negative Expression”**

We identify a noun-phrase as part of a negative expression, if:

- (i) The noun-phrase is “none”.
- (ii) Or, the noun-phrase appears in a chain of prepositional objects that is linked to “none”.
- (iii) Or, a negation modifier modifies the noun-phrase.
- (iv) Or, a negation modifier modifies a verb, and the noun-phrase appears as the subject of the verb.
- (v) Or, a negation modifier modifies a verb, and the noun-phrase appears in a chain of prepositional objects linked to the verb.
- (vi) Or, a negation modifier modifies a verb, and the noun-phrase appears in a chain of prepositional objects linked to the subject of the verb.
- (vii) Or, a negation modifier modifies a verb, and the noun-phrase appears in an object-like position to the verb.
- (viii) Or, the noun-phrase appears in the same clause as the negation modifiers: “no”, “not”, “n’t”, “neither”, “nor”, or “never”.
- (ix) Or, the noun-phrase appears as a subject to the negative implicative verbs (Karttunen, 1971), e.g. “fail”, “reject”, “refuse”, “deny”, “cancel” etc.
- (x) Or, the noun-phrase appears in a chain of prepositional objects to a subject to the negative implicative verbs (Karttunen, 1971), e.g. “fail”, “reject”, “refuse”, “deny”, “cancel” etc.

- (xi) Or, the noun-phrase appears in an object-like position to a verb, which appears in open clausal complement of the negative implicative verbs (Karttunen, 1971), e.g. “fail”, “reject”, “refuse”, “deny”, “cancel” etc.

In any of the above cases, we record the value of this feature as “true”, or “false” otherwise.

In relation to the COSMIC data-movements, the noun-phrases that are part of negative expressions do not usually convey the sense of a moving data-attribute/data-group.

Additionally, we also extract features indicating different types of clausal dependencies between noun phrases and other words of the requirements sentences. Some of these features are included to our selection pool of features (e.g. F6, F7, F15, F17, F19, F21, F23 and F25, as presented in Sections 7.4.2 and 7.4.3), so that our supervised learning-based data-movement classification approach can determine the discriminating complex rules involving these complex features.

## Appendix D

# An Example of Automated FSM

In this appendix, we present some screenshots of the GATE environment (Cunningham H., *et al.*, 2011) executing our pipelines that implement our approaches, as presented in Chapter 7, to automate the extraction of the artifacts of COSMIC FSM model, and quantify the functional size using the formulas, presented in Section 7.8. The details of the execution is shown in the following steps.

**Step 1:** We first load the textual requirements belonging to a functional process as GATE document, as shown in Figure 55.

