University of Novi Sad

Faculty of Sciences

Department of Mathematics and Informatics

Miloš Savić

Extraction and analysis of complex networks from different domains

– Doctoral dissertation –

Advisor:
Mirjana Ivanović, Ph.D.

Novi Sad, 2015

"Čovek se štimuje celoga života
u potrazi za harmonijom svog akorda
koji ga predstavlja celom svetu."

- Zoran Kostić Cane

Preface

Almost any large-scale system can be viewed as a network that shows interactions among entities which are constituent parts of the system. Understanding, controlling and improving complex systems essentially implies that we are able to quantify, characterize and comprehend the structure and evolution of underlying network representations. Complex networked systems are all around us: we can find complex networks in social systems (e.g., online social networks, collaboration networks), biological systems (e.g., networks of protein interactions, neural networks, metabolic pathways networks), technological systems (e.g., power-grids, communication networks, WWW), and conceptual systems (conceptual maps, linguistic networks). The focus of this dissertation is on complex networks from three domains: (1) networks extracted from source code of computer programs that represent the design of software systems, (2) networks extracted from semantic web ontologies that describe the structure of shared and reusable knowledge, and (3) networks extracted from bibliographic records that depict collaboration in science.

The dissertation consists of the following six chapters:

1. Introduction
2. Theoretical background
3. Software networks
4. Ontology networks
5. Co-authorship networks
6. Conclusions

In the first chapter we provide a brief discussion about networks that are the subject of the dissertation. The next chapter describes theoretical background of the dissertation presenting metrics and models developed under the framework of complex networks theory. Techniques for extraction and analysis of software, ontology and co-authorship networks, overview of existing related research works, as well as concrete case studies are presented in the next three chapters. Each of those chapters has its own summary which also indicates possibilities for future work. Chapter six concludes the dissertation and gives an overview of its contributions.

At this moment, I would like to thank my supervisor professor Mirjana Ivanović for her advice and unreserved support. This dissertation would not have been possible without her and encouragement she has given me over the last years. My special thanks are also addressed to other professors and colleagues with whom I collaborated on the research related to this dissertation: professor Zoran Budimac, assistant professor Miloš Radovanović, Gordana Rakić, professor Zoran Ognjanović, Aleksandar Pejović, Tatjana Jakšić Krüger, and professor Marjan Heričko. I would also like to thank Toki, Savan, Gorionik, Rilke, Jovičići, Bolex and the rest of the company from the Petnica Science Center where I made the first steps in computer science. In Petnica I also had met professor Mašulović who inspired me to enroll the studies of informatics at the Faculty of Sciences in Novi Sad. My thanks also go to my colleagues from the Chair of Computer Science for nice and pleasant working environment,

especially to Djurica & Vlada (for smoking breaks), Dejan (for being almost regular roommate at workshops and conferences we attended together), and Davorka & Doni (for taking over my classes during my visits to Maribor). I would also like to thank CEEPUS for the scholarships which enabled me to work on ontology networks and metrics in Maribor, as well as professor Marijan Heričko and other colleagues from the Faculty of Electrical Engineering and Computer Science in Maribor for their hospitality. Especially, my thanks are addressed to Rok Žontar for fruitful discussions on ontology metrics. Finally, I thank my parents and brother for their endless support and patience during my studies.

Contents

Preface ii

List of Figures vii

List of Tables ix

Abbreviations xiii

1 Introduction 1
- 1.1 Software networks . . . 2
- 1.2 Ontology networks . . . 3
- 1.3 Co-authorship networks . . . 4
- 1.4 Contributions . . . 4

2 Theoretical background 6
- 2.1 Basic definitions . . . 6
- 2.2 Basic metrics on graphs . . . 8
  - 2.2.1 Connectivity metrics . . . 9
  - 2.2.2 Metrics of small-worldliness . . . 11
  - 2.2.3 Centrality metrics . . . 11
  - 2.2.4 Link reciprocity . . . 13
  - 2.2.5 Metrics of clustering . . . 14
- 2.3 Basic models of complex networks . . . 17

3 Software networks 21
- 3.1 Taxonomy of software networks . . . 21
  - 3.1.1 General Dependency Network . . . 23
- 3.2 Software networks and software design metrics . . . 24
  - 3.2.1 Coupling metrics . . . 25
  - 3.2.2 Cohesion metrics . . . 26
  - 3.2.3 Hierarchy trees and compositional software metrics . . . 27
- 3.3 Graph clustering evaluation metrics as software metrics . . . 27
  - 3.3.1 GCE metrics and software networks . . . 27
  - 3.3.2 Theoretical analysis . . . 30
- 3.4 Extraction of software networks . . . 33
  - 3.4.1 Extraction of software networks for statistical analysis . . . 35
  - 3.4.2 Software networks extraction in reverse engineering tools and environments . . . 36

3.4.3 SNEIPL - a novel language-independent approach to the extraction of software networks . . . 37
3.4.3.1 eCST representation of source code . . . 38
3.4.3.2 eCST universal nodes used by SNEIPL . . . 39
3.4.3.3 SNEIPL architecture . . . 41
3.4.3.4 Phase 1 of GDN extraction . . . 43
3.4.3.5 Phase 2 of GDN extraction . . . 44
3.4.3.6 Applicability of SNEIPL – controlled experiment . . . 49
3.4.3.7 Applicability of SNEIPL – extraction of software networks from real-world software systems . . . 50
3.4.3.8 Comparative analysis . . . 54
3.5 Analysis of software networks . . . 58
3.5.1 Related work . . . 59
3.5.2 Experimental dataset . . . 62
3.5.3 Methodological framework . . . 63
3.5.4 Connected component analysis . . . 66
3.5.5 Degree distribution analysis . . . 72
3.5.6 Characteristics of highly coupled classes . . . 77
3.6 Summary and future work . . . 82

4 Ontology networks **85**
4.1 Preliminaries and definitions . . . 85
4.2 Ontology metrics . . . 90
4.3 Graph clustering evaluation metrics as ontology metrics . . . 92
4.4 Extraction of ontology networks . . . 93
4.4.1 Integration of OWL2 into SSQSA . . . 95
4.4.1.1 Benefits of the eCST representation of an ontology . . . 98
4.4.1.2 New metrics to evaluate ontologies . . . 99
4.4.2 ONGRAM tool . . . 100
4.5 Analysis of ontology networks . . . 103
4.5.1 Related work . . . 104
4.5.2 Case study . . . 105
4.5.2.1 Connected component analysis . . . 106
4.5.2.2 Degree distribution analysis . . . 109
4.5.2.3 Characteristics of hubs . . . 111
4.5.2.4 Cohesion of ontology modules . . . 113
4.5.2.5 Correlations between ontology metrics . . . 115
4.5.2.6 Final remark on SWEET modularization quality . . . 117
4.6 Summary and future work . . . 117

5 Co-authorship networks **119**
5.1 Formal definition of co-authorship networks . . . 119
5.2 Extraction of co-authorship networks . . . 120
5.2.1 Initial-based approaches to name disambiguation . . . 121
5.2.2 Heuristic approaches to name disambiguation . . . 123
5.2.3 Machine learning approaches to name disambiguation . . . 124
5.2.4 Author identification in massive bibliography databases . . . 125
5.3 Analysis of co-authorship networks . . . 126

5.3.1 Co-authorship networks of mathematicians . . . 131
5.4 Case study: ELib co-authorship network . . . 133
5.4.1 Extraction of the eLib co-authorship network . . . 134
5.4.1.1 Preliminary analysis of data . . . 134
5.4.1.2 Extraction procedure . . . 135
5.4.1.3 Analysis of author names . . . 137
5.4.2 Analysis of the eLib co-authorship network . . . 141
5.4.2.1 Publication dynamics . . . 144
5.4.2.2 Author dynamics . . . 145
5.4.2.3 Basic characteristics of collaboration and productivity of eLib authors 146
5.4.2.4 The structure of the eLib co-authorship network . . . 148
5.4.2.5 Communities in the eLib co-authorship network . . . 155
5.4.2.6 The evolution of the eLib co-authorship network . . . 158
5.5 Summary . . . 163

6 Conclusions and future work **166**

A Degree distributions of software networks **168**

Bibliography **176**

Sažetak **197**

Prošireni izvod **199**

Kratka biografija kandidata **211**

Ključna dokumentacijska informacija **212**

Key Words Documentation **214**

List of Figures

3.1 General Dependency Network for a software system consisting of two classes. 24
3.2 Class collaboration network of a simple software system and appropriate cluster quality measures. 30
3.3 Concrete syntax tree (a), abstract syntax tree (b), and enriched concrete syntax tree (c) representing Java fragment "class A extends B { }". 39
3.4 Two code fragments in Modula-2 and Java with the same structure of eCST universal nodes in the eCST representation. 41
3.5 Data flow in software networks extraction process. 42
3.6 Two phases in GDN extraction: Phase 1 forms hierarchy tree while Phase 2 creates horizontal dependencies. 47
3.7 Extracted static call graphs and class collaboration networks for two identical program written in Java and C#. 50
3.8 Densification of strongly connected components in (A) Tomcat, (B) Lucene, (C) Ant, (D) Xerces, and (E) JFreeChart. 70
3.9 Complementary cumulative out-degree distribution for (A) Lucene and (B) Ant. . . . 76
3.10 In-Out degree disbalance for (A) Tomcat, (B) Lucene, (C) Ant, (D) Xerces, and (E) JFreeChart. 78

4.1 Simple ontology O with its RDF and ontology graph representations. 89
4.2 Normalization of complex class expressions. 89
4.3 Ontology graph of a simple modularized ontology. 93
4.4 The eCST representation of a simple ontology. 97
4.5 ONGRAM architecture. 101
4.6 In-out degree disbalance for the SWEET ontology networks: (A) ontology module network, (B) ontology class network and (D) ontology subsumption network. 113

5.1 Data flow in the extraction of the eLib co-authorship network. 136
5.2 The number of papers published in the eLib journals per year. Above the line are shown the names (or abbreviations) of journals in the time they were founded, while important events in Yugoslav/Serbian history are positioned below the line. 145
5.3 The number of eLib authors (a) and the number of male and female authors (b) per year. 146
5.4 The fraction of returning authors per year. 146
5.5 The evolution of the average number of authors per paper (a), and the fraction of single authored papers (b). 147
5.6 Complementary cumulative distribution of the number of papers per author (a), and the distribution of the number of authors per paper (b). 148
5.7 Fraction of papers written by the most prolific eLib authors. 149
5.8 The distribution of the size of components (a), and the number of papers per component (b) in the eLib co-authorship network. 151

5.9 The distribution of link weights (a) and link timespans (b) in the eLib co-authorship network. . . . 154
5.10 Visualization of the largest connected component in the eLib co-authorship graph. Nodes from the same community are in the same color. Additionally, each community is marked with an appropriate identifier (C1, C2, etc.) used in Table 5.14. . . . 156
5.11 Visualization of the second largest connected component in the eLib co-authorship graph after community detection. . . . 157
5.12 Visualization of the third largest connected component in the eLib co-authorship graph after community detection. . . . 158
5.13 The evolution of the ratio between the number of links and non-isolated nodes (LNR), and the fraction of isolated nodes (ISOL) in the eLib co-authorship network. . . . 159
5.14 The evolution of the average component size (a), and clustering coefficient (b) for non-trivial components in the eLib co-authorship graph. . . . 161
5.15 The evolution of Spearman's correlations between co-authorship network based author metrics (degree and betweenness centrality) and metrics of productivity (the number of publications and author timespan) for eLib authors. . . . 162
5.16 The evolution of average link weight for non-trivial connected components in the eLib co-authorship graph. . . . 164

A.1 Complementary cumulative degree distribution of the Tomcat class collaboration network. 168
A.2 Complementary cumulative in-degree distribution of the Tomcat class collaboration network. . . . 169
A.3 Complementary cumulative out-degree distribution of the Tomcat class collaboration network. . . . 169
A.4 Complementary cumulative degree distribution of the Lucene class collaboration network. 170
A.5 Complementary cumulative in-degree distribution of the Lucene class collaboration network. . . . 170
A.6 Complementary cumulative out-degree distribution of the Lucene class collaboration network. . . . 171
A.7 Complementary cumulative degree distribution of the Ant class collaboration network. 171
A.8 Complementary cumulative in-degree distribution of the Ant class collaboration network. 172
A.9 Complementary cumulative out-degree distribution of the Ant class collaboration network. 172
A.10 Complementary cumulative degree distribution of the Xerces class collaboration network. 173
A.11 Complementary cumulative in-degree distribution of the Xerces class collaboration network. . . . 173
A.12 Complementary cumulative out-degree distribution of the Xerces class collaboration network. . . . 174
A.13 Complementary cumulative degree distribution of the JFreeChart class collaboration network. . . . 174
A.14 Complementary cumulative in-degree distribution of the JFreeChart class collaboration network. . . . 175
A.15 Complementary cumulative out-degree distribution of the JFreeChart class collaboration network. . . . 175

List of Tables

3.1 Properties of graph clustering metrics as (lack of) cohesion software metrics. 33
3.2 List of eCST universal nodes used to extract software networks. 40
3.3 Software networks extracted by SNEIPL and the parameterization of "select-connected by" queries. 43
3.4 The summary of software systems used in the extraction experiment. 51
3.5 The number and distribution of nodes and links in extracted General Dependency Networks. 51
3.6 Characteristics of extracted hierarchy networks: #nodes - the number of nodes, #links - the number of links, IN_0 - the number of nodes without in-coming links, OUT_0 - the number of nodes without out-going links, UPP - the average number of units per package, FPU - the average number of functions per unit, and VPU - the average number of global variables per unit. 52
3.7 Characteristics of extracted package collaboration networks: #nodes - the number of nodes, #links - the number of links, #isol - the number of isolated nodes, MaxAC - the highest value of in-degree (afferent coupling), MaxEC - the highest value of out-degree (efferent coupling). 52
3.8 Characteristics of extracted class/module collaboration networks: #nodes - the number of nodes, #links - the number of links, #isol - the fraction of isolated nodes, MaxIn - class/module having the highest in-degree, MaxOut - class/module having the highest out-degree (the exact values of in- and out- degrees are given in brackets). 53
3.9 Characteristics of extracted static call graphs/method collaboration networks. 53
3.10 Functions with the maximal values of in- and out- degree in extracted static call graphs. 53
3.11 Java software systems used in the comparative analysis. 54
3.12 Similarity between class collaboration networks extracted by A = SNEIPL and B = Dependency Finder. 56
3.13 Similarity between class collaboration networks extracted by A = Dependency Finder and B = Doxygen. 56
3.14 Quantification of missing CALLS dependencies in networks extracted by SNEIPL: Calls resolved (%) – the fraction of resolved function calls, HTM – the fraction of hard to match functions in the source code, HTM resolved – the number of resolved calls to hard to match function, and HTM unresolved – the number of unresolved calls to hard to match functions. 57
3.15 Results of two-sample Kolmogorov-Smirnov tests: D – Kolmogorov-Smirnov statistics, p – the value of the significance probability. "Accepted" denotes if the null hypothesis (no statistically significant differences between distributions) is accepted or not. 58
3.16 The distribution of CBO differences ($\triangle CBO$) when they are calculated using CCNs extracted by SNEIPL and Dependency Finder. 58
3.17 Experimental dataset of class collaboration networks. N is the number of nodes, while L is the number of links. 62

3.18 Results of the weakly connected component analysis: $\#WCC$ – the number of WCCs, $N(LWCC)$ – the number of nodes in the largest WCC, $L(LWCC)$ – the number of links in the largest WCC, $N(SWCC)$ – the number of nodes in the second largest WCC, I – the number of isolated nodes. All quantities are given in percentages with respect to the total number of nodes (links). 66
3.19 Characteristics of giant weakly connected components: SW – small-world coefficient, SW_r – the small-world coefficient of comparable random graph, D – diameter, CC – clustering coefficient, CC – clustering coefficient of comparable random graph, A – assortativity index. 67
3.20 Characteristics of strongly connected components: $\#SCC$ – the number of SCCs, $LSCC$ – the size of the largest SCC, $N(SCC)$ – the total number of nodes contained in SCCs , R – reciprocity, R_n – normalized reciprocity, R^p – path reciprocity. 68
3.21 Densification of strongly connected components. $\rho(N(S), \frac{L(S)}{N(S)})$ – Spearman's rank correlation between size and average intra-component degree of SCCs, α – scaling exponent of empirically observed densification law $L(S) \approx N(S)^{\alpha}$. 69
3.22 The results of the metric-based comparison test for strongly connected components. . 71
3.23 The basic characteristics of empirically observed degree distributions. 73
3.24 The results of the power-law test. 74
3.25 The results of the power-law test through the whole range of values ($x_m = 1$). 75
3.26 The fraction of highly coupled classes (H) and the minimal total degree (CBO) of highly coupled classes (H_d). 77
3.27 The top ten most coupled classes in Xerces. 79
3.28 The top ten most coupled classes in Ant. 79
3.29 The results of the metric-based comparison test for hubs. 81

4.1 The results of testing of the OWL2 grammar testing using ontologies from the TONES repository. 96
4.2 Conversion of SWEET ontologies to the eCST representation. 105
4.3 The distribution of nodes and links in the SWEET ontology graph. 106
4.4 The number of nodes and links in the SWEET ontology networks. 106
4.5 Weakly connected components of the SWEET ontology networks. OMN denotes the ontology module network, OCN the ontology class network and OSN ontology subsumption network. #WCC – the number of weakly connected components (WCCs), LWCCN – the fraction of nodes in the largest WCC, LWCCL – the fraction of links in the largest WCC, SW – the small-world coefficient, SW-rnd – the small-world coefficient of a comparable random graph, CC – the clustering coefficient, CC-rnd – the clustering coefficient of a comparable random graph, A – assortativity index. 107
4.6 Strongly connected components of the SWEET ontology networks. #SCC – the number of strongly connected components (SCCs), LSCCN – the fraction of nodes in the largest SCC, LSCCL – the fraction of links in the largest SCC, S – the number of nodes contained in all SCCs, R – link reciprocity, R_n – normalized link reciprocity, R_p – path reciprocity, C – the number of SCCs of trivial complexity. 108
4.7 Characteristics of the largest strongly connected component in the SWEET ontology module network. 109
4.8 The results of the power-law test for degree distributions of the SWEET ontology networks. OMN, OCN and OSN denote ontology module network, ontology class network and ontology subsumption network, respectively. 110
4.9 The comparison of the best Poisson, log-normal and exponential fits to the empirically observed total-degree distributions. 111

4.10 The coefficient of variation (c_v), skewness (G_1) and the average value (μ) of the total-degree sequence for examined ontology networks. The coefficient of variation and skewness of the Poisson fit are equal to $\mu^{-0.5}$. 111
4.11 The fraction of highly coupled nodes (the size of the H set) and the minimal total degree of nodes contained in H for SWEET networks. 111
4.12 The top five highest coupled nodes in the SWEET ontology networks. Total, In and Out denote total-degree, in-degree and out-degree, respectively. 112
4.13 Characteristics of hub modules in the SWEET ontology module network. 114
4.14 Spearman correlations between GCE metrics. 114
4.15 Spearman correlations between GCE metrics and metrics of internal density (DEN) and connectedness (COMP). 115
4.16 Poorly cohesive modules in SWEET. A denotes the average value of metrics considering all SWEET ontology modules. Ontology modules are sorted by conductance. 115
4.17 Spearman correlations between metrics of internal complexity. 116
4.18 Spearman correlations between metrics of design complexity. 116
4.19 Spearman correlations between metrics of internal and design complexity. 117