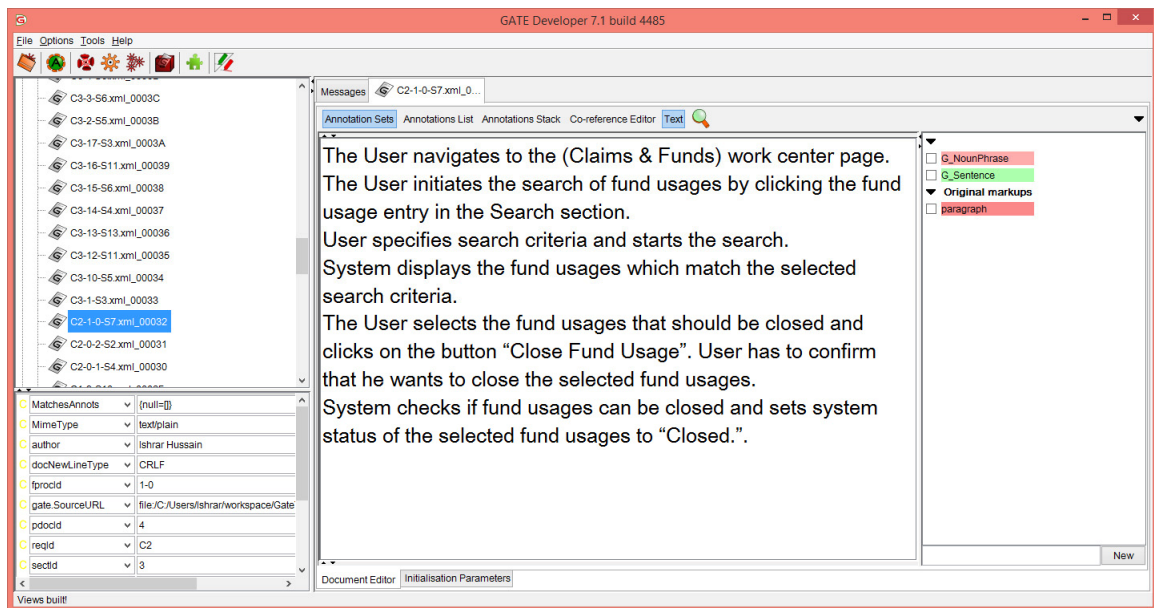


Figure 55: Screenshot (1) of GATE, Loaded with the Textual Requirements of a Functional Process

**Step 2:** We then create a corpus (named “TestCorpus”, as shown in the figure) and add the newly loaded GATE document to the corpus, as shown in Figure 56.

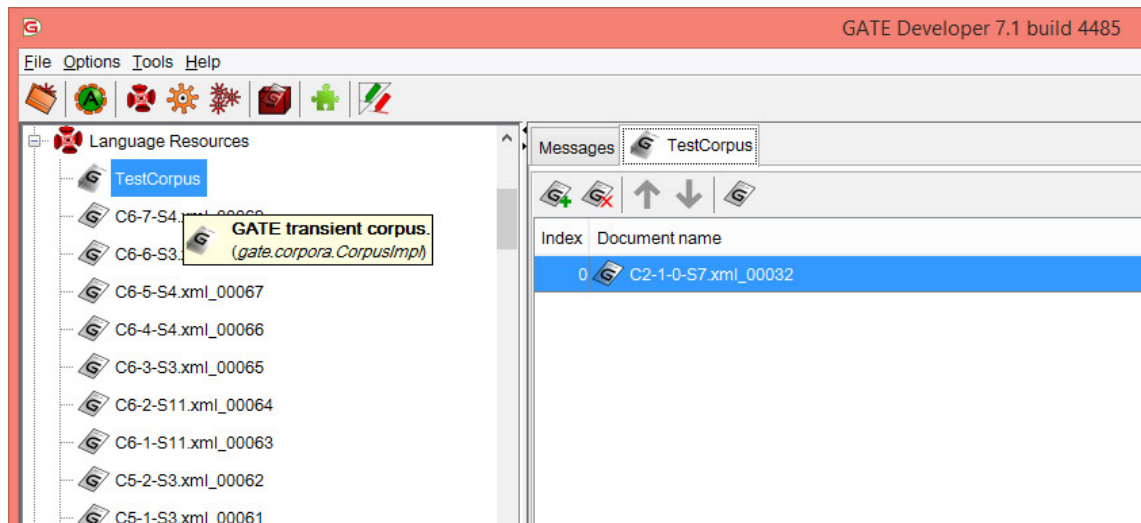


Figure 56: Screenshot (2) of GATE, Creating a Corpus

We can add more documents to execute the pipelines on a batch of multiple functional processes.

**Step 3:** We then run our pipeline called “TestDGExtraction” over the corpus, that pre-processes and extracts all feature values from the document(s) of the corpus. We run the pipeline as shown in Figure 57.