5.1 Examples of name pairs representing different persons that have high degree of similarity. JT, JN, JW, TIT and TIN denote the value of Jaccard token, Jaccard n-gram, Jaro-Winkler, TF-IDF token, TF-IDF n-gram proximity, respectively. 140
5.2 Proximities of two name pairs where the first pair represent different persons and the second one the same persons. 140
5.3 Excerpt from the name correction lookup. 141
5.4 Examples of corrections identified in the second name analysis. AIC/Path denotes the author in common or path connecting nodes represented by names, while MS is the maximal similarity which is obtained by string similarity metric M. 141
5.5 Statistical properties of the name correction lookup. SD - standard deviation, CV - coefficient of variation, Min - minimal value, Max - Maximal value. 141
5.6 Lookup entries with the smallest proximity score per string similarity measure (indicated by the bold typeface). 141
5.7 Basic structural parameters of the eLib co-authorship network and co-authorship networks restricted to individual journals: #P – the number of papers, #N – the number of nodes (authors), #L – the number of links, I – the fraction of isolated authors, MFR – Male-Female Ratio, #C – the number of connected components (isolated nodes are excluded), LC – the relative size of the largest connected component, SW – small-world coefficient, CC – clustering coefficient, AC – assortativity coefficient 150
5.8 The ten largest connected components in the eLib co-authorship network: #N – the number of nodes (authors), #L – the number of links, #P – the number of papers that authors in the component published, #J – the number of journals where authors from the component published their papers, EP – evolution period of the component, $\langle d \rangle$ – average degree of node in component, SW – small world coefficient, D – diameter, and CC – clustering coefficient. 152
5.9 The top ten highest degree authors in the largest component of the eLib co-authorship network: Deg. – degreee, #P – the number of published papers, #PR – rank of author according to the number of published papers, S – author timespan, SR – rank according to timespan, B – betweenness, BR – rank according to betweenness centrality, and CC – clustering coefficient. 153
5.10 Values of Spearman's correlation coefficient for author metrics. 153
5.11 Values of Spearman's correlation coefficient for link (collaboration) metrics. 154

5.12 Comparative analysis of performance of different community detection methods applied to the largest connected component: C – the number of detected communities, Q – modularity score, Strong – the percentage of Radicchi strong communities. 155
5.13 Results of community detection for ten largest connected components in the eLib co-authorship graph: N – the number of nodes in the component, Q – modularity score, C – the number of detected communities. 155
5.14 Description of detected communities for the largest connected eLib component. 157
5.15 Description of detected communities for the second largest connected eLib component. 158
5.16 Description of detected communities for the third largest connected eLib component. . 158
5.17 The number of collaborations between old authors (Old-Old), old and new authors (Old-New) and new authors (New-New) for the last four characteristic periods in the evolution of the eLib co-authorship network. The most dominant types of collaborations are bold. 160
5.18 The top ranked author according to the number of papers, degree, and betweenness centrality in different periods of eLib evolution. 163

Abbreviations

ER	Erdős-Renyi
BA	Barabási-Albert
KS	Kolmogorov-Smirnov
MWU	Mann-Whitney U
WCC	Weakly connected component
GWCC	Giant weakly connected component
SCC	Strongly connected component
GSCC	Giant strongly connected component
CCD	Complementary cumulative distribution
SNA	Social network analysis
OO	Object-oriented
PCN	Package collaboration network
CCN	Class collaboration network
SCG	Static call graph
FUGV	Function uses global variable
GDN	General dependency network
LOC	Lines of code
CK	Chidamber-Kemerer
CBO	Coupling between objects
DIT	Depth of inheritance tree
NOC	Number of children
LCOM	Lack of cohesion in methods
TCC	Tight class cohesion
LCC	Loose class cohesion
RFC	Response for a class
CHA	Class hierarchy analysis

RTA	Rapid type analysis
eCST	Enriched concrete syntax tree
SSQSA	Set of software quality static analyzers
AST	Abstract syntax tree
CST	Concrete syntax tree
ASTM	Abstract syntax tree metamodel
KDM	Knowledge discovery metamodel
GASTM	Generic abstract syntax tree metamodel
SASTM	Specific abstract syntax tree metamodel
SNEIPL	Software networks extractor independent on programming language
IN	In-degree
OUT	Out-degree
TOT	Total-degree
PR	Page rank
BET	Betweenness centrality
GCE	Graph clustering evaluation
ODF	Out-degree fraction
EBNF	Extended Backus-Naur form
OWL	Web Ontology Language
W3C	World Wide Web Consortium
RDF	Resource Description Framework
IRI	Internationalized Resource Identifiers
OMN	Ontology Module Network
OCN	Ontology Class Network
OSN	Ontology Subsumption Network
OON	Ontology Object Network
NEC	Number of external classes
REC	References to external classes
RI	Referenced includes
ONGRAM	Ontology graphs and metrics
EXPC	Expression complexity
TEXPR	Total expression complexity
AEXPR	Average expression complexity
AXM	The number of axioms

HVOL	Halstead volume
HDIF	Halstead difficulty
NCLASS	Number of classes
NINST	Number of instances
HK	Henry-Kafura complexity
AP	Average population
CR	Class richness
RR	Relationship richness
CON	Conductance
EXP	Expansion (when denoting metric), exponential (when denoting distribution)
CUTR	Cut-ratio
AODF	Average out-degree fraction
MODF	Maximal out-degree fraction
FODF	Flake out-degree fraction
SWEET	Semantic Web for Earth and Environmental Terminology
eLib	The Electronic Library of the Mathematical Institute of the Serbian Academy of Sciences and Arts
PIM	Publications de l'Institut Mathématique (journal)
MV	Matematički Vesnik (journal)
ZR	Zbornik Radova (journal)
PDA	Publications of Department of Astronomy (journal)
NM	Nastava Matematike (journal)
TTM	The Teaching of Mathematics (journal)
VM	Visual Mathematics (journal)
KJM	Kragujevac Journal of Mathematics (journal)
Bulletin	Bulletin, Classe des Sciences Mathématiques et Naturelles, Sciences mathématiques (journal)
RNCD	Review of the National Center for Digitization (journal)
ComSIS	Computer Science and Information Systems (journal)

Chapter 1

Introduction

In the past decade, a huge and growing body of research has investigated statistical properties of complex, real-world networks. Watts and Strogatz [Watts and Strogatz, 1998b] discovered that three large and sparse real-world networks exhibit the small-world effect (small distance between two randomly chosen nodes) and a high level of clustering (highly dense sub-graphs induced by a randomly chosen node and adjacent nodes). The discovery was significant because the classical theory of random graphs, used until then in modeling complex networked structures, cannot explain the presence of these two qualities together in one large and sparse graph. Analysis of statistical properties of graphs that represent large portions of the World Wide Web [Albert et al., 1999; Kumar et al., 1999] and the Internet at the physical level [Faloutsos et al., 1999] led to the discovery that their degree distributions (probability $P(k)$ that a randomly chosen node has exactly k links) follow power-laws of the form $P(k) \sim Ck^{-\gamma}$, a property that the Erdős-Renyi model of random graphs [Bollobás, 2001; Erdős and Rényi, 1959, 1960] does not predict. Networks obeying the previously mentioned connectivity pattern are also known as *scale-free networks* [Barabasi and Albert, 1999]. A majority of nodes in scale-free networks are loosely connected, but they also contain a small, but significant, fraction of nodes (called hubs or preferential nodes) whose degree of connectedness is unexpectedly high and tends to increase as networks evolve [Barabasi and Albert, 1999]. Two important consequences of the scale-free network organization are (1) the "robust, yet fragile" property [Albert et al., 2000; Bollobás and Riordan, 2003] (robustness to random failures, but extreme disintegration of the network when failures occur in hubs) and (2) the absence of a propagation threshold in spreading processes [Pastor-Satorras and Vespignani, 2001]. Newman's studies of complex networks from different domains revealed another two important characteristics of real-world networks: assortativity mixing patterns (tendency that hubs either establish or avoid connections among themselves) [Newman, 2002, 2003a] and community organization (existence of highly dense sub-graphs in a sparse graph) [Girvan and Newman, 2002; Newman and Girvan, 2004]. After that, researchers analyzed a variety of complex biological, social, technological and conceptual systems represented as networks, looking for presence of the newly discovered phenomena (good overviews can be found in [Albert and Barabási, 2002; Boccaletti et al., 2006; Costa et al., 2011; Newman, 2003b]). These studies initiated a new theory of complex networks (also known as network science) whose focus is on analysis techniques and mathematical models which can reveal, reproduce and explain frequently observed topological characteristics of real-world networks.

The research that will be presented in this dissertation is focused on three different types of real-world networks. Namely, we will study methods for the extraction and analysis of networks representing two different kinds of engineered systems, software and ontology systems, and networks

depicting self-organized systems of research collaboration.

1.1 Software networks

Modern software systems consist of many hundreds or even thousands of interacting entities at different levels of abstraction. For example, complex software systems written in Java consist of packages, packages group related classes and interfaces, while classes and interfaces declare or define related methods and class attributes. Interactions, dependencies, relationships, or collaborations between software entities form various types of *software networks* that provide different granularity views of corresponding software systems. In the literature software networks are also known as software collaboration graphs [Myers, 2003], software architecture maps [Valverde et al., 2002], and software architecture graphs [Jenkins and Kirk, 2007]. Depending on the level of abstraction specific software networks, such as package, class and method collaboration networks [Hylland-Wood et al., 2006], can be distinguished. Additionally, different coupling types between entities of the same type determine different software networks [Wheeldon and Counsell, 2003]. Due to the terminological and type diversity we use a generic term "software network" to refer to any architectural (entity-level) graph representation of real-world software systems, and to distinguish them from networks representing other complex natural, social, conceptual or man-made systems.

The importance of software networks extraction spans multiple fields such as empirical analysis of complexity of software systems, their reverse engineering and computation of software metrics [Savić et al., 2014]. Links in software networks denote various relationships between software entities such as coupling, inheritance, and invocation. This means that software networks can be used to compute software metrics related to the quality of software design. The primary goal of a reverse engineering activity is to identify system's components and relationships among them in order to create the representation of the system at a higher level of abstraction [Chikofsky and Cross II, 1990]. A typical reverse engineering activity starts with the extraction of fact bases [Beszédes et al., 2005; Kienle and Müller, 2010; Shtern and Tzerpos, 2012]. Source code is the most popular, valuable, and trusted source of information for fact extraction because other artifacts (documentation, release notes, information collected from version management or bug tracking systems, etc.) may be missing, outdated, or unsynchronized with the actual implementation. Fact extraction is an automatic process during which the source code is analyzed to identify software entities and their mutual relationships. In other words, software networks can be viewed as fact bases used in reverse engineering, architecture recovery, and software comprehension activities. They are also used as a part of input for computing reflection models in software reflexion analysis [Murphy et al., 1995]. Architecture recovery techniques usually perform software network partitioning [Chiricota et al., 2003; Mancoridis et al., 1998; Mitchell and Mancoridis, 2006; Scanniello et al., 2010; Wu et al., 2005] or cluster software entities according to feature vectors that can be constructed from software networks [Anquetil et al., 1999; Maqbool and Babri, 2007; Schwanke, 1991]. Graphical representations of software entities and dependencies between them have long been accepted as comprehension aids to support reverse engineering processes [Lanza and Ducasse, 2003]. Moreover, the nodes in a software network can be enriched with software metrics information in order to provide visual, polymetric views of analyzed software systems [Lanza and Ducasse, 2003; Risi and Scanniello, 2012]. Additionally, software networks can be exploited to identify and remove "bad smells" from source code [Oliveto et al., 2011], to support static concept location in source code [Scanniello and Marcus, 2011], to support program comprehension during incremental change [Buckner et al., 2005], to identify design patterns in source code [Lucia

et al., 2009], to support software component retrieval [Inoue et al., 2005] and to predict defects in software systems [Bhattacharya et al., 2012; Bird et al., 2009b; Oyetoyan et al., 2013; Tosun et al., 2009; Zimmermann and Nagappan, 2008].

1.2 Ontology networks

The term ontology has a very broad meaning. In information sciences the term is defined as a specification of a conceptualization [Gruber, 1993]. An ontology formally describes concepts and relations present in a domain of discourse and as such models a certain part of reality. In the context of the Semantic Web vision [Berners-Lee et al., 2001; Shadbolt et al., 2006], ontologies are formal specifications of shared and reusable knowledge that can be used to support automated data-driven reasoning, data integration and interoperability of computer programs which process web accessible resources. The traditional World Wide Web can be viewed as a collection of inter-linked documents created by humans for humans. The Semantic Web is an extension of the World Wide Web based on the concept of structured inter-linked data that can be "consumed" by both humans and autonomous software agents [Berners-Lee et al., 2001].

Information always involves multiple inter-related entities positioned in some context. Therefore, if we want to build computer programs that are able to perform tasks involving publicly available data then we have to specify the structure of data and their surrounding context. The sentence "John Doe is the doctoral adviser of Richard Roe" is an example of information that involves two entities positioned in the context of university education. If a person does not have basic knowledge about the structure of academic organizations then he/she is unable to understand that information and possibly act upon it. Even worse situation is for computer programs which do not have any inherent cognitive capabilities. However, if we formally specify the structure of academic organizations by stating relevant concepts (such as Professor and PhdStudent), relations (such as AdvisorOf), and their mutual associations in the form of "subject-predicate-object" (Professor-AdvisorOf-PhdStudent) then a software agent will be able to interpret the symbols of the triplet "John Doe-AdvisorOf-Richard Roe", i.e. it can infer that John Doe is an university professor and that Richard Roe is a PhD student. A semantic network of concepts, relations and data fragments naturally emerges from a sequence of triplets like the two previous given reflecting knowledge present in a particular domain.

Ontology networks show dependencies among ontological entities present in an ontological description. Ontological descriptions contain axioms that define associations among ontological entities and axioms that specify non-relational properties of ontological entities. For example, the transitivity of a relation is one of non-relational properties that can be used to infer new, not explicitly stated, associations among ontological entities. Since ontology networks are backbones of ontologies they are naturally used to evaluate quality and complexity of ontological descriptions [Zhang, 2008; Zhang et al., 2010]. Similarly to software networks, ontology networks can be exploited in a variety of reverse engineering and comprehension activities such as automated modularization ontologies [Coskun et al., 2011; Stuckenschmidt and Schlicht, 2009], ontology summarization [Zhang et al., 2007] and visualization [Katifori et al., 2007].

1.3 Co-authorship networks

Collaboration among researchers is one of the key factors of scientific progress. One of the most productive mathematicians of all time Paul Erdős has written over 1500 papers with over 500 other researchers [Grossman, 2013]. This enormously high productivity inspired the concept of the Erdős number [Goffman, 1969], which is defined to be one for his co-authors, two for co-authors of his co-authors, and so on. In other words, the Erdős number for a scientist is the length of the shortest path connecting him/her to Erdős in the appropriate *co-authorship network*. The nodes in a co-authorship network represent researchers – people who published at least one research paper. Two researchers are connected by an undirected link if they authored at least one paper together, with or without other co-authors. Additionally, link weights can be introduced in order to express the strength of collaboration.

Co-authorship networks can be viewed as ordinary social networks restricted to people doing science: links in a co-authorship network denote temporal and collegial relationships, and imply a strong academic bond. It has long been realized that the analysis of co-authorship networks can help us to understand the structure and evolution of corresponding academic societies [Newman, 2001b; Savić et al., 2014]. Moreover, those networks can be used to develop models for ranking and determining most influential authors in digital libraries [Gollapalli et al., 2011; Mimno and McCallum, 2007], to automatically determine the most appropriate reviewers for a manuscript [Rodriguez and Bollen, 2008; Rodriguez et al., 2006], or even to predict future research collaborations [Guns and Rousseau, 2014; Liben-Nowell and Kleinberg, 2003; Yan and Guns, 2014].

Co-authorship networks are not the only type of complex networks relevant to empirical analysis of scientific practice. There are two other important types of scientometrics networks: citation and affiliation networks. Citation networks show citations among scientific papers and thus represent the structure of scientific knowledge. Affiliation networks are bipartite graphs that capture the affiliation of researchers to institutions. Affiliation networks can be combined with co-authorship networks in order to study scientific collaboration at the institutional and country levels. On the other hand, a study that combines analysis of co-authorship and citation networks can reveal correlations and intersections between authorship and citation [Martin et al., 2013], and the influence of collaboration on citation practices [Wallace et al., 2012].

1.4 Contributions

The contributions of the dissertation can be categorized as follows:

- Design and implementation of an extensible, language-independent approach to the extraction software networks [Savić et al., 2012, 2014]. The approach is based on the eCST representation [Rakić and Budimac, 2011a,b] and realized as a back-end of the SSQSA framework [Budimac et al., 2012; Kolek et al., 2013; Rakić et al., 2013].

- Design and implementation of an extensible, language-independent approach to the extraction of ontology networks whose nodes are attributed with a rich, hybrid set of metrics. Similarly to the first contribution, the approach is based on the eCST reprezentation and realized as a back-end of the SSQSA framework after SSQSA was extended to support the OWL2 language [Savić et al., 2013].

- Introduction of new ontology metrics and adaptation of existing software metrics of internal complexity for ontology evaluation. Introduction of graph clustering evaluation metrics as software [Savić and Ivanović, 2014] and ontology metrics.

- Introduction of a statistical procedure that compares two groups of nodes in a network, where each node is characterized by a metric vector, and groups are formed according to some topological criterion.

- Design and implementation of semi-automated approach to the extraction of co-authorship networks that is suitable for sparse and fragmented networks and based on heuristics for the identification of name synonyms and homonyms [Savić et al., 2014].

- Analysis of an experimental corpus of real-world networks that are extracted using tools developed in the dissertation. The corpus consists of 5 class collaboration networks associated to open source Java software systems, 3 ontology networks representing one modularized ontology at different levels of abstraction, and the network extracted from the electronic library of the Mathematical institute of the Serbian Academy of Sciences and Arts [Savić et al., 2015; Savić et al., 2014]. In comparison to previous related studies, analyses presented in the dissertation are not purely topological, but combine techniques and metrics developed under the framework of complex network theory with metrics from the domains (software and ontology metrics, metrics of productivity and longevity).

Chapter 2

Theoretical background

In this dissertation the term "network" denotes a graph representation of some real-world system. Graph is one of the fundamental and widely studied mathematical abstractions. Each graph consists of nodes (vertices, points) and links (edges, lines) that connect pairs of nodes. Therefore, graphs capture relations or associations in a set of objects.

2.1 Basic definitions

Definition 2.1 (**Graph).** Graph G is a pair (V, E) where V is a set of nodes and E is a set of two-elements subsets of V, $E = \big\{\{a, b\} \mid a, b \in V\big\}$.

Let $e = \{a, b\}$ be a link in G that connects nodes a and b. Then we say that (1) nodes a and b are adjacent or neighbors, (2) nodes a and b are directly connected, (3) e is incident with both a and b, and (4) nodes a and b are end-points of e. A graph may contain parallel links when two nodes are connected by more than one link, as well as loops which are links connecting a node to itself. In such cases E has to be a multiset of unordered pairs of nodes, so those graphs are also called multigraphs.

Definition 2.2 (**Sub-graph, super-graph).** Graph $G' = (V', E')$ is a sub-graph of graph $G = (V, E)$, $G' \subseteq G$, iff $V' \subseteq V$ and $E' \subseteq E$. If E' contains all links that connect nodes from V' in G then we say that G' is an induced sub-graph of G and that G' is induced by V'. Conversely we say that G is super-graph of G'. If V' is a proper subset of V ($V \subset V'$) and E' is a proper subset of E ($V \subset V'$) then G' is a proper sub-graph of G and G is a proper super-graph of G'.

Two nodes a and b are indirectly connected if there is a path from a to b.

Definition 2.3 (**Path, length, distance).** A path from a to b in graph G is a sub-graph $G' = (V', E')$ of G such that:

- $V' = \{a, v_1, v_2, \ldots, v_n, b\}$, where the elements of V' are distinct nodes.
- $E' = \big\{\{a, v_1\}, \{v_1, v_2\}, \ldots, \{v_i, v_{i+1}\}, \ldots, \{v_n, b\}\big\}$.

The number of edges in G' is the *length* of the path G. The *distance* between a and b is the length of the shortest path from a to b.

If there is a path from a to b then we say that b can be reached from a and vice versa. If every node in a graph can be reached from every other node then the graph is *connected*. In other words, in a connected graph every pair of nodes is either directly or indirectly connected. If a graph is not connected then it consists of more than one connected component.

Definition 2.4 (**Union of graphs).** Graph $C = (V_c, E_c)$ is the union of graphs $A = (V_a, E_a)$ and $B = (V_b, E_b)$, $C = A \cup B$, iff $V_c = V_a \cup V_b$ and $E_c = E_a \cup E_b$.

Definition 2.5 (**Intersection of graphs).** Graph $C = (V_c, E_c)$ is the intersection of graphs $A = (V_a, E_a)$ and $B = (V_b, E_b)$, $C = A \cap B$, iff $V_c = V_a \cap V_b$ and $E_c = E_a \cap E_b$.

Definition 2.6 (**Connected components).** Connected components of graph G are sub-graphs $(C_i)_{i=1}^{k}$ of G such that

- C_i is connected for each i.
- $\bigcup C_i = G$.
- $C_x \cap C_y = \emptyset$ for each distinct x and y.

Please note that there is no path connecting two nodes that belong to different connected components and consequently the distance between them is either undefined or treated as an infinite distance. If a graph has a component that encompasses the vast majority of nodes then the component is called a giant connected component. A formal definition of giant connected component can be given assuming that there is some unbounded process governing the evolution of G in time. In practice, we usually consider a component as giant if its size (the number of nodes contained in the component) is drastically larger than the size of the second largest component.

Definition 2.7 (**Giant connected component).** Let $(G_i)_{i=1}^{\infty}$ be an evolutionary sequence of graph G, i.e. G_i is the state of graph G at time i. Then G has a giant connected component C iff

$$\lim_{i \to \infty} \frac{|N(C_i)|}{|N(G_i)|} = c > 0,$$

where $N(A)$ denotes the set of nodes of A, $|N(A)|$ is the cardinality of $N(A)$ (the number of nodes in A) and c is a constant. In other words, the size of C size grows in proportion to the size of G [Newman, 2010].

Definition 2.1 is the definition of so-called *undirected graph* in which links have no orientation. If two nodes a and b are connected in an undirected graph then, metaphorically speaking, something can be transferred from a to b as well as from b to a. When links have directions then we speak about directed graphs.

Definition 2.8 (**Directed graph).** Directed graph G is a pair (V, E) where V is a set of nodes and E is a set of ordered pairs of V, $E = \{(a, b) \mid a, b \in V\}$. For a link $e = (a, b)$, written also as $e = a \to b$, node a is called the source node, while node b is called the destination node. Similarly as for undirected graphs, we say that a and b are end-points of e when it is not important which of them is the source or destination node.

Link $e = a \to b$ in a directed graph (digraph) denotes that a is directly connected to b in the sense that e emanates from a pointing to/referencing b. The converse is not necessarily true. Namely, b is directly connected to a if and only if there is another link $b \to a$. a is indirectly connected to b if there is a path from a to b. A path from a to b exists if b can be reached from a following an alternating sequence of nodes and links $(a, e_0, v_1, e_1, v_2, \ldots, v_k, e_k, b)$ satisfying the following conditions

- $e_0 = a \to v_1$,

- $e_i = v_i \to v_{i+1}$ for each i in $[1, k-1]$,
- $e_k = v_k \to b$.

The distance from a to b is the length of the shortest path from a to b. Please note that the distance from a to b is not necessarily equal to the distance from b to a due to the directed nature of links. Even more, there may not be a path from b to a. There are two types of connected components in directed graphs: weakly connected and strongly connected components. Consequently we have two types of giant connected components.

Definition 2.9 (**Undirected projection of directed graph).** Let G be a directed graph. The undirected projection of G is the undirected graph obtained by ignoring link directions and assembling parallel links connecting two nodes into one link.

Definition 2.10 (**Weakly connected component).** A sub-graph $W = (V_w, E_w)$ of directed graph $G = (V, E)$ is its weakly connected component if V_w forms a connected component in the undirected projection of G. E_w contains all links from E whose end points belong to V_w, i.e. W is induced by V_w.

Definition 2.11 (**Strongly connected component).** A sub-graph $S = (V_s, E_s)$ of directed graph $G = (V, E)$ is its strongly connected component if for each two nodes a and b from V_s there is a path from a to b and a path from b to a. V_s is the maximal subset of V with respect to inclusion which means that any proper super-graph of S is not strongly connected. E_s contains all links from E whose end points belong to V_s, i.e. S is induced by V_s.

Definition 2.12 (**Connected graph).** An undirected graph G is connected if it contains exactly one connected component.

Definition 2.13 (**Strongly connected graph).** A directed graph G is strongly connected if it contains a strongly connected component that encompasses all nodes in G.

Both directed and undirected graphs can be weighted, attributed, or typed. In weighted graphs, a real value that determines the strength of connection between end-points is assigned to each link. This idea can be further generalized to any type of information that can be assigned to nodes and/or links. In such cases we say that a graph is attributed. If a unique identifier is assigned to each node and each link in a graph then the graph is called labeled. If there is some predefined categorization of nodes and links then a graph is typed. Typed graphs enable us to represent heterogeneous associations among heterogeneous entities.

2.2 Basic metrics on graphs

Throughout this section, we assume that $G = (V, E)$ denotes an arbitrary non-weighted undirected graph without loops and parallel links that contains n nodes labeled by numbers from 1 to n, except in cases when it is explicitly mentioned that G has some other properties. Two basic quantities describing G are the number of nodes (n) and the number of links which is denoted by l. l is a number in the range $[0, \max(l)]$ where $\max(l) = n(n-1)/2$ is the maximal number of links that can exist in G. If G is directed then $\max(l) = n(n-1)$. The relation between l and $\max(l)$ can be used to characterize G as a sparse or dense graph.

Definition 2.14 (**Graph density).** The density of G, denoted by $D(G)$, is equal to $l/\max(l)$.

Definition 2.15 (**Sparse graph).** G is sparse if $l \ll \max(l)$.

Definition 2.16 (**Dense graph).** G is dense if $l \approx \max(l)$.

Definition 2.17 (**Complete graph).** G is complete if $D(G) = 1$. In other words, G is complete when each node is directly connected to every other node.

It is often useful to represent G by an adjacency matrix. The adjacency matrix A of G is a $n \times n$ square matrix where A_{ij} has value 1 if nodes i and j are directly connected or 0 otherwise. For undirected graphs we have that A is a symmetric matrix with respect to the main diagonal, $A_{ij} = A_{ji}$. Also, $A_{ii} = 0$ when G does not contain loops. For directed graphs A_{ij} takes value 1 if i is directly connected to j or zero otherwise. In the case that G contains parallel links then A_{ij} is the number of links connecting i and j in the undirected case, or the number of links emanating from i to j in the directed case. When G is weighted then A_{ij} is the weight of the link connecting i and j.

2.2.1 Connectivity metrics

The most basic topological characteristic of a node is its degree.

Definition 2.18 (**Node degree).** The degree of node i in G, denoted by k_i, is the number of links incident with i, i.e. $k_i = \sum_{j=1}^{n} A_{ij}$.

Definition 2.19 (**Isolated node).** Node i is an isolated node if $k_i = 0$.

If G does not contain parallel links then k_i is equal to the number of nodes to which i is directly connected. By the first theorem of graph theory, the average degree of G, denoted by $\langle k \rangle$, is equal to $2l/n$. The density of G can be also expressed in terms of $\langle k \rangle$:

$$D(G) = \frac{2l}{n(n-1)} = \frac{\langle k \rangle}{n-1}.$$

Therefore, G is sparse when $\langle k \rangle \ll n - 1$, while it is dense when $\langle k \rangle \approx n - 1$.

Definition 2.20 (**Regular graph).** G is regular if all of its nodes have the same degree.

The connectivity of nodes in a regular graph can be described by one number – the average degree. For non-regular graphs the connectivity of nodes can be expressed in terms of their degree distributions.

Definition 2.21 (**Degree distribution).** The degree distribution of G is given by the probability mass function $P(k) = P\{D = k\}$, where D is a random variable that represents the degree of a randomly chosen node. In other words, $P(k)$ is the fraction of nodes in G whose degree is equal to k.

Definition 2.22 (**Complementary cumulative degree distribution).** Complementary cumulative degree distribution function $CCD(k)$ is the probability of observing a node with degree greater than or equal to k, that is, $CCD(k) = \sum_{i=k}^{\infty} P(i)$, where $P(i)$ is the degree distribution of G. Equivalently, $CCD(k)$ is the fraction of nodes in G whose degree is greater than or equal to k.

When G is directed then we can distinguish between the in-degree and out-degree of a node. The degree of the node is then the sum of its in-degree and out-degree and it is called total-degree in order to emphasize the directed nature of G. Consequently, in directed graphs we have three degree distributions describing the connectivity of nodes: in-, out- and total-degree distributions.

Definition 2.23 (**Node in-degree).** The in-degree of node i in G, denoted by $k_{in}(i)$, is the number of links pointing to i, i.e. $k_{in}(i) = \sum_{j=1}^{n} A_{ji}$.

Definition 2.24 (**Node out-degree).** The out-degree of node i in G, denoted by $k_{out}(i)$, is the number of links emanating from i, i.e. $k_{out}(i) = \sum_{j=1}^{n} A_{ij}$.

One of important features of social systems is the presence of homophily. Homophily means that an individual tends to establish relationships with other individuals that are similar by one or more attributes such as age, ethnicity, professional vocation, social status, etc. This behavior can be spotted not only in social networks but also in a variety of real-world networks from other domains. Considering degree as the most basic topological characteristic of nodes in a network, we can distinguish between three types of networks:

- Assortative networks are networks in which highly connected nodes, nodes that have a high degree, tend to be connected among themselves.
- Disassortative networks are networks in which highly connected nodes tend to avoid connections to other highly connected nodes.
- Networks that are neither assortative nor disassortative. Such networks are also called non-assortative or uncorrelated networks.

The degree of assortativity of a network can be quantified by the measure known as assortativity index [Newman, 2002, 2003a].

Definition 2.25 (**Assortativity index).** The assortativity index of G, denoted by $a(G)$, is the Pearson correlation coefficient between random variables X and Y, where X and Y are the degrees (total-degrees when G is directed) of end-points of a randomly selected link.

$a(G)$ takes a value in the range $[-1, 1]$. A positive value of $a(G)$ implies that G is assortative, a negative value implies that G is disassortative, while $a(G) = 0$ means that G is uncorrelated. The definition given above can be generalized to any numerical characteristic of nodes or any pair of numerical characteristics.

Definition 2.26 (**Generalized assortativity index).** The generalized assortativity index of G, denoted by $a(G, M_1, M_2)$, is the Pearson correlation coefficient between random variables X and Y, where X is the value of metric M_1 and Y is the value of metric M_2 of end-points of a randomly selected link.

Thus, for directed networks we can measure in-degree assortativity $a(G, k_{in}, k_{in})$, out-degree assortativity $a(G, k_{out}, k_{out})$, in-out-degree assortativity $a(G, k_{in}, k_{out})$, and out-in-degree assortativity $a(G, k_{out}, k_{in})$.

Another way to quantify the degree of assortativity is to compute the slope of the function $k_{nn}(k)$, where $k_{nn}(k)$ denotes the average degree of the nearest neighbors of nodes with degree k [Pastor-Satorras et al., 2001]. If $k_{nn}(k)$ increases with k then G exhibits assortative mixing, while for disassortative networks $k_{nn}(k)$ is a decreasing function of k.

2.2.2 Metrics of small-worldliness

Two nodes in G can be connected via more than one path. However, the shortest paths are the most important considering the efficiency of transportation, communication or information flow. Please recall that the distance between nodes i and j, denoted by d_{ij}, is defined as the length of the shortest path connecting them.

Definition 2.27 (**Small-world coefficient, characteristic path length).** The small-world coefficient or the characteristic path length of G, denoted by $L(G)$, is the average distance between two randomly chosen nodes:

$$L(G) = \frac{1}{n(n-1)} \sum_{i,j \in V, i \neq j} d_{ij}.$$

The problem with the definition given above is that some distances may be undefined (or infinite) if G contains more than one connected component. One possibility to avoid this problem is to focus computation only to the largest connected component (largest strongly connected component when G is directed). This approach is usually taken when G has a giant (strongly) connected component. The second one is to consider only those pairs of nodes that are directly or indirectly connected. The third possibility is to take the harmonic mean of the distances [Boccaletti et al., 2006]. This measure is also called *efficiency* of G and it is computed as:

$$E(G) = \frac{1}{n(n-1)} \sum_{i,j \in V, i \neq j} \frac{1}{d_{ij}},$$

where $\frac{1}{d_{ij}} = 0$ if there is no path connecting i and j.

Definition 2.28 (**Small-world property).** Having the small-world property means that the small-world coefficient of G is a small value (typically $L(G) \approx log(n)$), much smaller than the number of nodes in the network.

Definition 2.29 (**Diameter).** The diameter of G is the length of the longest shortest path:

$$\text{Diam}(G) = \max \sum_{i,j \in V, i \neq j} d_{ij}.$$

2.2.3 Centrality metrics

Centrality measures rank nodes of G with respect to their topological importance in G according to some criteria. Node degree can be viewed as the simplest measure of node importance within a graph. For example, an actor in a social network can be considered important if he/she is involved in a large number of interactions simply because he/she is in position to directly articulate his/her opinions or disseminate his/her interests to a large number of other actors. Other notions of node importance also originate from social network analysis. Here we define basic metrics of node importance: betweenness centrality [Brandes, 2008; Freeman, 1977; Freeman et al., 1991], closeness centrality [Bavelas, 1950; Beauchamp, 1965] and eigenvector centrality [Bonacich, 1972, 1987].

Definition 2.30 (**Betweenness centrality).** The betweenness centrality of node z in G, denoted by $C_b(z)$, is the extent to which z is located on the shortest paths connecting two arbitrary nodes different than z:

$$C_b(z) = \sum_{x,y \in V, x \neq y \neq z} \frac{\sigma(x,y,z)}{\sigma(x,y)},$$

where $\sigma(x, y)$ is the total number of shortest paths connecting x and y, and $\sigma(x, y, z)$ is the total number of shortest paths connecting x and y that pass through z.

$C_b(z)$ can be normalized to the unit interval by dividing by $(n-1)(n-2)/2$ (by $(n-1)(n-2)$ in the case that G is directed). The normalized version of the measure can be interpreted as the probability that a randomly selected shortest path contains z. If a large fraction of shortest paths contain z then z is an important junction point of G, and consequently has a vital role to the overall connectivity of G. In a recent study Boldi et al. [2013] showed that removing nodes in the betweenness centrality order causes a quick fragmentation of real-world networks into a large number of disjoint connected components. If G have a clustered/community organization then nodes with high C_b tend to be located at the intersections of communities which means that they connect together various different parts of G. Nodes with low C_b are typically located at the periphery of G. In social networks, betweenness can be viewed as a measure of the influence an actor has over the spread of information through the network, i.e. actors having a high betweenness centrality are in the position to maintain and control the spread of information. Betweenness centrality can be also defined for links. Betweenness centrality of link e is the number of shortest paths from all nodes to all other nodes that contain link e.

Definition 2.31 (**Closeness centrality).** The closeness centrality of node z in G, denoted by $C_c(z)$, is inversely proportional to the cumulative distance between z to other nodes in G:

$$C_c(z) = \frac{1}{\sum_{i \in V, i \neq z} d_{zi}}$$

The intuition behind the closeness centrality is straightforward: a node can be regarded important if it is in proximity to many other nodes. If we consider a simple spreading process in which information reaching a node is propagated to all of its neighbors then information originating at nodes with high closeness centrality will propagate more efficiently through G. Similarly as for the small-world coefficient, problems with the definition given above appear when G consists of more than one (strongly) connected component. If we restrict the cumulative distance only to reachable nodes then the closeness centrality measure is strongly biased toward nodes that have a small set of reachable nodes [Boldi and Vigna, 2014]. An alternative solution is to consider the normalized harmonic centrality that is defined as:

$$C_h(z) = \frac{1}{n-1} \sum_{i \in R(z)} \frac{1}{d_{zi}},$$

where $R(z)$ denotes the set of nodes that are reachable from z.

Definition 2.32 (**Eigenvector centrality).** The eigenvector centrality of node z in G, denoted by $C_e(z)$, is proportional to the sum of eigenvector centralities of its neighbors:

$$C_e(z) = \frac{1}{\lambda} \sum_{i \in N(z)} C_e(i),$$

where $N(z)$ denotes the set of nodes that are directly connected to z and λ is a constant.

The intuition behind the eigenvector centrality is that a node can be considered important if it is surrounded by other important nodes. Since

$$\sum_{i \in N(z)} C_e(i) = \sum_{i \in V} A_{iz} C_e(i),$$

the recurrence relationship of eigenvector centralities can be given as the eigenvector equation

$$\lambda C_e = AC_e,$$

where C_e is the vector of eigenvector centralities of the nodes. If A is irreducible (which means that G is connected) then the only strictly positive eigenvectors, according to the Perron–Frobenius theorem, are those associated with the dominant eigenvalue.

The set of eigenvalues λ which satisfy the aforementioned equation $\lambda C_e = AC_e$ is also called the spectrum of G. A branch of graph theory known as spectral graph theory studies properties of graphs with respect to eigenvectors and eigenvalues of associated matrices. Important structural information are not only contained in the spectra of the adjacency matrix, but also in the spectra of the Laplacian matrix which is defined as $L = D - A$, where D is the diagonal matrix satisfying $D_{ii} = k_i$. For example, the number of connected components in G is equal to the multiplicity of the lowest eigenvalue of L. The second smallest eigenvalue of L, which is also known as algebraic connectivity, has important properties related to the graph partitioning problem. The article by Cvetković and Simić [2011] gives an overview of the applications of spectral graph theory in various disciplines of computer science including also complex networks. On the other hand, a comprehensive overview of spectral graph theory can be found in the monograph by Cvetković and co-authors [Cvetković et al., 1995].

The page rank is a popular variant of the eigenvector centrality metric for directed graphs. It was primarily designed for ranking nodes in a Web graph crawled by the Google search engine [Brin and Page, 1998b]. Also, this measure has a nice probabilistic interpretation. Let W be a graph walker initially positioned at a randomly selected node of directed graph G. Let a denote the node at which the walker is currently positioned, and let S be the out-neighborhood of a ($S(a) = \{b : (a, b) \in E\}$). The walker behaves according to the following rules:

1. With probability α it jumps to a randomly selected node from S. If $S = \emptyset$ then it stays at a.
2. With probability $1 - \alpha$ it moves to a randomly selected node from G.

The second rule ensures that the walker is never "trapped" at some node when G is not strongly connected. Therefore, α (usually set to 0.85) is also called damping probability or damping factor. Then, the page rank of node i is the probability that the walker ends up at i after k steps when $k \to \infty$.