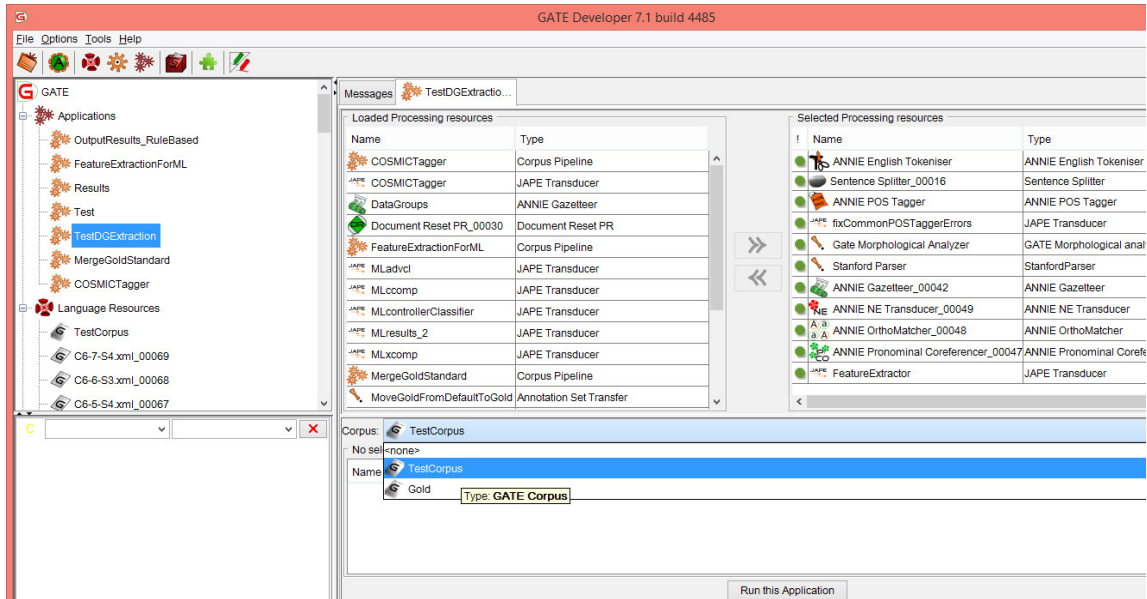


Figure 57: Screenshot (3) of GATE, Loading a Pipeline for Preprocessing and Feature Extraction

This GATE pipeline processes each document in the corpus and extracts all the Sentences, the Noun Phrases and the values of all their features, as presented in Section 7.4.

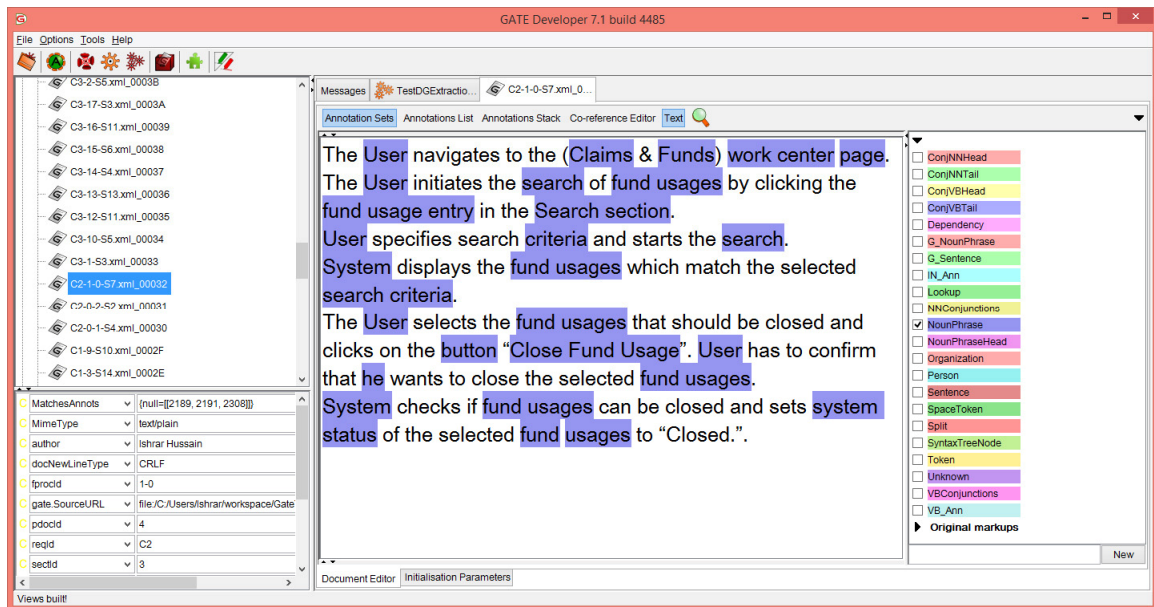


Figure 58: Screenshot (4) of GATE, Showing the Output of Running the Preprocessing & Feature Extraction Pipeline

Thus, the extracted feature values of any Noun Phrase can be viewed as shown in Figure 59.