Definition 2.33 (Page rank). The page rank of node z in directed graph G, denoted by $PR(z)$, is given by the following recurrence relation:

$$PR(z) = \frac{1 - \alpha}{n} + \alpha \sum_{i=1}^{n} A_{iz} \frac{PR(i)}{k_{out}(i)}.$$

2.2.4 Link reciprocity

In a directed graph, links $e_1 = a \to b$ and $e_2 = b \to a$ are called reciprocal links. The tendency of node pairs to form reciprocal links can be quantified by the link reciprocity metric.

Definition 2.34 (Link reciprocity). The link reciprocity of a directed graph G, denoted by $r(G)$, is the fraction of reciprocal links in G, or equivalently the probability that a randomly selected link

is reciprocated. Therefore, the link reciprocity of G can be computed by the following formula:

$$r(G) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} A_{ji}}{l}.$$

Obviously, $r(G) = 0$ implies that G does not contain reciprocal links, while $r(G) = 1$ implies that all links are reciprocated. However, $r(G)$ does not tell us whether reciprocal links occur more or less frequently than it can be expected by random chance. Garlaschelli and Loffredo [2004] proposed a variant of link reciprocity that effectively takes into account the Erdős-Renyi model of random graphs as the null model. In this dissertation we call their measure normalized link reciprocity.

Definition 2.35 (**Normalized link reciprocity).** The normalized link reciprocity of a directed graph G, denoted by $r_n(G)$ is the Pearson correlation coefficient between entries A_{ij} and A_{ji} ($i \neq j$, which means that possible loops are excluded) of the adjacency matrix A of G. Therefore, $r_n(G)$ is computed by the following formula:

$$r_n(G) = \frac{\sum_{i \neq j} (A_{ij} - \overline{A})(A_{ji} - \overline{A})}{\sum_{i \neq j} (A_{ij} - \overline{A})^2} = \frac{r(G) - \overline{A}}{1 - \overline{A}},$$

where $\overline{A}$ is the average value of the entries in A with the main diagonal being excluded, i.e.

$$\overline{A} = \frac{\sum_{i \neq j} A_{ij}}{n(n-1)}.$$

As it can be observed $\overline{A}$ is actually the density of G. This quantity can be interpreted as the probability that two nodes are connected in a directed graph of the same size in which links are formed at random. Graphs obeying $r_n < 0$ are called anti-reciprocal meaning that their link reciprocity is smaller than expected by random chance, while $r_n > 0$ implies exactly the opposite.

2.2.5 Metrics of clustering

One of the basic property of social systems is the transitivity of relations. Strong homophily and transitivity together leads to the formation of the most cohesive sub-graphs that are also known as cliques.

Definition 2.36 (**Clique).** A clique is a completely connected sub-graph S of G. This means that every two nodes in S are directly connected. Being a clique is an invariant property under the node removal operation: if a node is removed from a clique the rest of the sub-graph is still a clique.

Transitivity can be measured both locally, at the level of ego-networks, as well as globally, at the level of graph partitions.

Definition 2.37 (**Ego network).** The ego network of node i in G, denoted by $Ego(i)$, is a sub-graph of G induced by i and its nearest neighbors.

The tendency of the neighbors of a node to be neighbors themselves is quantified by the clustering coefficient.

Definition 2.38 (**Clustering coefficient**). The clustering coefficient of node i in G, denoted by $CC(i)$, is the probability that two randomly selected neighbors of i are connected in G:

$$CC(i) = P(a \leftrightarrow b \mid i \leftrightarrow a, i \leftrightarrow b) = \frac{|\{(a,b) \ : \ a,b \in N_i, (a,b) \in E\}|}{\frac{k_i(k_i-1)}{2}},$$

where N_i is the set of nodes that are directly connected to i (the neighborhood of i), and k_i denotes the degree of i.

As it can be observed the clustering coefficient of i is actually the density of a graph that is obtained by removing i from $Ego(i)$. Therefore, $CC(i) = 0$ implies that $Ego(i)$ without i consists of isolated nodes, while $CC(i) = 1$ means that both $Ego(i)$ and $Ego(i)$ without i are cliques. The same approach, the density of an ego network without the ego node, can be taken for measuring local transitivity in directed graphs. Now, the number of links that could exist in the neighborhood of i is equal to $k_i(k_i - 1)$, where k_i is the total-degree of i. Therefore, the clustering coefficient of i is equal to:

$$CC(i) = \frac{|\{(a,b) \ : \ a,b \in N_i, (a,b) \in E\}|}{k_i(k_i - 1)}.$$

Informally speaking, a *community*, *cluster* or *module* in a graph is a subset of its nodes that are more densely connected among themselves than with the rest of the graph. Community structure is a typical feature of social networks, but is also characteristic to other types of complex networks. For example, tightly connected groups of nodes in the WWW often correspond to pages dealing with the same topic (as was shown in [Flake et al., 2002]). Uncovering communities helps us to understand internal structure of complex networks at a higher level of abstraction, to identify cohesive subgraphs, and to obtain readable maps of extremely large networks by constructing their coarse-grained descriptions (networks of communities). Therefore, identification of communities in a network, also known as *community detection*, is one of the important algorithmic problems in theory of complex networks. An important advance in community detection was made by Girvan and Newman [2002], who introduced a measure for the quality of a partition of a network into communities called *modularity* which is usually denoted by Q.

Definition 2.39 (**Graph partition**). A partition of $G = (V, E)$ into communities, denoted by $P(G)$, is an assignment of nodes in G to n_c sets of nodes C_i, $P(G) = \{C_i\}$, such that $V = \bigcup_i C_i$ and $(\forall i, j, i \neq j)\ C_i \cap C_j = \emptyset$.

Definition 2.40 (**Community membership function**). A community membership function, denoted by CM, is a function that maps a node to its community, with respect to some partition $P(G)$.

Definition 2.41 (**Intra-community link**). Link $e = (a, b)$ in G is an intra-community link, with respect to some partition $P(G)$, if a and b are members of the same community, i.e. $CM(a) = CM(b)$.

Definition 2.42 (**Inter-community link**). Link $e = (a, b)$ in G is an inter-community link, with respect to some partition $P(G)$, if it connects two nodes that belong to different communities, i.e. $CM(a) \neq CM(b)$.

Definition 2.43 (**Modularity**). The modularity of partition $P(G)$ is given by the following formula:

$$Q = \frac{1}{2l} \sum_{i,j} (A_{ij} - P_{ij}) \delta(i, j),$$

where $\delta(i,j)$ is one if nodes i and j are in the same community ($CM(i) = CM(j)$) or zero otherwise, and P_{ij} is the expected number of links between i and j considering some null model.

As the null model we can take the configuration model [Bollobás, 2001] – a random graph that has the same size and the same degree distribution as G. Then, the expected number of links between i and j is equal to

$$P_{ij} = \frac{k_i k_j}{2l}.$$

Now, Q can be rewritten as

$$Q = \sum_{c=1}^{n_c} \left[\frac{l_c}{l} - \left(\frac{d_c}{2l} \right)^2 \right],$$

where n_c is the number of communities, l_c denotes the total number of intra-cluster links in community c and d_c is the total degree of nodes in c. The first term of summand in the previous equation is the fraction of links inside community c. The second term represents the expected fraction of links inside c considering the configuration model. If the first term is much larger than the second then there are more links inside the community than expected by random chance. Therefore, the comparison with a random graph leads to the modularity-based definition of community: a sub-graph S of G can be considered as a community if

$$\frac{l_c}{l} \gg \left(\frac{d_c}{2l} \right)^2.$$

The modularity measure can be in a straightforward fashion generalized to directed, weighted and directed-weighted graphs [Fortunato, 2010; Leicht and Newman, 2008; Newman, 2004d].

Although widely used, the modularity measure has two weaknesses. Fortunato and Barthélemy [2007] showed that the modularity measure has an intrinsic scale that depends on the total number of links in the network (the resolution limit problem). Communities that are smaller than this scale cannot be detected through modularity maximization methods, even in the extreme case when they are complete sub-networks connected by single bridges. Good et al. [2010] showed that there are typically an exponential number of structurally diverse alternative partitions with modularity scores very close to the maximum (the degeneracy problem). Therefore, it is highly important to consider other notions of community when performing community detection relying on the modularity measure. Radicchi et al. [2004] proposed definitions of two types of communities that are based on node internal and external degree.

Definition 2.44 (**Node internal degree).** The internal degree of node i in G, denoted by $k_{int}(i)$, with respect to some partition $P(G)$, is the number of intra-community links incident with i:

$$k_{int}(i) = \sum_{j \in CM(i)} A_{ij}.$$

Definition 2.45 (**Node external degree).** The external degree of node i in G, denoted by $k_{ext}(i)$, with respect to some partition $P(G)$, is the number of inter-community links incident with i:

$$k_{ext}(i) = \sum_{j \notin CM(i)} A_{ij}.$$

Definition 2.46 (**Radicchi strong community**). Subgraph C is a Radicchi strong community (community in a strong sense) if

$$(\forall i \in C) k_{int}(i) > k_{ext}(i).$$

Definition 2.47 (**Radicchi weak community**). Subgraph C is a Radicchi weak community (community in a weak sense) if

$$\sum_{i \in C} k_{int}(i) > \sum_{i \in C} k_{ext}(i).$$

A comprehensive overview of other quantitative notions of community and community detection techniques can be found in [Fortunato, 2010].

2.3 Basic models of complex networks

Mathematical models of complex networks help us to understand properties of complex networks and to make predictions about their evolution. If some property of a real-world network is reproducible by a theoretical model then it can be explained by the founding principles of the model. By studying mathematical models of complex networks we build our intuition about networks in the sense that we associate evolutionary principles to typically observable outcomes of those principles.

The study of disordered graph models started with the pioneering work of Erdős and Renyi who proposed what is nowadays called the Erdős-Renyi (ER) model of random graphs [Bollobás, 2001; Erdős and Rényi, 1959, 1960]. The model is extensively investigated by the mathematical community since it is the base of the probabilistic method used in graph theory. Let $S(N, L)$ denote the set of all graphs having N nodes and L links. Let us suppose that we want to prove that there is a graph from $S(N, L)$ exhibiting certain property P. Naturally, the problem can be solved by taking a brute-force, constructivist approach: we construct members of $S(N, L)$ in some order and check whether the last constructed member satisfies P. The profound idea of Erdős is that we can solve the problem in a non-constructivist way by showing that a randomly selected graph from $S(N, L)$ exhibits P with a non-zero probability.

Definition 2.48 (**Erdős-Renyi random graph**). A random graph $G_{er}(N, L)$ is a labeled graph containing N nodes and L links that are randomly selected (with equal probabilities) from the set of all possible links that can be formed among those N nodes.

The closely related but a slightly broad view of random graphs is given by the following definition originally introduced by Gilbert [1959].

Definition 2.49 (**Random graph**). A random graph $G(N, p)$ is a labeled graph consisting of N nodes where each two nodes are connected with some fixed probability p.

$G_{er}(N, L)$ is an outcome of an stochastic process defined over the sample space $S(N, L)$ where the members of $S(N, L)$ have equal probabilities of realization. On the other hand, $G(N, p)$ is an outcome of an stochastic process defined over a bigger sample space $S(N)$ which includes every graph with N nodes. The members of $S(N)$ are not equiprobable – the most probable realizations are those graphs with pM links where M is the maximal number of links among N nodes ($M = N(N-1)/2$). In the asymptotic limit $N \to \infty$ the models $G(N, p)$ and $G_{er}(N, pM)$ are interchangeable [Boccaletti et al., 2006; Bollobás and Riordan, 2005]. This means that we can use both models to study properties of large-scale random graphs. However, Gilbert's definition is usually preferred since it enables easier analytical treatment of random graphs.

In analysis of complex real-world networks the random graph model is used as a null model. Random graph represents a typical graph of some size, typical considering the ensemble of all graphs having that size. Therefore, statistical properties of real-world networks are usually firstly compared to analytical predictions obtained using the random graph model. If an empirically observed statistic of a real-world network significantly differs from the prediction obtained using the random graph model then we have some property that is not typical in the sense that is not characteristic for a majority of graphs having the same number of nodes and the same number of links. The basic properties of random graphs can be summarized as follows:

1. The degree distribution of a random graph $G(N,p)$ follows the binomial distribution $B(N-1,p)$ that can be well approximated by the Poisson distribution $Pois(\langle k\rangle)$ for large N, where $\langle k\rangle$ is the average degree.

2. The small-world coefficient of a random graph grows logarithmically with the size of the graph which means that random graphs are small-worlds. The small-world coefficient of $G(N,p)$ can be approximated as:
$$\mathrm{SW} \approx \frac{log(N)}{log(p(N-1))} = \frac{log(N)}{log(\langle k\rangle)}.$$

3. A link in $G(N,p)$ is created independently of previously established links. Therefore, the clustering coefficient of $G(N,p)$ is equal to p. This implies that large and sparse random graphs exhibit a low degree of local clustering. Another consequence of independent formation of links is that $G(N,p)$ is non-assortative.

4. The structure of connected components of $G(N,p)$ depends on p. Namely, there is a critical connection probability $p_c = 1/(N-1)$ that corresponds to a critical average degree $\langle k_c\rangle = 1$. If the average degree of $G(N,p)$ is higher than the critical average degree then $G(N,p)$ almost surely has a giant connected component. On the other hand, if the average degree is less than critical then $G(N,p)$ almost surely does not have a giant connected component.

The empirical investigation of three real-world networks conducted by Watts and Strogatz [1998b] showed that examined networks possess the small-world property but exhibit drastically higher degree of local clustering than comparable random graphs. Watts and Strogatz observed that the highest degree of local clustering is characteristic for regular ring lattices – regular graphs in which nodes are placed on a ring and each node is besides its ring neighbors also connected to their ring neighbors. On the other hand, those structures do not exhibit the small-world property. Therefore, regular ring lattices have exactly the opposite characteristics of random graphs regarding the length of shortest paths and the intensity of local clustering. Watts and Strogatz investigated a model which interpolate between ring lattices and ER random graphs. In their model we start from a ring lattice and then reconnect each link with a probability p which is the parameter of the model. For $p = 1$ each link is rewired which means that a random graph is obtained. Watts and Strogatz showed that for $0.01 < p < 0.1$ the rewiring procedure results in a graph that exhibit both the small-word property and a high degree of local clustering.

Both Erdős-Renyi random graphs and Watts-Strogatz small-world graphs are homogeneous in the sense that each node has approximately the same degree close to the average degree. In contrast, many real-world networks belong to a class of heterogeneous networks characterized by a heavy-tailed, power-law degree distribution [Albert and Barabási, 2002; Boccaletti et al., 2006; Newman, 2003b], which means that the distribution has a long right tail of values that are far above the average degree.

Definition 2.50 (**Power-law).** Discrete probability distributions of the form $P(k) = Ck^{-\gamma}$, where C and γ are constants, are said to follow a power-law. The constant γ is called the scaling exponent of the power-law. Smaller γ implies slower decay of $P(k)$ and consequently causes more skewed distribution. The constant C is determined by the normalization requirement that $\sum_{k=k_{min}}^{\infty} P(k) = 1$.

Definition 2.51 (**Scale-free network).** Networks whose degree distributions follow a power-law in the tail, $P(k) \sim Ck^{-\gamma}$, are known as scale-free networks.

If $P(k)$ follows the power-law $P(k) \sim Ck^{-\gamma}$ then the plot of $P(k)$ on logarithmic scales appears as a straight line of slope $-\gamma$ ($\log P(k) \sim -\gamma \log k$). Complementary cumulative degree distribution of scale-free networks also appears as a straight line on log-log plots but with slope $-(\gamma - 1)$ rather than $-\gamma$ [Newman, 2003b, 2005]. For $P(k) \sim Ck^{-\gamma}$ we have that $CCD(k) = \sum_{i=k}^{\infty} P(i) \sim \frac{C}{\gamma-1} k^{-(\gamma-1)}$. This can be easily seen if we treat (and thus approximate) discrete random variable D, that represents the degree of a randomly chosen node, as a continuous random variable. Then the sum of $P(i)$ can be expressed as an integral of a power function which is also a power function (with the exponent increased by one).

Barabási and Albert [Barabasi and Albert, 1999] proposed a one–parameter model, known as the BA model, for generating scale-free networks. The model is based on two principles:

- Network growth. In each iteration of the model one new node is created and it will establish m connections with previously created nodes, m is the only parameter of the model.
- Preferential attachment. The probability that the newly introduced node n establishes connection with node i depends on the degree of node i, i.e.

$$P(n \leftrightarrow i) = \frac{\text{degree}(i)}{\sum_{j \in S} \text{degree}(j)},$$

  where S denotes the set of "old" nodes – nodes created in previous growth steps.

As it can be observed the preferential attachment probability is based on the "rich get richer" principle which is also known as the principle of cumulative advantage or the Matthew effect. Large networks generated by the BA model satisfy the power-law with $\gamma \approx 3$ ($\gamma \to 3$ as $N \to \infty$, where N is the size of the network). In order to show that network growth and preferential attachment are necessary ingredients that lead to a power-law degree distribution, Barabási and Albert examined a model that keeps the growing character of the network, but uses uniform attachment probability (each node has equal probability to establish a connection with the newly added node). This model results in networks whose degree distribution decays exponentially following $P(k) = C \exp(-\beta k)$. Krapivsky et al. [2000] showed that only linear preferential attachment leads to power-law degree distributions: sub-linear preferential attachment probability leads to stretched exponential degree distributions, while super-linear attachment probability produces "winner takes all" star topologies. Various generalizations of the BA model were proposed in order to obtain tunable power-law scaling exponent [Dorogovtsev et al., 2000], tunable clustering coefficient [Holme and Kim, 2002], tunable assortativity index [Guo et al., 2006] and community organization [Li and Maini, 2005; Pollner et al., 2006]. The model can be also generalized in a straightforward manner for directed networks [Bollobás et al., 2003]. Amaral et al. [2000] investigated the effects of node aging and the limited capacity of nodes on the local structure of networks whose evolution is governed by the preferential attachment principle. They showed that time or capacity constraints when incorporated into the BA model lead to a truncated power-law degree distributions – distributions that have a power-law regime followed by a sharp cut-off.

Another class of models of scale-free networks is based on a copying mechanism [Kleinberg et al., 1999; Krapivsky and Redner, 2005; Kumar et al., 2000; Leskovec et al., 2007]. The core intuition behind copying-based models is related to topic-oriented citation practices that occur for example in the WWW or scientific publishing. For example, if paper A references paper B on the same topic then it is likely that A also references a paper on the same topic that is referenced by B. Two essential steps are performed in copying-based models when a new node A integrates into a network:

- A establishes connection with a "prototype" node that is chosen uniformly at random.
- The rest of links incident with A are distributed according to the following rule: A connects to a randomly selected node with probability p, while with probability $(1 - p)$ A establishes connection with a randomly selected neighbor of the prototype node.

The principle of connecting to neighbors of prototype nodes is effectively the principle of preferential attachment. Namely, the probability that node B is the neighbor of prototype node C is proportional to the degree of B because C is chosen uniformly at random, i.e. highly connected nodes have higher probability to be neighbors of a randomly selected node compared to loosely connected nodes.

Chapter 3

Software networks

This chapter of the dissertation is devoted to the extraction and analysis of software networks. In Section 3.1 we will define different types of software networks. The relationship between software networks and software design metrics is explained in Section 3.2. The first original contribution of the dissertation is presented in Section 3.3 where we argue that graph clustering evaluation metrics can be viewed as software metrics and applied to measure cohesion of software entities [Savić and Ivanović, 2014]. In Section 3.4 we will review existing approaches to the extraction of software networks. In the same section we will present SNEIPL [Savić et al., 2012, 2014] – a novel, language-independent approach to the extraction of software networks which is one of the original contributions of the dissertation. In Section 3.5 we will review existing research works focused on the analysis of real-world, large-scale software networks. In the same section we will also present analysis of connected components, degree distributions and hubs (highly connected nodes) in five class collaboration networks extracted using SNEIPL which is also the original contribution of the dissertation. Finally, Section 3.6 provides concluding remarks and directions for future research work.

3.1 Taxonomy of software networks

Software networks are directed graphs of static dependencies between source code entities. High-level programming languages provide mechanisms to define or declare different types of software entities at different levels of abstraction in order to support modularity and reuse of source code. In general, to each entity is assigned a name that is used to reference entity by other entities defined in other parts of the source code. Thus, software networks can be viewed as networks connecting identifiers introduced in the source code. Depending on the types of entities and relationships, software networks can be classified either as homogeneous or heterogeneous. Homogeneous software networks encompass software entities of the same type that are connected by links denoting the same kind of relationship. On the other side, in heterogeneous software networks entities and/or connections are of different types.

In general, high-level programming languages provide at most three levels of abstraction to define entities that can be referenced:

- Function-level entities are referable entities at the lowest level of abstraction. They cannot be composed out of non-anonymous higher-level entities. Function-level entities in procedural programming languages are procedures (functions, subroutines), global variables (variables that

are not declared inside functions) and user-defined data types. In object-oriented languages methods and class attributes belong to this category.

- Class-level entities are entities at the middle level of abstraction. They group related lowest level entities and can also contain definitions of other class-level entities. Class-level entities in procedural programming languages are definition and implementation modules, while in object-oriented languages class-level entities are classes and interfaces.

- Package-level entities are entities at the highest level of abstraction. They group related entities from the lower levels of abstraction. Pure procedural programming languages do not have package-level entities. Packages, namespaces and units in OO languages are considered as package-level entities.

Links in software networks that connect entities from the same level of abstraction will be called "horizontal". Clearly, all links in homogeneous software networks are horizontal. On the other hand, links in heterogeneous software networks that connect entities appearing at different levels of abstraction will be called "vertical".

Most programs written in a procedural programming language consists of procedures (also called subroutines or functions) which collaborate using the call-return mechanism provided by the language. In object-oriented software systems, software entities known as methods collaborate using the same mechanism. Call-return relationships between procedures define a homogeneous software network that is often referred to as a *static call graph*.

Definition 3.1 (**Static call graph (SCG)).** A static call graph encompasses all functions defined or declared in a software system. Two functions A and B are connected by the directed link $A \rightarrow B$ if A explicitly calls B.

Static call graphs for object-oriented (OO) software systems are also known as *method collaboration networks* [Hylland-Wood et al., 2006]. It is important to observe that function calls through a reflection mechanism, if it is present in a language, do not form static (structural, compile-time) dependencies between functions, but run-time dependencies.

Similarly as for procedures and methods, we do not make the explicit distinction between global variables in procedural style and class member variables (class attributes) in OO style. Dependencies between functions and global variables can be described by heterogeneous software networks which we call FUGV (Function Uses Global Variable) networks.

Definition 3.2 (**FUGV graph).** A FUGV graph encompasses all functions and global variables defined in a software system. Function A is directly connected to global variable B by link $A \rightarrow B$ if B is used (read or written) in the statements that constitute the body of A.

It can be observed that FUGV networks are bipartite directed graphs. FUGV networks can be used to compute metrics measuring (lack of) cohesion of software entities because in those metrics we are interested to know if two different functions defined in a same entity access at least one common global variable [Briand et al., 1998].

Collaborations of classes and interfaces in an OO software system constitute a *class collaboration network* (CCN). By the term class collaboration network will be also assumed the term *module collaboration network* that denotes collaborations of modules in procedural programming languages.

Definition 3.3 (**Class collaboration network (CCN)).** A class collaboration network encompasses all classes and interfaces defined in a software system. Two nodes A and B contained in the CCN are connected by directed link $A \rightarrow B$ if the class or interface represented by node A references the class or interface represented by node B.

Class A can reference another class B in many ways: by extending the functionality of B, defining a member variable (class attribute) whose type is B, realizing a method which calls some method defined in B, etc. Class collaboration networks can be viewed as simplified class diagrams that preserve only the existence of relations between classes, and discard other types of information about nodes (classes) and links (OO relations). Additionally, homogeneous software networks that represent different forms of class coupling, such as inheritance trees or aggregation networks, can be isolated from class collaboration networks [Wheeldon and Counsell, 2003].

At the highest level of abstraction, package-level entities form a *package collaboration network*.

Definition 3.4 (**Package collaboration network (PCN)).** A package collaboration network encompasses all packages defined in a software system. Two packages PA and PB are connected by the directed link $PA \rightarrow PB$ if package PA contains a class or interface that references at least one class or interface from package PB.

Hierarchy tree is a heterogeneous software network that contains all entities defined in a software system. This type of network captures vertical dependencies between entities.

Definition 3.5 (**Hierarchy tree).** A hierarchy tree contains all package-, class- and function-level entities defined in a software system. Two entities A and B are connected by the directed link $A \rightarrow B$ if entity A defines or declares entity B.

Hierarchy tree can be used when we are interested to know where an entity is defined (the parent of the entity), and which other entities it defines (the children of the entity).

3.1.1 General Dependency Network

In this thesis we introduce a heterogeneous software network called *General Dependency Network* (GDN). GDN is a directed and attributed multigraph: the nodes have type and name, while the links have type and weight (the strength of connection) as attributes. Also, a pair of nodes can be connected by parallel links denoting different coupling types. GDN nodes represent package-, class- and function-level entities defined in the corresponding software system. GDN links represent various types of relations: CALLS relations between functions, REFERENCES relations between package-level entities, REFERENCES relations between class-level entities, USES relations between functions and variables, and CONTAINS relations that reflect the hierarchy of entities. There are also seven types of relations that represent different forms of coupling between class-level entities:

- EXTENDS relation $A \rightarrow B$ denotes that A extends the functionality of B,
- IMPLEMENTS relation $A \rightarrow B$ denotes that A implements the declarations contained in B,
- INSTANTIATES relation $A \rightarrow B$ denotes that A instantiates the objects of B,
- AGGREGATES relation $A \rightarrow B$ denotes that A contains a global variable whose type is B,
- WEAK_AGGREGATION relation $A \rightarrow B$ denotes that A contains at least one function that declares a local variable whose type is B,

- PARAMETER_TYPE relation $A \to B$ denotes that A contains at least one function that has a parameter whose type is B,
- RETURN_TYPE relation $A \to B$ denotes that A contains at least one function whose return type is B.

It can be observed that GDN is designed to represent a union of collaboration networks at different levels of abstractions with incorporated CONTAINS links that maintain the hierarchy of entities. Thus, all previously defined software networks can be obtained by GDN filtration, i.e. by the selection of nodes of specified types that are connected by links of specified types.

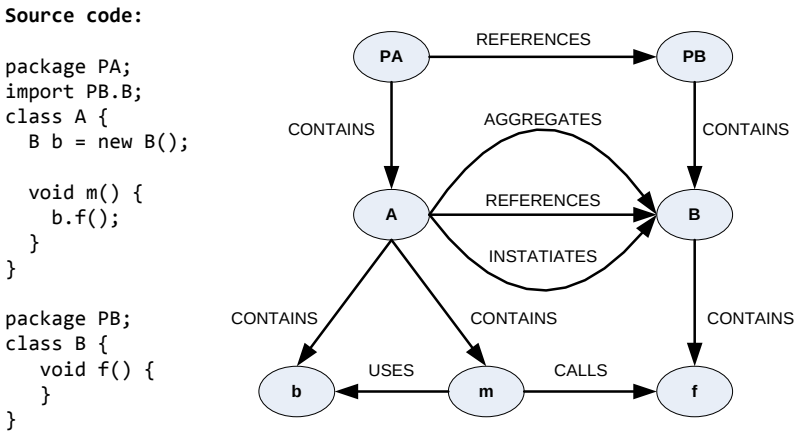


FIGURE 3.1: General Dependency Network for a software system consisting of two classes.

Figure 3.1 shows the GDN representation of a simple software system written in Java that consists of two classes (A and B) contained in two packages (PA and PB). The selection of nodes representing packages connected by REFERENCES links isolates the package collaboration network of the system. Similarly, the class collaboration network is the sub-network of the GDN induced by nodes representing classes connected by REFERENCES links. The static call graph can be obtained by the selection of nodes representing functions connected by CALLS links ($m \to f$). The FUGV network consists of USES links connecting functions to global variables ($m \to b$). The hierarchy tree has the same set of nodes as the GDN, but the set of links is restricted to CONTAINS links. AGGREGATES and INSTANTIATES links appear in the networks showing specific forms of coupling between classes.

3.2 Software networks and software design metrics

Software engineering practice, or even the application of simple software metrics such as LOC, can show us that modern software systems are complex artifacts. An essential complexity of software is a consequence of a high number of software entities defined in the source code and the complex interactions among them [Brooks, 1987]. Most of traditional software metrics used to estimate software complexity (such as LOC, Cyclomatic complexity, Halstead metrics, etc.) are mainly oriented towards the internal complexity of software entities. They are used to identify algorithmically complex entities that should be re-decomposed into the sets of smaller, less complex, easily maintainable entities that can be reused later as the software system evolves. The main characteristic of the metrics of internal complexity is that they do not take into account existing interactions between software entities. The

complexity of interactions among software entities can be quantified by the class of software design metrics that reflect *coupling*, *cohesion*, *inheritance*, and *invocation*. Widely known and used metrics from this category are those introduced in the Chidamber-Kemerer metric suite [Chidamber and Kemerer, 1994]: CBO (Coupling between objects), DIT (Depth of inheritance tree), NOC (Number of children), LCOM (Lack of cohesion of methods), and RFC (Response for a class). In order to compute software design metrics, source code entities and relations between them have to be identified, which means that network representations of the software system have to be extracted.

3.2.1 Coupling metrics

"Low coupling, high cohesion" is one of the basic design principles in software engineering [Yourdon and Constantine, 1979]. This principle states that the coupling between modules of a software system has to be minimal as possible keeping at the same time strong relations between elements of each module. In other words, the principle itself promotes encapsulation and modularity of software systems. In order to measure the level of coupling among classes defined in an object-oriented software system Chidamber and Kemerer [1994] proposed a metric called Coupling Between Objects (CBO). Two classes are coupled when methods declared in one class use methods or class attributes of another class. CBO for a class is the number of other classes that the class is coupled to, i.e the number of unique classes referenced by the class plus the number of classes that refer to the class. In other words, CBO is the total degree of a node representing the class in appropriate class collaboration network. In accordance with the principle of low coupling low values of the CBO metric are desirable in software engineering practice.

Internal reuse (export, afferent coupling, fan-in) and internal aggregation (import, efferent coupling, fan-out) jointly constitute the coupling of a software entity. The degree of internal reuse of an entity is the number of other entities defined in the system that directly use the entity [Fenton, 1991]. Internal reuse can be considered as good software engineering practice, since it reduces or eliminates duplicated code inside the system. However, there are two potential negative aspects of extensive internal reuse: high criticality and low testability of highly reused entities [Briand et al., 1999]. High criticality means that defects present in an excessively internally reused entity are more likely to propagate to the rest of the system. Additionally, any significant change of highly reused entity can cause a large chain of changes in entities that directly and indirectly depend on it. Those entities when internally complex are also difficult to test if defects need to be propagated to dependable entities in order to be detected. The degree of internal aggregation of an entity is the number of other entities defined in the system that are used by the entity. As summarized by Briand et al. [1999], high internal aggregation can negatively impact the following external attributes of software systems: understandability, error-proneness, maintainability, and external reusability. Internal reuse and internal aggregation of an entity can be measured by the in-degree and out-degree of the entity in the appropriate software network. In other words:

- In- and out-degree of a node in the package collaboration network represent the degree of internal reuse and internal aggregation of corresponding package.
- In- and out-degree of a node in the class collaboration network represent the degree of internal reuse and internal aggregation of corresponding class or interface.
- In- and out-degree of a node in the static call graph represent the degree of internal reuse and internal aggregation of corresponding function.

A class hierarchy can be described by a software network that is known as an inheritance graph. Inheritance graph is a sub-network of the class collaboration network which is restricted to EXTENDS relation between classes. This means that two classes A and B are connected in the inheritance graph if and only if A extends the functionality of B. Object oriented programming languages do not allow cyclic EXTENDS relations. This implies that inheritance graphs are acyclic directed graphs. Node R is called the root of the inheritance graph if its out-degree is equal to zero and its in-degree is different than zero. One class hierarchy can have multiple roots. In the case of programming languages that prohibit multiple inheritance each connected component of the inheritance graph has exactly one root node. Additionally, there is an unique path connecting an arbitrary class A to the root node.

In order to measure the quality of class hierarchies Chidamber and Kemerer [1994] introduced two metrics: NOC (Number of children) and DIT (Depth of inheritance tree). The NOC of class A is the number of other classes that directly extends the functionality of A. High values of NOC indicate classes that are highly internally reused through inheritance. Therefore, both positive and negative aspects of high internal reuse can be attributed to this metric as well. The DIT of class A is the length of the path connecting A to the root node. In the case of programming languages which allow multiple inheritance, the DIT of A is the length of the longest path connecting A to a root class. DIT indicates the degree of class specialization. Classes with high DIT are classes that indirectly reused a large number of other classes. Those classes are hard to understand compared to classes with lower DIT since they inherited a large number of methods. Also their existence implies deep class hierarchies and consequently higher design (structural) complexity.

3.2.2 Cohesion metrics

Cohesion of a software module reflects how strongly related are the elements of the module. Cohesion metrics are commonly based on two forms of method coupling, data and call coupling, that can be identified using FUGV networks and static call graphs, respectively. Perhaps the widest known cohesion metric in software engineering is LCOM [Chidamber and Kemerer, 1994]. LCOM is an inverse cohesion metric: a low value of LCOM indicates a high class cohesion and vice versa. LCOM is based on a specific coupling between methods: two methods in a class are considered as data coupled if they use at least one common class attribute. Then LCOM is the number of non-coupled methods (P) reduced by the number of coupled methods (Q) if $P > Q$, or zero otherwise.

The approach of Hitz and Montazeri [1995] to measure cohesion of software entities followed the previous research. For a class we can construct graph G whose nodes are methods defined in the class, and two methods are connected by an undirected link if they are data coupled. The LCOM of Hitz and Montazeri is the number of connected components in G. The same authors also proposed another variant of the same metric where G includes method calls relations. Finally, they introduced a metric called *connectivity* which quantifies to which extent G is far from being completely connected.

Bieman and Kang [1995] introduced two cohesion metrics called tight class cohesion (TCC) and loose class cohesion (LCC). The basic element in their metrics is again a graph that shows relations among methods of a class. TCC/LCC is the density (the actual number of links divided by the maximal number of links) of a TCC/LCC graph. Two methods are connected in TCC graph if they both access the same variable or there is a direct call between them. LCC graph is an extension of TCC graph that includes indirect method calls.

Lee et al. [1995] introduced a class cohesion metric based on information flow. The basic idea is that the strength of call coupling between invoking and invoked method is determined by the

number of parameters of invoked method: the more information passed through formal parameters, the stronger call coupling between classes. Then, the cohesion of a method is defined as the number of calls to other methods multiplied by the number of formal parameters. Finally, the cohesion of a class is the sum of cohesion of its methods.

3.2.3 Hierarchy trees and compositional software metrics

Software entities at a higher level of abstraction are composed out of software entities at lower levels of abstraction. Some property of a software entity can be expressed considering its constitutional parts. For example, the internal complexity of a high-level entity is usually viewed as a function of internal complexities of its constituent parts or even more simpler as the number of constituent parts. Therefore, hierarchy trees are naturally used in the computation of compositional metrics, either alone or in combination with other software networks. For example, software metrics such as NOC for packages (the number of classes and interface contained in a package), NOM/NOA (the number of methods/attributes defined in a class) and abstractness (the number of abstract classes divided by the total number of classes in a package) can be easily computed relying only on appropriate hierarchy tree. On the other hand, the computation of the RFC (response for a class) metric from the Chidamber-Kemerer suite requires information contained in the static call graph and hierarchy tree of the system.

3.3 Graph clustering evaluation metrics as software metrics

In this thesis we argue that graph clustering evaluation (GCE) metrics can be applied on graph representations of software systems in order to evaluate the degree of cohesiveness of software entities. If a software system is designed according to the principle of low coupling and high cohesion then highly cohesive modules can be viewed as clusters in a software network that encompasses software entities defined at the lower level of abstraction.

From the review of widely used software engineering cohesion metrics (see Section 3.2.2) it can be concluded that the cohesiveness of a software entity is estimated in isolation. In other words, those metrics rely only on internal dependencies. However, external dependencies can also be important when estimating cohesiveness of software modules. Firstly, a module that has much more external than internal dependencies hardly can be considered as strongly cohesive regardless of the density or the connectedness of its internal parts. Secondly, having two modules that have the same degree of internal density the one with the smaller number of external dependencies can be considered as more cohesive compared to the other.

3.3.1 GCE metrics and software networks

Let $G = (V, E)$ be a directed graph where V is the set of nodes and E is the set of links. Let C denote a cluster in V ($C \subseteq V$), and let $c \in C$. An intra-cluster link emanating from c connects c to another node from C, while an inter-cluster link emanating from c connects c to a node that does not belong to C. Intra-cluster (inter-cluster) out-degree of node c is the number of intra-cluster (inter-cluster) links emanating from c.

The most basic formulation of the graph partitioning problem asks for a division of the set of nodes into balanced, disjoint subsets of nodes such that the edge cut (links connecting nodes from different clusters) is minimized. Therefore, the basic graph clustering evaluation (GCE) metrics are based on

the size of the edge cut. Let E_C denote the size of the cut (the number of inter-cluster links) for cluster C,

$$E_C = \mid \{(x, y) \ : \ x \in C, y \notin C\} \mid = \sum_{x \in C} \text{inter-cluster out-degree}(x),$$

I_C the number of intra-cluster links for C,

$$I_C = \mid \{(x, y) \ : \ x \in C, y \in C\} \mid = \sum_{x \in C} \text{intra-cluster out-degree}(x),$$

N_C the number of nodes in C, and N the number of nodes in the graph. Then cut based GCE metrics, conductance, expansion and cut-ratio, are defined as follows [Leskovec et al., 2010]:

1. The conductance of cluster C is the size of the cut normalized by the total number of links incident to nodes contained in C,

$$\text{Conductance}(C) = \frac{E_C}{E_C + I_C}.$$

2. The expansion of cluster C is the size of the cut divided by the total number of nodes in C,

$$\text{Expansion}(C) = \frac{E_C}{N_C}.$$

3. The cut-ratio of cluster C is the size of the cut divided by the size of the maximal possible cut,

$$\text{Cut-ratio}(C) = \frac{E_C}{N_C(N - N_C)}.$$

Probably the oldest definition of graph cluster originates from circuit theory. Luccio and Sami [1969] introduced the notion of LS-set that is also known as Raddichi strong community [Radicchi et al., 2004]. For directed graphs, an LS-set is a subgraph such that the intra-cluster out-degree of each node in the set is higher than its inter-cluster out-degree. The nodes having zero out-degree are not taken into account. If the number of intra-cluster links is higher than the number of inter-cluster links then the subgraph is considered as Radicchi weak cluster. Each Radicchi strong cluster is at the same time Radicchi weak cluster, while the converse is not generally true. If a cluster is Radicchi weak or strong then its conductance is smaller than 0.5. The difference between the number of intra- and inter-cluster links inspired ODF (out-degree fraction) family of cluster quality measures [Leskovec et al., 2010]:

1. The maximum-ODF of cluster C is the maximum fraction of inter-cluster links of a node observed in the cluster,

$$\text{Maximum-ODF}(C) = \max_{c \in C} \frac{\mid \{(c, d) \ : \ d \notin C\} \mid}{D_{out}(c)},$$

   where $D_{out}(c)$ stands for out-degree of node c.