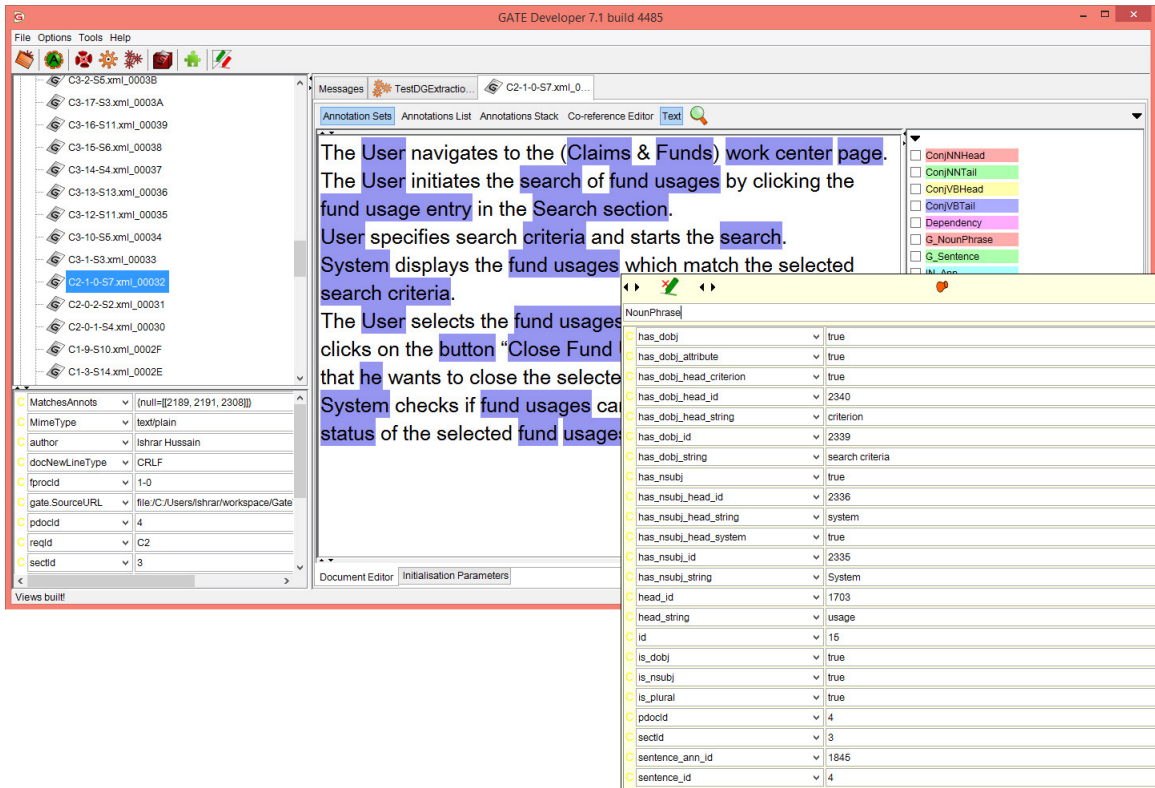


Figure 59: Screenshot (5) of GATE, Showing Feature Values Extracted from a Noun-Phrase

**Step 4:** We now run our pipeline called “COSMICTagger” over the corpus, that implements our heuristic-based approaches of Data-Movement classification, Data-Group identification and CFP quantification, as presented in Sections 7.5, 7.7 and 7.8. The interface of running this pipeline is as shown in Figure 60.