2. The average-ODF of cluster C is the average fraction of inter-cluster links of nodes from C,

$$\text{Average-ODF}(C) = \frac{1}{N_C} \sum_{c \in C} \frac{\mid \{(c, d) \ : \ d \notin C\} \mid}{D_{out}(c)}$$

3. The Flake-ODF of cluster C is the fraction of nodes in C that have higher intra-cluster out-degree than inter-cluster out-degree,

$$\text{Flake-ODF}(C) = \frac{|\{x : x \in C, E_{out}(c) < D_{out}(c)/2\}|}{N_C},$$

where $E_{out}(c)$ is the inter-cluster out-degree of node c. In other words Flake-ODF measures the extent to which C is close to being a Radicchi strong cluster: if Flake-ODF(C) is equal to 1 then C is Radicchi strong.

We can distinguish between two types of links in General Dependency Network (see Section 3.1.1): "vertical" (CONTAINS) links that maintain the hierarchy of software entities and "horizontal" links that show dependencies between entities from the same level of abstraction. Horizontal links of GDN can be separated into two categories:

- Intra-cluster link connects two entities from the same level of abstraction that are contained in the same software entity, i.e. $A \to B$ is an intra-cluster link if and only if

$$(\exists O)\ \text{CONTAINS}(O \to A) \wedge \text{CONTAINS}(O \to B).$$

- Inter-cluster link connects two entities from the same level of abstraction that are contained in two different software entities, i.e. $A \to B$ is an inter-cluster link if and only if

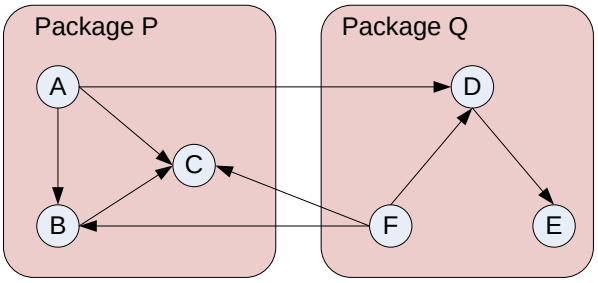
$$(\exists O_1, O_2)\ O_1 \neq O_2 \wedge \text{CONTAINS}(O_1 \to A) \wedge \text{CONTAINS}(O_2 \to B).$$

The separation of links into intra- and inter-cluster links enables us to apply graph clustering evaluation (GCE) metrics to:

1. Class collaboration networks in order to evaluate cohesiveness of packages.
2. Static call graphs extended with FUGV graphs in order to evaluate cohesiveness of classes in OO systems or modules in procedural software systems.

It can be easily seen from the definition of GCE metrics that only the Flake-ODF metric measures cohesion, while other examined GCE metrics are inverse cohesion measures. As it can be observed GCE metrics do not ignore references to external entities. To the contrary, they use the number of dependencies to external entities to determine to what extent the entity is isolated from the rest of the system. In other words, GCE metrics are based on the following principle: an entity can be considered as highly cohesive if its elements are better connected between themselves than with the entities defined outside the entity.

Figure 3.2 shows a class collaboration network that represent a simple software system that consists of two packages P and Q where both packages contain three classes. It can be observed that class F has higher inter-cluster out-degree than intra-cluster out-degree: this class references one class from its package and two classes from package P. Therefore, package Q is not Radicchi strong cluster. This package is neither Radicchi weak cluster since the number of intra-cluster links is not higher than the the number of inter-cluster links. It can also be observed that the system presented in Figure 3.2 can be refactored in order improve the overall degree of cohesion: if we move class F from package Q to package F then both packages will be Radicchi strong.



	Package P	Package Q
Intra-cluster links	3	2
Inter-cluster links	1	2
Radicchi strong	yes	no
Radicchi weak	yes	no
Conductance	0.25	0.5
Expansion	0.33	0.66
Cut-ratio	0.11	0.22
Maximum-ODF	0.33	0.66
Average-ODF	0.11	0.22
Flake-ODF	1.0	0.66

FIGURE 3.2: Class collaboration network of a simple software system and appropriate cluster quality measures.

3.3.2 Theoretical analysis

Briand et al. [1996, 1998] defined several properties that a software metric should satisfy in order to be theoretically sound (lack of) cohesion metric. Those properties are:

1. **Nonnegativity**. A cohesion (lack of cohesion) metric cannot take a negative value.

2. **Normalization**. The metric value belongs to an interval $[0, M]$, where M is the fixed maximal value.

3. **Null value**. The cohesion of a software entity is null if R_c is empty, where R_c denotes the set of relationships within the software entity. This means that if there are no intra-cluster links the cohesion of the entity should be zero. On the other side, a metric measuring the lack of cohesion should be zero if R_c is maximal. R_c is maximal if all possible relationships within the entity are present.

4. **Maximum value**. If R_c is maximal then a metric of cohesion takes the maximal value. If $R_c = \emptyset$ then a metric measuring the lack of cohesion takes the maximal value.

5. **Monotonicity**. Let e be a software entity. Let e' be the software entity such that $R_e \subseteq R_{e'}$, i.e. we added some relationships (intra-cluster links) in e to obtain e'. Then the following inequalities must hold

$$C(e) \leq C(e'), \tag{3.1}$$
$$L(e) \geq L(e'), \tag{3.2}$$

   where C and L denote a cohesion and a lack of cohesion metric, respectively. In other words, the property states that addition of new intra-cluster links must not decrease/increase the value of the cohesion/lack of cohesion metric.

6. **Merge property**. Let e_1 and e_2 be two unrelated (unconnected) software entities. This means that e_1 does not reference e_2 and vice versa, i.e. there are no relationships (inter-cluster links) between e_1 and e_2. Let e be the software entity which is the union of e_1 and e_2. Then the following inequalities must hold

$$C(e) \leq max\{C(e_1), C(e_2)\}, \tag{3.3}$$
$$L(e) \geq min\{L(e_1), L(e_2)\}. \tag{3.4}$$

This property says that merging two unrelated entities must not increase/decrease the value of the cohesion/lack of cohesion metric.

As a first step in our theoretical analysis of graph clustering evaluation metrics, we state and prove the following lemma that will be frequently used in this section.

Lemma 1. Let P and Q be two nonnegative numerical properties of a module. If P and Q are additive under the merge operation then a (lack of) cohesion metric defined as $C = P/Q$ satisfies the merge property.

Proof. Let m_1 and m_2 be two modules. Without loss of generality we can assume that $C(m_1) \leq C(m_2)$. Due to the nonnegativity of P and Q the following inequality holds

$$P(m_1)Q(m_2) \leq P(m_2)Q(m_1). \tag{3.5}$$

Let m denote the module obtained by merging m_1 and m_2. Due to the additivity of P and Q we have that

$$C(m) = \frac{P(m_1) + P(m_2)}{Q(m_1) + Q(m_2)}.$$

C is a cohesion metric. Let us suppose that the merge property is not satisfied, i.e.

$$C(m) > max\{C(m_1), C(m_2)\} = C(m_2).$$

By elementary algebraic transformation we obtain that

$$P(m_1)Q(m_2) > P(m_2)Q(m_1) \tag{3.6}$$

which is in contradiction with inequality 3.5.

C is a lack of cohesion metric. Again we give a proof by contradiction. If

$$C(m) < min\{C(m_1), C(m_2)\} = C(m_1)$$

then by elementary algebraic transformation we again obtain inequality 3.6. □

From the definition of GCE metrics (see Section 3.3.1) it can be easily seen that all of them are nonnegative. The maximal value of conductance is equal to 1 when $R_c = \emptyset$ and consequently this measure satisfies both the normalization property and the maximum value property. When R_c is maximal conductance is not necessarily equal to zero. Conductance is equal to zero if and only if a module does not depend on other modules. Adding intra-cluster relationships increases only the denominator of conductance and consequently conductance satisfies the monotonicity property. The merge property of conductance is the consequence of Lemma 1 when P is the number of inter-cluster links and Q the sum of the number of inter- and intra-cluster links. The number of intra-cluster links is an additive property under the merge operation. Secondly, if two modules are unrelated then they have disjoint sets of inter-cluster links. This means that the number of inter-cluster links is also an additive property for unrelated modules.

In contrast to conductance, expansion does not satisfy the normalization property. If we modify expansion to be a value in the interval [0,1] then we actually obtain the cut-ratio metric. Expansion also does not satisfy the null value property and the maximum value property: both the numerator and denominator in the definition of expansion are independent of the number of intra-cluster links.

The expansion of a module remains the same under the addition of intra-cluster links. Therefore, this metric also satisfies the monotonicity property. As already mentioned, the number of inter-cluster links is an additive property for disjoint modules. The number of nodes in a module is also an additive property under the merge operation. Therefore, by Lemma 1 expansion satisfies the merge property.

Cut-ratio satisfies the normalization property: the maximal value of cut-ratio is equal to 1 which is obtained when each entity from the module references all entities defined outside the module. Both the numerator and the denominator of cut-ratio are independent on the number of intra-cluster links and similarly as expansion this measure does not satisfy the null and the maximum value property. If we add a new intra-cluster link the cut-ratio does not change and consequently this metric satisfies monotonicity property. The following lemma shows that the cut-ratio metric satisfied the merge property.

Lemma 2. Cut-ratio satisfies the merge property.

Proof. Let C_x denote the number of inter-cluster links emanating from nodes contained in module x, N_x the number of nodes in module x, and N the number of nodes in the whole network. Let p and q be two disconnected modules such that the cut-ratio of p is smaller than the cut-ratio of q, i.e.

$$\frac{C_p}{N_p(N-N_p)} \leq \frac{C_q}{N_q(N-N_q)} \quad \Leftrightarrow \quad C_pN_q(N-N_q) \leq C_qN_p(N-N_p). \tag{3.7}$$

Let r denote the union of p and q. Let us suppose that the merge property is not satisfied, i.e.

$$\frac{C_p + C_q}{(N_p+N_q)(N-N_p-N_q)} < \frac{C_p}{N_p(N-N_p)} \tag{3.8}$$

If we multiply both sides of inequality 3.8 by $(N_p+N_q)(N-N_p-N_q)N_p(N-N_p) > 0$, then we obtain

$$\begin{aligned} (C_p+C_q)N_p(N-N_p) &< C_p(N_p+N_q)(N-N_p-N_q) && (3.9)\\ C_qN_p(N-N_p) &< C_pN_q(N-N_q) - 2C_pN_pN_q && (3.10)\\ &\leq C_pN_q(N-N_q) && (3.11) \end{aligned}$$

which is in contradiction with inequality 3.7. □

From the definitions of ODF measures it can be easily seen that they take values in the range [0, 1], which means that they satisfy nonnegativity and normalization properties. When $R_c = \emptyset$ then Maximum-ODF and Average-ODF are equal to 1, while Flake-ODF is equal to 0, which means that Maximum- and Average-ODF satisfy the maximum value property, while Flake-ODF satisfies the null value property (recall that Flake-ODF measures cohesion, while Maximum- and Average-ODF are lack of cohesion metrics). The numerator of Maximum- and Average-ODF is independent of the number of intra-cluster links. Consequently, those metrics do not satisfy the null value property. Addition of new intra-cluster links decreases the out-degree fraction of nodes from which added links emanate. Thus, an extension of a module with new intra-module relationships (1) lowers the average-ODF, and (2) cannot increase the Maximum-ODF. Consequently, Average-ODF and Maximum-ODF satisfy the monotonicity property. The merge property is trivially satisfied for Maximum-ODF.

Lemma 3. Average-ODF satisfies the merge property.

Proof. Let D'_a denote the number of inter-cluster links emanating from node a, D_a out-degree of node a ($D'_a \leq Da$), and N_x the number of nodes in module x. Let p and q be two disconnected modules

such that the Average-ODF of p is smaller than the Average-ODF of q, i.e.

$$\frac{1}{N_p}\sum_{u\in p}\frac{D'_u}{D_u} \leq \frac{1}{N_q}\sum_{u\in q}\frac{D'_u}{D_u} \quad \Leftrightarrow \quad N_q\alpha \leq N_p\beta, \quad \text{where} \quad \alpha = \sum_{u\in p}\frac{D'_u}{D_u},\ \beta = \sum_{u\in q}\frac{D'_u}{D_u} \tag{3.12}$$

Let r denote the union of p and q. The Average-ODF of r is equal to

$$\text{Average-ODF}(r) = \frac{1}{N_p+N_q}\sum_{u\in r}\frac{D'_u}{D_u} = \frac{1}{N_p+N_q}\left(\sum_{u\in p}\frac{D'_u}{D_u} + \sum_{u\in q}\frac{D'_u}{D_u}\right) = \frac{\alpha+\beta}{N_p+N_q} \tag{3.13}$$

Let us suppose that Average-ODF does not satisfy the merge property, i.e. $(\alpha+\beta)/(N_p+N_q) < \alpha/N_p$. Then we obtain that $\beta N_p < \alpha N_q$ which is in contradiction with inequality 3.12. □

The addition of new intra-cluster links can only increase the number of entities defined in a module whose intra-cluster out-degree is greater than inter-cluster out-degree. Therefore, Flake-ODF satisfies the monotonicity property. The number of entities in the module whose intra-cluster out-degree is greater than inter-cluster out-degree is an additive property under the merge operation. Therefore, Flake-ODF also satisfies the merge property by Lemma 1.

TABLE 3.1: Properties of graph clustering metrics as (lack of) cohesion software metrics.

Metric	Nonnegativity	Normalization	Null value	Maximum value	Monotonicity	Merge
Conductance	yes	yes	no	yes	yes	yes
Expansion	yes	no	no	no	yes	yes
Cut-ratio	yes	yes	no	no	yes	yes
Maximum-ODF	yes	yes	no	yes	yes	yes
Average-ODF	yes	yes	no	yes	yes	yes
Flake-ODF	yes	yes	yes	no	yes	yes

The properties of graph clustering metrics as (lack of) cohesion software metrics are summarized in Table 3.1. This table indicates the limitations of GCE metrics as software metrics. As observed by Briand et al. [1998], only a few widely known software cohesion metric fulfill all of the cohesion properties. In other words, a measure which does not satisfy all of the properties can be considered as poorly defined. Secondly, we can see that GCE metrics reflecting lack of cohesion does not satisfy the null value property, while GCE metrics reflecting cohesion does not satisfy the maximum value property. However, we believe that this is not the disadvantage of GCE metrics. Firstly, it is very unlikely to observe fully connected software modules in practice (each class from a package reference each other; each method from a class calls each other and access to each class attribute). Secondly, in such cases GCE metrics favorite loosely coupled software modules emphasizing the principle of low coupling.

3.4 Extraction of software networks

Extraction of software networks assumes identification of architectural entities of a software system (packages, classes, functions, global variables) as well as identification of dependencies among them. There is a variety of software networks extractors. However, they are usually tied to a particular programming language and extract just one type of software network. For example, the review of static call graphs extractors for C programming language can be found in [Murphy et al., 1998], while [Telea et al., 2009] summarizes existing C++ static call graph extractors. Language-dependent

software networks extractors rely either on traditional parsing techniques or employ more lightweight, but less precise approaches based on lexical analysis [Kienle and Müller, 2010]. Syntactical based software networks extractors can be built either from scratch or using parser generators such as Yacc, ANTLR, etc. On the other side, lexical based software networks extractor perform pattern matching based on regular expressions to collect dependencies among software entities [Murphy and Notkin, 1996].

From the definition of package collaboration networks it can be seen that package-level dependencies are entirely induced from class-level dependencies. In other words, package collaboration networks can be trivially obtained when class collaboration networks are formed. There are two types of class-level dependencies: direct and induced. Direct dependencies between class-level entities represent dependencies explicitly stated in source code. Other class-level dependencies are induced from function calls and usage of variables defined outside a class. For example, Java class `Simple` defined below depends on classes/interfaces `HashMap`, `Iterator`, `Entry`, `Integer` and `String` which are explicitly mentioned as types in the definition of the class. However, this class also depends on interface `Set` which is the return type of method `entrySet()`.

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map.Entry;

public class Simple {
  public Simple(HashMap<String, Integer> hm) {
    Iterator<Entry<Integer, String>> it = hm.entrySet().iterator();
    while (it.hasNext()) {
      it.next();
    }
  }
}
```

In programming languages that support dynamic dispatch of function calls or function pointers one call site can refer to multiple targets, i.e. a function call can correspond to more than one function definition. In an ideal case the run-time type of a function call receiver (object/variable on which the function is called) has to be determined with respect to any execution of the program which is a clearly an undecidable problem. Therefore, we can distinguish between three types of static call graphs [Murphy et al., 1998]:

- Conservative. In conservative static call graphs the call from function A to function B can be omitted if and only if it can never occur in any execution of the program.
- Optimistic. Optimistic static call graphs do not contain non-existent (false positive) call links, but missing (false negative) call links may occur.
- Approximate. They can contain both false negative and false positive call links.

Clearly, a static call graph in which each node is connected to each other is conservative. Similarly, a static call graph in which each node is isolated can be considered as optimistic. However, such call graphs are hardly useful for any purpose. A name-based resolution of function calls is the most

simple approach to the construction of useful conservative and optimistic call graphs. This approach takes into account only names of functions and number of arguments. Let A be a function that contains function call site $v.f(a_1, a_2, ..., a_n)$ and let V denotes the declared type of variable v. Let D be the subset of functions declared in a software system whose names are f and which have exactly n arguments. In the conservative static call graph A is connected to each function from D. If the cardinality of D is equal to one then the call link is created in the optimistic static call graph. Class hierarchy analysis (CHA) [Dean et al., 1995] is the first improvement of the name-based resolution algorithm. In this approach D is formed considering only V and classes derived from V. Rapid type analysis (RTA) [Bacon and Sweeney, 1996] improves CHA by taking into account actual instantiation of classes. In this approach D is formed considering only V and subclasses of V that are instantiated in A or classes directly or indirectly coupled to A. The main characteristic of CHA and RTA is that those techniques are flow-insensitive, i.e. they do not maintain and keep information per statement and neither perform control and data flow analysis. Further improvements of CHA and RTA based on control-flow graphs and inter-procedural analysis are also proposed in the literature [Grove and Chambers, 2001; Grove et al., 1997].

In software networks extractors that build software networks at different levels of abstraction static call graphs are used to directly induce "hidden" class-level dependencies and indirectly "hidden" package-level dependencies. Therefore, it the extractor forms conservative static call graphs then it can also create false positive, non-existent class/package dependency links. On the other side, if it extracts optimistic static call graphs then some class/package dependency links may be missing. In other words, software networks representing systems written in programming languages that support dynamic dispatch of function calls are necessarily approximate.

3.4.1 Extraction of software networks for statistical analysis

In research works that deal with the analysis of software systems under the framework of complex network theory software networks are usually extracted using language-specific tools [Baxter et al., 2006; de Moura et al., 2003; Jenkins and Kirk, 2007; Louridas et al., 2008; Puppin and Silvestri, 2006; Sudeikat and Renz, 2007; Taube-Schock et al., 2011; Valverde and Solé, 2007; Wang et al., 2013; Wheeldon and Counsell, 2003]. Software networks associated to Java software systems are usually extracted from Java bytecode [Baxter et al., 2006; Jenkins and Kirk, 2007; Louridas et al., 2008; Sudeikat and Renz, 2007]. Usage of a language-specific software networks extraction tool naturally restricts statistical study to software systems written in a particular language. However, the authors of [Myers, 2003] and [Hylland-Wood et al., 2006] used basically the same extraction methodology based on Doxygen[1] to form software networks associated to software systems written in different programming languages. Doxygen was also used as dependency extractor in empirical investigations of architecture and reusability of open-source software systems [Capiluppi and Boldyreff, 2008; Capiluppi and Knowles, 2009; Capiluppi et al., 2011], as well as in research works dealing with the prediction of vulnerable software components [Nguyen and Tran, 2010] and software validation [Berner et al., 2005]. Doxygen is a documentation generator tool that supports more than ten programming languages. This tool can be configured to extract local class collaboration graphs that show inheritance and aggregation dependencies for individual classes. The extraction of local class collaboration graphs in Doxygen for programs written in C, C++, C#, Objective-C, Java, JavaScript, D, PHP and IDL is based on the unified fuzzy parsing approach. This means that there is one light-weight parser for all mentioned

[1] http://www.stack.nl/~dimitri/doxygen/

languages realized as a big state machine generated by the Flex lexical analyzer generator. For other supported languages (Tcl, Python and Fortran) Doxygen has independent lightweight parsers and each of them realizes language specific extraction of local class collaboration graphs.

3.4.2 Software networks extraction in reverse engineering tools and environments

In this Section we will review how fact bases (source code models in terms of software networks) are formed in widely used language-independent reverse engineering tools, environments, and frameworks.

Rigi [Kienle and Müller, 2010] is a reverse engineering environment that allows the visual exploration of software systems in the form of graphs that shows software entities and their relationships. It offers the language-independent exchange format based on a graph-based data model, fact extractors for C, C++, and COBOL, and an interactive graph editor called Rigiedit. Rigi's graph-based data model is capable to represent architectural elements of software systems: program components (functions, global variables, etc.) and their relationships (calls to function, references to variables, etc.). Rigi's architecture decouples fact extractors from the graph editor via the exchange format. Rigi's fact extractors for C and COBOL are parsers built with the help of the Yacc parser generator. Those parsers identify software entities and their dependencies in a source code and store extracted information in the textual exchange format known as RSF (Rigi Standard Format). Therefore, Rigi is capable to analyze and visualize networks representing software systems written in different in programming languages, but their extraction is not language independent since for each supported language there is a separate fact extractor.

Moose [Ducasse et al., 2000] is an environment for reverse engineering and re-engineering of object-oriented software systems. It consists of a repository to store language-independent models of software systems, and provides query and navigation facilities. Moose models are instances of the FAMIX meta-model and capture architectural elements of software systems: defined entities (classes, methods, attributes, etc.) and their mutual dependencies (inheritance, invocation, access and reference). In other words, Moose operates on software networks and it is capable to visualize them in various forms. There are two ways to form Moose models. In the case of Smalltalk fact extraction is performed via built-in parser. For other languages, Moose provide an import interface for CDIF and XMI files. Over this interface Moose uses external parsers for languages other than Smalltalk. Therefore, Moose, similarly to Rigi, does not support language-independent fact extraction: each parser independently recognizes entities and relationships in order to instantiate the FAMIX meta-model with concrete information about a software system.

Gupro [Ebert et al., 2002] is an integrated workbench to support program understanding of heterogeneous software systems on different levels of granularity. The extraction of information is done by parsers generated using the PDL parser generator. PDL extends the Yacc parser generator by the EBNF syntax and notational support for compiling textual languages into TGraphs. TGraphs are directed graphs whose nodes and edges may be attributed, typed and ordered. Those graphs are used to conceptually represent software systems: software entities are represented by nodes, relationships among entities by edges, a common type is assigned to similar objects and relationships, and ordering of relationships is expressed by edge order. TGraphs are produced by individual PDL parsers and consequently the fact extraction in Gupro is not language-independent.

Bauhaus [Raza et al., 2006] is a tool suite that supports program understanding and reverse engineering on all layers of abstraction, from the source code to the architecture. It is capable to analyze

programs in Ada, C, C++ and Java. In Bauhaus two separate program representations exist: Inter-Mediate Language (IML) and Resource Flow Graph (RFG). The IML representation is defined by the hierarchy of predefined classes, where each class represents a certain universal programming language construct. IML is generated from the source code by a language-specific front-end. While IML represents the system on a very concrete and detailed level, the architectural aspects of the system are modeled by means of RFG. An RFG is a hierarchical graph that consists of typed nodes and edges. Nodes represent architecturally relevant elements of software systems (routines, types, files, components, etc.). Different aspects of the architecture (call graph, hierarchy of modules, etc.) can be obtained using different granularity views. In other words, RFG is, similarly to GDN, a union of software networks at different levels of abstraction. For C and C++, an RFG is automatically generated from the IML representation, whereas for other languages RFG is generated from other intermediate representations (such as Java class files) or compiler supported interfaces (such as Ada Semantic Interface Specification). Therefore, fact extraction in Bauhaus is not fully language-independent, since RFGs for some languages are not formed directly from IML.

It should be also mentioned that there are language-dependent reverse engineering tools which realize software network extraction procedures that can be generalized to a variety of languages. For example, MetricAttitude [Risi and Scanniello, 2012], a tool for the visualization of Java software, relies on improved class hierarchy analysis [Dean et al., 1995] to approximate the run-time types of function call receivers. The static analysis approach proposed in [Risi and Scanniello, 2012] distinguishes between two types of the function call relationship: virtual and abstract delegations. The authors of MetricAttitude observed that such differentiation can be exploited to identify design patterns in the source code. Similarly, the Soot framework for Java byte-code optimization uses variable-type analysis (VTA) and declared-type analysis (DTA) to extract static call graphs from Java byte code [Sundaresan et al., 2000]. Both of these analysis can be thought as more refined versions of rapid type analysis (RTA) [Bacon and Sweeney, 1996]. Whereas RTA simply collects instantiated types in order to approximate run-time types of function call receivers, DTA and VTA find which types reach each variable (i.e. which allocated objects might be assigned to a variable) using information contained in so called *type propagation graphs*.

3.4.3 SNEIPL - a novel language-independent approach to the extraction of software networks

In this thesis, as one of the original contributions of the dissertation, we present SNEIPL, a novel language-independent approach to the extraction of software networks [Savić et al., 2012, 2014]. The main characteristic of SNEIPL is that it uses the enriched Concrete Syntax Tree (eCST) representation [Budimac et al., 2012; Rakić and Budimac, 2011a] of the source code to form software networks. eCST is the language-independent source code representation, and consequently makes SNEIPL independent of programming language. Therefore, the main contribution of SNEIPL is that it enables language-independent analysis of software systems under the framework of complex network theory, language-independent computation of software design metrics, and language-independent extraction and representation of fact bases for reverse engineering activities. The applicability of the approach is demonstrated by the extraction of software networks representing real-world, medium to large software systems written in different languages which belong to different programming paradigms. To investigate the completeness and correctness of the approach, class collaboration networks (CCNs) extracted from real-world Java software systems are compared to CCNs obtained by two other tools.

We used Dependency Finder which extracts class-level dependencies from Java bytecode, and Doxygen which realizes language-independent fuzzy parsing approach to dependency extraction.

3.4.3.1 eCST representation of source code

The development of the eCST representation started with SMIILE [Rakić and Budimac, 2011b], a language-independent tool for computing software metrics that reflect the internal complexity of software entities (metrics such as LOC, Cyclomatic complexity, Halstead complexity metrics, etc.). In [Rakić and Budimac, 2011a] the authors of the eCST representation identified other fields of research where the eCST representation can be utilized to construct language-independent tools which solve particular language processing problems. This research also led to the constitution of the SSQSA framework [Budimac et al., 2012], a set of language-independent tools that operate on the eCST representation produced by the SSQSA front-end known as eCST Generator.

eCST is a tree representation of the source code. In this Section of the thesis we will explain the principal differences between eCST and two other widely used tree representations of source code: concrete syntax tree (CST) and abstract syntax tree (AST).

The concrete syntax tree (CST) representation shows how a programming language construct is derived according to the context-free grammar of the language. The root node of a CST represents starting non-terminal symbol of the grammar, interior nodes in CST correspond to syntactical categories of the language identified by non-terminal symbols of the grammar, while leaf nodes represent tokens of the construct. Abstract syntax tree (AST) is an alternative and more compact way to represent language constructs. The AST representation retains the hierarchical structure of language constructs, while omitting details that are either visible from the structure of AST or unimportant for a language processing task.

Figure 3.3 shows the CST, AST and eCST representations of a simple Java source code fragment ("class A extends B { }"). All tokens present in the fragment are leaf nodes of the CST and eCST. The tokens representing keywords ("class" and "extends") appear as interior nodes in the AST. Separator tokens are not present in the AST since grouping parentheses are implicit in the tree structure. All interior nodes in the CST are non-terminal symbols from the Java grammar, while the interior nodes in the eCST are different eCST universal nodes. CST is the language-dependent source code representation, since it is closely connected to the grammar of a language. On the contrary, AST abstracts away from the concrete syntax. However, interior nodes of ASTs are keywords and operators of the language, or imaginary tokens introduced to enable tree representation of constructs that can not be adopted to the "operator-operands" scheme. The usage of lexical elements of the language as interior nodes in the intermediate representation makes the representation language-dependent.

The concept of universal nodes introduced in the enriched Concrete Syntax tree (eCST) representation is what makes it substantially different from the AST and CST representations. Universal nodes contained in eCSTs, such as CONCRETE_UNIT_DECL (CUD) in Figure 3.3, are language-independent markers of semantic concepts expressed by language constructs. One universal node denotes particular semantic concept realized by the syntax construction embedded into the eCST sub-tree rooted at the universal node. For example, the CUD universal node in Figure 3.3 denotes that the sub-tree rooted at the CUD contains the definition of a concrete class-level entity. Nodes of eCST can be divided into three categories:

- Universal nodes with predefined, language-independent meanings which denote semantic concepts expressed by language constructs.

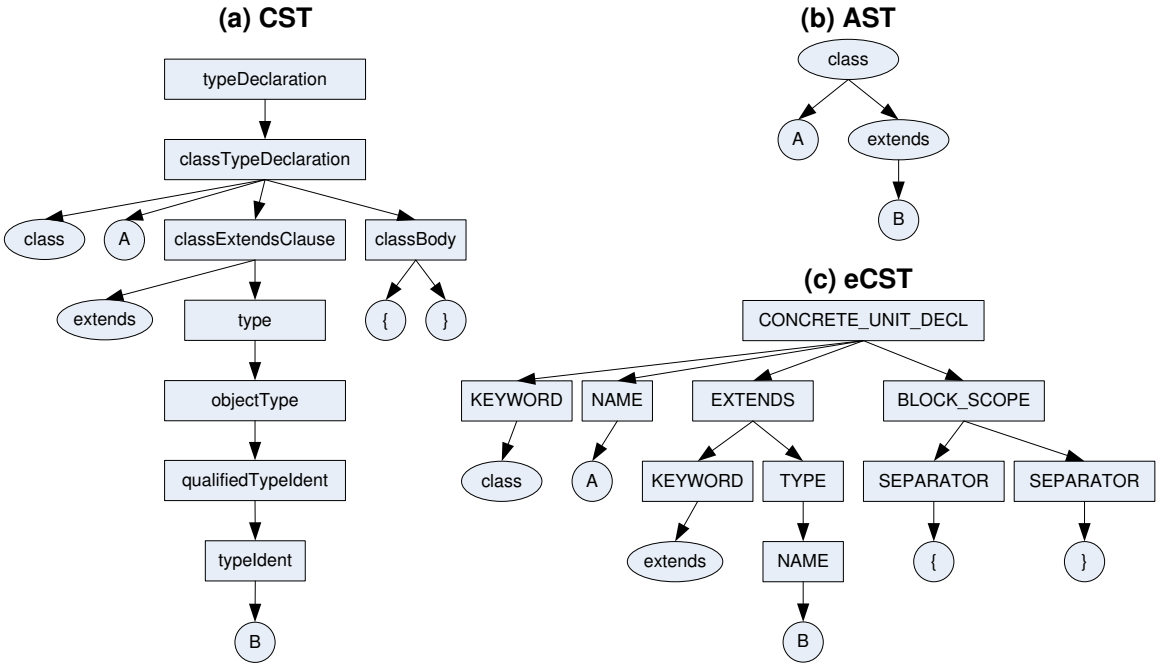


FIGURE 3.3: Concrete syntax tree (a), abstract syntax tree (b), and enriched concrete syntax tree (c) representing Java fragment "class A extends B { }".

- Imaginary nodes with language-dependent meanings which correspond to a subset of non-terminal symbols in the grammar. Those nodes serve only to retain natural hierarchical structure of language constructs in the case that there is no universal node that correspond to some non-terminal symbol.

- Tokens that are leaf nodes of eCSTs.

An eCST is usually more compact than the corresponding CST: one universal or imaginary node can substitute a chain of non-terminal symbols in the CST that is derived through a sequence of unary productions. As it can be observed from Figure 3.3 the TYPE universal node substituted the chain of three unary productions (type → objectType → qualifiedTypeIdent). On the other hand, the eCST is more voluminous than the corresponding AST, because the eCST includes all tokens present in the source code, while imaginary tokens in the AST are either universal or imaginary nodes in the eCST.

Each eCST universal node expresses some general concept of high-level programming languages. The set of universal nodes and the dependency constraints among them are determined by the problems solved by existing SSQSA back-ends, not by the syntactical structures of supported languages. When a new language processing problem is stated, the schema of eCST universal nodes is explored in order to determine if it can support the development of a new SSQSA back-end which solves the problem. This analysis may result in the introduction of new universal nodes in the schema. The support for a new programming language is achieved through the alignment of the schema with the grammar of the language. In this process each eCST universal node is mapped to one or more syntactical categories of the language that are represented by non-terminal symbols of the grammar.

3.4.3.2 eCST universal nodes used by SNEIPL

Currently the set of eCST universal nodes contains 33 different nodes divided into three groups:

- Lexical-level eCST universal nodes mark individual tokens with appropriate lexical category (keywords, separators, identifiers, etc.).
- Statement-level eCST universal nodes mark individual statements, groups of statements or parts of statements with appropriate concept expressed by them (jump statement, loop statement, branch statement, condition, import statement, block scope, etc.)
- Entity-level eCST universal nodes mark definitions and declarations of package, class and function level entities, and explicitly stated relations between them (such as inheritance, instantiation, implementation, etc.).

SNEIPL naturally relies on the entity-level eCST universal nodes to extract software networks. Table 3.2 shows the list of all eCST universal nodes used by SNEIPL. These universal nodes, except FUNCTION_CALL universal node, where introduced before SNEIPL was designed, implemented, and included in the SSQSA framework, and used by other SSQSA back-ends (SMIILE and SSCA).

TABLE 3.2: List of eCST universal nodes used to extract software networks.

Universal node	**Abbr.**	**Marks**
COMPILATION_UNIT	CU	Root of each eCST
PACKAGE_DECL	PD	Declaration of packages, namespaces and units
CONCRETE_UNIT_DECL	CUD	Declaration of classes and implementation modules
INTERFACE_UNIT_DECL	IUD	Declaration of interfaces and definition modules
TYPE_DECL	TD	User-defined data types that are not CUDs and IUDs
ATTRIBUTE_DECL	AD	Declaration of class attributes, class fields, global variables
FUNCTION_DECL	FD	Declaration of functions, procedures, methods
FORMAL_PARAM_LIST	FPL	List of parameters in FD definition
PARAMETER_DECL	PAR	One parameter in FPL
VAR_DECL	VD	Declaration of local variables in FD
IMPORT_DECL	ID	Import statements
BLOCK_SCOPE	BS	Block scope within a FD or another BS
FUNCTION_CALL	FC	Function call statements
ARGUMENT_LIST	AL	List of parameters passed to FC
ARGUMENT	ARG	One argument in AL
EXTENDS	EXT	CUD/IUD inheritance
IMPLEMENTS	IMP	IUD implementation
INSTANTIATES	INST	instantiation of objects
TYPE	TYPE	identifiers representing types
NAME	NAME	identifiers

Figure 3.4 shows a part of the eCST representation for two structurally equivalent code fragments written in Modula-2 and Java, respectively. The definition of class/implementation module A is marked with the CUD universal node. Entity A contains the definition of global variable/class attribute gv whose type is T, and the definition of procedure/method p. Therefore, the definitions of both mentioned entities are located in the sub-tree rooted at CUD A, and marked with appropriate eCST universal nodes (AD for gv and FD for p, see Table 3.2). Each identifier is marked with the NAME universal node. The parent of the NAME universal node determines what the identifier actually represents.

As already mentioned, SNEIPL is one of the back-ends present in the SSQSA framework. The SSQSA front-end (eCST Generator) produces the eCST representation for a given source code [Budimac et al., 2012]. Therefore, the set of programming languages supported by eCST Generator entirely determines the set of programming languages supported by SNEIPL and other SSQSA back-ends.

Modula-2:

```
IMPLEMENTATION MODULE A;
  VAR
     gv: T;

  PROCEDURE p(fp: T): RT;
  BEGIN
   …
  END p;
END A.
```

Java:

```
class A
{
  T gv;

  RT p(T fp)
  {
    …
  }
}
```

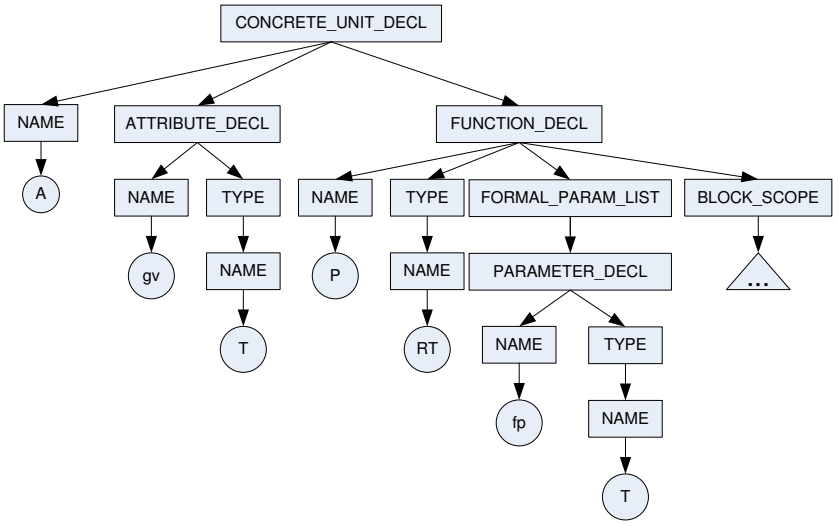


FIGURE 3.4: Two code fragments in Modula-2 and Java with the same structure of eCST universal nodes in the eCST representation.

Based on the extension of an input compilation unit, eCST Generator recognizes programming language and instantiates appropriate parser which forms the eCST representation. eCST Generator uses parsers generated by the ANTLR parser generator [Parr and Quong, 1995]. When we want to extend SSQSA to support a new language, we have to make the ANTLR grammar for the language and use the ANTLR tree-rewrite syntax to specify how existing eCST universal nodes are embedded into produced syntax trees. In other words, the support for a new language is done in a purely declarative way. For example, Listing 1 shows how CONCRETE_UNIT_DECL universal node is incorporated in the grammar productions which describe declarations of Modula-2 implementation modules and Java classes. The extensibility of the SSQSA framework is in details discussed in [Kolek et al., 2013].

3.4.3.3 SNEIPL architecture

SNEIPL consists of two components: GDN Extractor and GDN Filter (see Figure 3.5). From a set of eCSTs produced by eCST Generator, GDN Extractor constructs the General Dependency Network (GDN) representation of a software system. GDN Filter, as the name of the component suggests, filters extracted GDN to form the output set of software networks.

GDN is incrementally built in two phases, where both phases analyze each eCST in the input set. Phase 1 recognizes declarations of software entities and creates GDN nodes. Vertical dependencies (CONTAINS links) are also created in Phase 1. This means that Phase 1 results in the hierarchy tree representation of analyzed software system. Phase 2 creates the rest of GDN links, which means that in this phase SNEIPL identifies horizontal dependencies. Analyzers in both phases traverse eCST trees

Listing 1. CONCRETE_UNIT_DECL universal node in ANTLR grammar rules describing the declaration of Modula-2 implementation modules and Java classes, respectively.