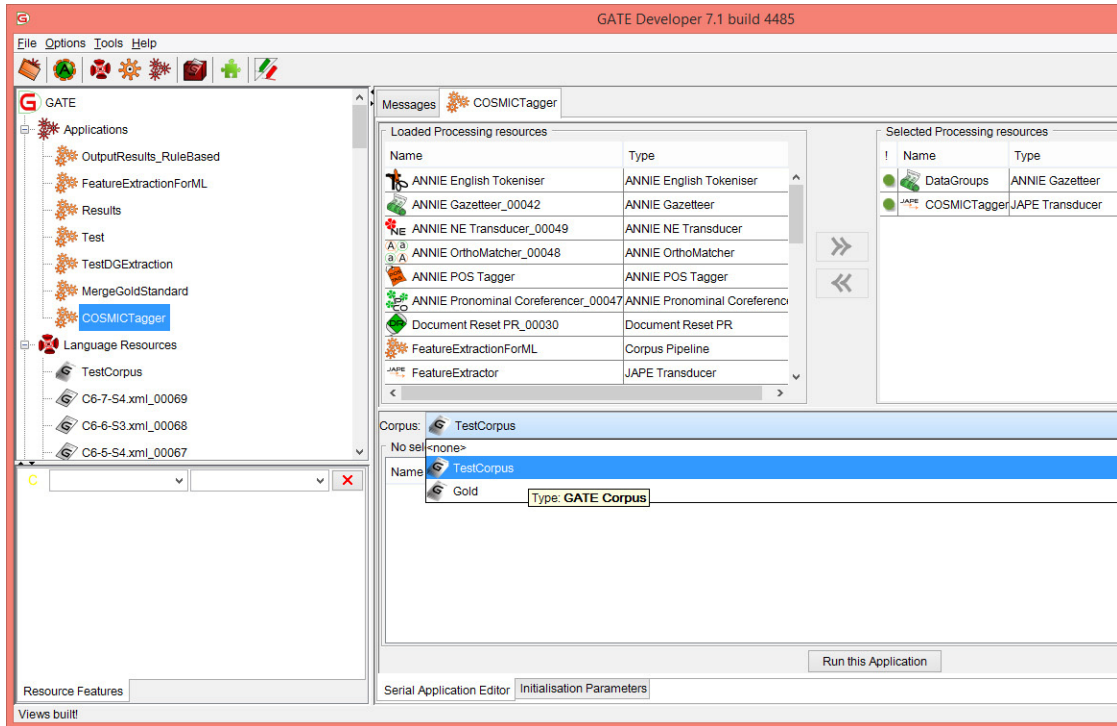


Figure 60: Screenshot (6) of GATE, Loading a Pipeline for COSMIC FSM Model Extraction & Quantification

This GATE pipeline again processes each document in the corpus and extracts the artifacts of the COSMIC FSM standard, e.g. the Data-Groups and their types of Data-Movements, as shown in Figure 61 and Figure 62.

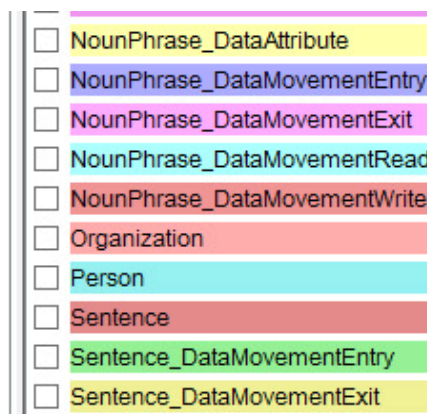


Figure 61: Screenshot (7) of GATE, Showing the Output Annotations of Noun-Phrases & Sentences Classified into Data-Movement Types

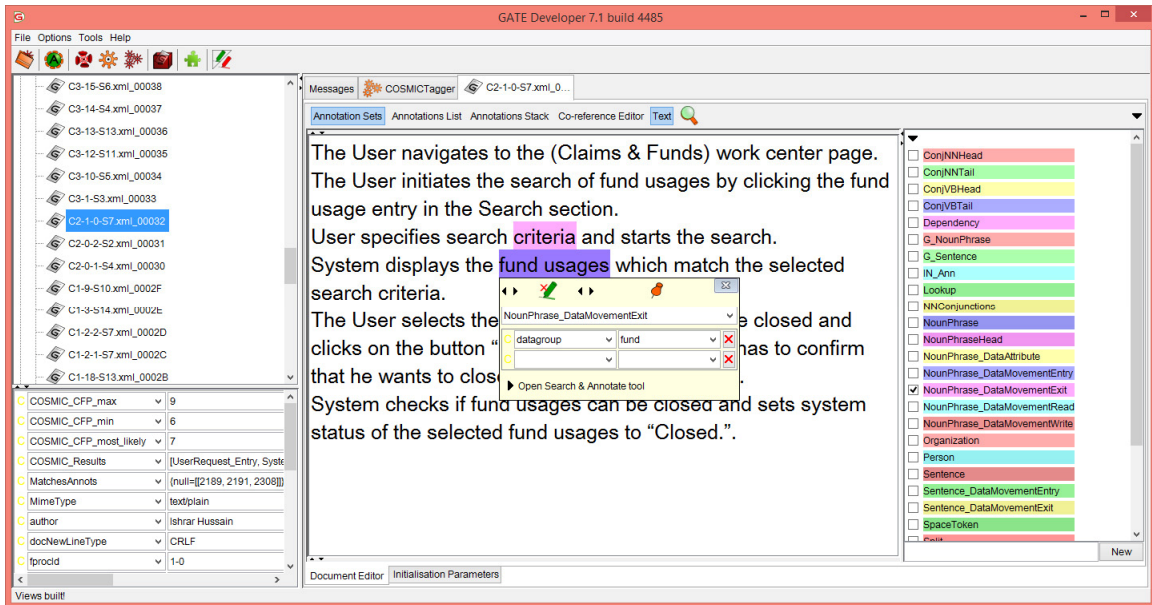


Figure 62: Screenshot (8) of GATE, Showing the Identified Data-Group That is Associated with an Output Annotation of Noun-Phrase

And it also outputs the quantification of functional size for each functional process by showing the minimum, most-likely and maximum CFP and lists the Data-Groups and their corresponding types of Data-Movements — all as features to the corresponding GATE document, as shown in Figure 63.

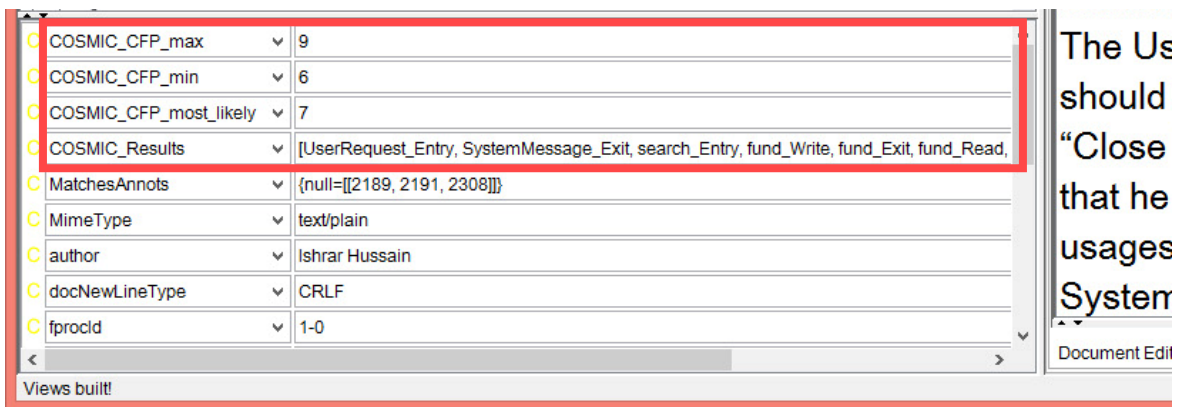


Figure 63: Screenshot (9) of GATE, Showing the Output CFP and the List of Extracted Artifacts of COSMIC FSM Standard