```
// excerpt from Modula-2 grammar
moduleDeclaration : 'IMPLEMENTATION'? 'MODULE' ident priority? ';'
                    importList* export* block ident
  -> ^(CONCRETE_UNIT_DECL
           ^(KEYWORD 'IMPLEMENTATION')? ^(KEYWORD 'MODULE') ^(NAME ident)
           priority? ^(SEPARATOR ';') importList* export* block
      );

// excerpt from Java grammar
classDeclaration : 'class' ident genTypes? extClause? impClause? classBody
  -> ^(CONCRETE_UNIT_DECL
         ^(KEYWORD 'class') ^(NAME ident) genTypes? extClause? impClause? classBody
      );
```

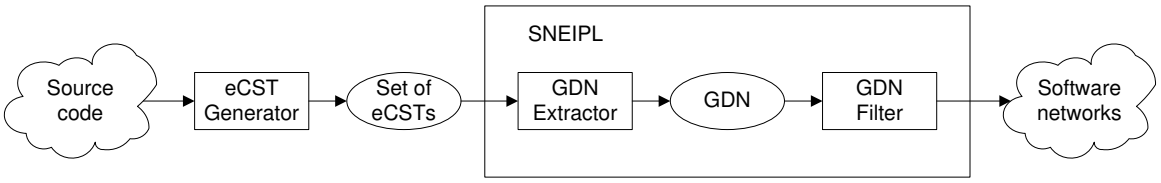


FIGURE 3.5: Data flow in software networks extraction process.

level by level realizing so called *trigger-deduce-action* mechanism. Triggers are some of eCST universal nodes, the aim of the deduce part of the mechanism is to determine the source and destination node for a link that will be created, while actions create GDN nodes and links or rewrite an eCST sub-tree with one node. Actions can be postponed, i.e. put on a stack to be executed after eCST traversal (see Algorithm "eCST analysis").

eCST analysis

input : t (eCST tree), *gdn* (General Dependency Network), *phase* (Phase of GDN extraction)

$q \leftarrow$ empty queue
$q.addLast(t.getRoot())$
$postponed \leftarrow$ empty stack

while not $q.empty()$ **do**
  $n \leftarrow q.removeFirst()$
  if n **is trigger in** *phase* **then**
    if n **has to be postponed in** *phase* **then**
      $postponed.push(n)$
    else
      deduce-action$(t, n, gdn, phase)$
  $children \leftarrow n.getChildren()$
  for $c \in children$ **do**
    $q.addLast(c)$
while not $postponed.empty()$ **do**
  $n \leftarrow postponed.pop()$
  deduce-action$(t, n, gdn, phase)$;

GDN Filter takes extracted GDN and executes a sequence of parameterized "Select NT Connected by LT" queries to isolate software networks. Parameters NT and LT specify node and link types, respectively. For example, query "Select {FD} connected by {CALLS}" forms a static call graph, while query "Select {CUD, IUD} connected by {REFERENCES}" isolates a class/module collaboration network. Table 3.3 shows the parametrization of queries that are executed by GDN Filter.

TABLE 3.3: Software networks extracted by SNEIPL and the parameterization of "select-connected by" queries.

Software network	**Select**	**Connected by**
Package collaboration network	PD	REFERENCES
Class collaboration network	CUD, IUD	REFERENCES
Static call graph	FD	CALLS
FUGV network	FD, AD	USES
Aggregation network	CUD, IUD	AGGREGATES
Weak aggregation network	CUD, IUD	WEAK_AGGREGATION
Inheritance network	CUD	EXTENDS
Bipartite network of implemented interfaces	CUD, IUD	IMPLEMENTS
Instantiate network	CUD	INSTANTIATES
Parameter type network	CUD, IUD	PARAMETER_TYPE
Return type network	CUD, IUD	RETURN_TYPE
Hierarchy network	PD, CUD, IUD, FD, AD, TD	CONTAINS

Query filter algorithm

input : GDN, lt (link type), nts (node types)
output: sn (software network)

```
/* execute "select-connected by" query */
sn ← empty
links ← gdn.linkSet()
for l ∈ links do
    if l of lt and l.src ∈ nts and l.dst ∈ nts then
        sn.updateWithNode(l.src)
        sn.updateWithNode(l.dst)
        sn.updateWithLink(l)
/* update software network with isolated nodes */
nodes ← gdn.nodeSet()
for n ∈ nodes do
    if n ∈ nts then
        sn.updateWithNode(n)
```

3.4.3.4 Phase 1 of GDN extraction

To recognize software entities SNEIPL relies on the following subset of universal nodes which are triggers for actions initiated in Phase 1 of GDN extraction:

$$U_{Phase_1} = \{\text{PD, CUD, IUD, FD, TD, AD}\}.$$

Each universal node from the U_{Phase_1} set has the NAME universal node in the sub-tree that contains the name of the software entity, while universal nodes themselves determine the type of newly created GDN nodes. Each GDN node is uniquely identified by a name that itself reflects the position of the

corresponding software entity in the hierarchy tree. For example, GDN node that corresponds to FD "F" (function "F") that is defined in the scope of CUD "C" (class "C") which belong to PD "P" (package "P") has name "P.C.F".

Vertical dependencies are induced from the structure of U_{Phase_1} nodes in eCST. Let a and b denote two software entities declared in the same compilation unit represented by eCST e. Let A and B ($A, B \in U_{Phase_1}$) denote universal nodes that mark declarations of a and b in e, respectively. Entity a is declared in the body of entity b, and connected by CONTAINS link $b \rightarrow a$ in GDN, if B is the first universal node from U_{Phase_1} found on the backwards path connecting A with the root of e (see Algorithm "Deduce-action in Phase 1").

Deduce-action in Phase 1

input : t (eCST tree), n ($n \in U_{Phase_1}$), gdn (General Dependency Network)

deduce:
$parentNode \leftarrow n.getParent()$
while $parentNode \neq t.getRoot()$ **and** $parentNode \notin U_{Phase_1}$ **do**
  $parentNode \leftarrow parentNode.getParent()$

if $parentNode \neq t.getRoot()$ **then**
  $fullyQualifiedName \leftarrow parentNode.getFullyQualifiedName() + "." + buildName(n \rightarrow$ NAME subtree$)$
else
  $fullyQualifiedName \leftarrow buildName(n \rightarrow NAME$NAME subtree$)$
$n.setFullyQualifiedName(fullyQualifiedName)$

action:
if gdn **does not contain node identified by** $fullyQualifiedName$ **then**
  $newNode \leftarrow gdn.createNode(name = fullyQualifiedName, type = n)$
  $n.setCorrespondingGDNNode(newNode)$
  if $parentNode \neq t.getRoot()$ **then**
    $scopeNode \leftarrow parentNode.getCorrespondingGDNNode()$
    $gdn.createLink(type = CONTAINS, src = scopeNode, dst = newNode)$

3.4.3.5 Phase 2 of GDN extraction

The extraction of horizontal dependencies is much harder task than the extraction of vertical, CONTAINS links. In order to deduce horizontal dependencies identifiers have to be matched with their definitions. SNEIPL realizes the name resolution algorithm based on information contained in import statements (marked with the IMPORT_DECL universal node), lexical scoping rules (BLOCK_SCOPE and universal nodes in U_{Phase_1} reflect different lexical scopes), and rapid type analysis that is adopted for the eCST representation. EXTENDS, IMPLEMENTS, and INSTANTIATES universal nodes in the eCST representation enable that rapid type analysis can be adopted for the eCST representation.

The extraction of horizontal dependencies is based on the following principles:

- Horizontal dependencies between class-level entities are determined before horizontal dependencies between other types of software entities. This principle enables rapid type analysis when resolving horizontal dependencies between function-level entities, because EXTENDS and IMPLEMENTS relations among class-level entities are already identified.

- Horizontal dependencies between function-level entities are resolved in the bottom-up manner: function calls that appear as arguments of other function calls are evaluated first. When a FUNCTION_CALL subtree is evaluated it is rewritten by a single node which contains the return type of called function.

- Horizontal dependencies between function-level entities can induce additional, "hidden" horizontal dependencies between class-level entities. Those are dependencies induced from function calls and statements in which a global variable from some other compilation unit is accessed. In both cases appropriate entities are referenced by fully qualified names, which means that they are not explicitly imported,

- Horizontal dependencies between package-level entities are directly induced from horizontal dependencies between class-level entities.

In order to match an identifier with its definition SNEIPL internally maintains two data structures: import list and symbol space. Import list is a list of GDN nodes that represent imported (visible) names declared outside an eCST that is currently processed. There is one import list per eCST (compilation unit). The import list is populated by the analysis of subtrees rooted at IMPORT_DECL universal nodes. Software entities declared in the scope of one PACKAGE_DECL are mutually visible without explicit import statements. Those entities are automatically added to the import list using the hierarchy tree formed in the first phase of GDN extraction.

An identifier marked with the TYPE universal node represents some class-level entity. The hierarchy tree extracted in Phase 1 is used to determine if the class-level entity corresponding to the type identifier is declared in the same eCST. If the type identifier is not declared in the currently processing eCST then the type identifier is matched against the import list in order to determine the corresponding GDN node (if exists). The TYPE universal node also indicates that there is a horizontal dependency between the GDN node corresponding to the first enclosing class-level universal node (CUD, IUD) and the GDN node corresponding to the type identifier. Thus, REFERENCES links between class-level entities are created by the analysis of eCST subtrees rooted at TYPE universal nodes. The parent of the TYPE universal node determines the form of coupling between two class-level entities. It is important to notice that class-level REFERENCES links are not established by connecting class-level entities declared in the compilation unit with all entities contained in the import list. This means that SNEIPL discards unused imports. Also, it is possible to have definitions of two or more class-level entities in one eCST (for example, two or more Java classes can be defined in one .java file). In other words, different class-level entities may share the same import list. The analysis of eCST subtrees rooted at TYPE universal nodes ensures that the REFERENCES link between class-level entity A and imported class-level entity B is created if and only if B is referenced in the body of A (see Algorithm "TYPE deduce-action"). Figure 3.6 illustrates the identification of class-level horizontal dependencies when two simple eCSTs are provided as the input.

SNEIPL attaches a local symbol table to each BLOCK_SCOPE and FUNCTION_DECL universal nodes in eCST. Local symbol table is the list of tuples (name, type) describing local variables declared in the scopes that are determined by the mentioned eCST universal nodes. They are created by the analysis of subtrees rooted at VAR_DECL and PARAMETER_DECL universal nodes. Variables contained in those subtrees are added to the local symbol table of the first enclosing BLOCK_SCOPE or FUNCTION_DECL universal node. Each identifier will be located either in some of local symbol tables or in the hierarchy tree extracted in Phase 1. Thus, local symbol tables together with the

TYPE deduce-action

input : t (eCST tree), TYPE (TYPE universal node in t), *gdn*

deduce:

/* determine source node for REFERENCES link: find first enclosing CUD or IUD */
enclosing ← TYPE.*getParent*()
while *enclosing* ≠ *CUD* **and** *enclosing* ≠ *IUD* **do**
  enclosing ← *parent.getParent*();
enclosingScope = *enclosing.getCorrespondingGDNNode*()

/* determine coupling type */
couplingType ← TYPE.*getParent*()

/* determine destination node for REFERENCES link */
refType ← *null*
refTypeName ← *buildName*(TYPE → NAME subtree)

/* first possibility: refTypeName is fully qualified name */
refType ← *gdn.findNode*(*refTypeName*)
if *refType* = *null* **then**
  /* second possibility: refTypeName declared in current compilation unit */
  e ← *enclosingScope*
  while *e.getType*() = *CUD* **or** *e.getType* = *IUD* **do**
    if *refTypeName matches a GDN node directly accessible from e via CONTAINS link* **then**
      refType found in current compilation unit, declared in entity *e*
      break;
    e ← GND node connected to *e* via CONTAINS link
  if *refType* = *null* **then**
    /* third possibility: refTypeName imported */
    refType = *t.importList.match*(*refTypeName*)

action:

if *refType* ≠ *null* **then**
  TYPE.setGDNNode(*refType*)
  gdn.createLink(*type* = REFERENCES, *srcNode* = *enclosingScope*, *dstNode* = *refType*)
  gdn.createLink(*type* = *couplingType*, *srcNode* = *enclosingScope*, *dstNode* = *refType*)
  srcPack ← *gdn.getPackageFor*(*enclosingScope*)
  dstPack ← *gdn.getPackageFor*(*enclosingScope*)
  gdn.createLink(*type* = REFERENCES, *srcNode* = *srcPack*, *dstNode* = *dstPack*)
else
  emit (*refTypeName* not declared in analyzed project)

set of GDN nodes formed in Phase 1 constitute the symbol space structure of the whole program. SNEIPL relies on basic lexical scoping rules when trying to match identifiers with their definitions. If an identifier is not found in local symbol tables the search is continued using GDN. Starting from the CUD that declares the last enclosing FD, the search is backwards propagated according to the EXTENDS and IMPLEMENTS GDN links.

NAME universal nodes trigger actions that create USES links. Those links appears in FUGV graphs which describe dependencies among FDs (functions) and ADs (global variables) used in FDs. NAME actions are executed only if NAME universal node is located under some FD universal node.

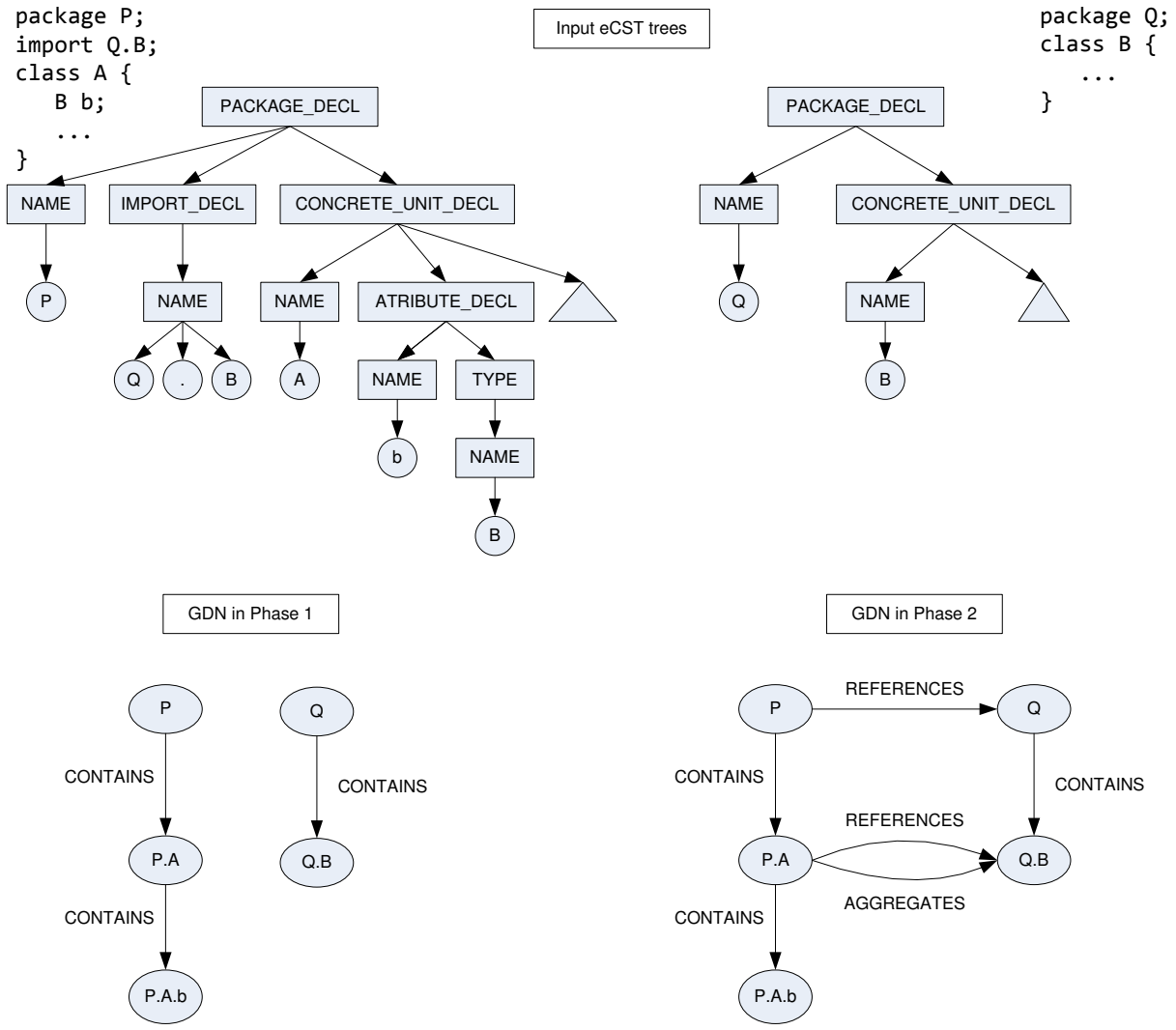


FIGURE 3.6: Two phases in GDN extraction: Phase 1 forms hierarchy tree while Phase 2 creates horizontal dependencies.

USES links can induce REFERENCES links between CUDs/IUDs and PDs. For example, if FD $P.A.f$ (function f from CUD A declared in PD P), accesses AD $Q.B.v$ (global variable v from CUD/IUD B declared in PD Q) whose type is $R.C$ (CUD/IUD C in package R), then REFERENCES links $P.A \rightarrow Q.B$ (direct dependency), $P.A \rightarrow R.C$ (hidden dependency), $P \rightarrow Q$ (direct dependency) and $P \rightarrow R$ (hidden dependency) will be induced (see Algorithm "NAME deduce-action").

The FUNCTION_CALL universal node has two children: the NAME universal node representing name of the called function and the ARGUMENT_LIST universal node representing arguments passed to the function call. Each argument in ARGUMENT_LIST is labeled by the ARGUMENT universal node. Actions triggered by FUNCTION_CALL universal nodes in Phase 2 form CALLS links, but also can induce REFERENCES links among CUDs/IUDs and PDs in the same way as actions triggered by NAME universal nodes induce REFERENCES links. FUNCTION_CALL deduce-actions are postponed (put on the stack to be executed after the whole eCST tree is traversed in Phase 2). Such mechanism ensures that function calls that appear as arguments of other function calls are evaluated first. The name of called function can be composite: it can consist of a variable (object) on which the function is called or of a fully qualified name of an entity which defines/declares the static function.

NAME deduce-action algorithm

input : t (eCST tree), NAME (NAME universal node in t), *gdn*, *SymbolSpace*

deduce:

$FD \leftarrow$ first enclosing FUNCTION_DECL with respect to NAME
if $FD = null$ **then**
  return
$srcFunction \leftarrow FD.getCorrespondingGDNNode()$
$name = buildName$(NAME subtree)
$searchRes = SymbolSpace.search(name)$
$dstVar \leftarrow null$
$dstVarType \leftarrow null$
if $searchRes = null$ **then**
  emit($name$ not declared in analyzed project)
else if $searchRes.scope = BS$ **or** $searchRes.scope = FD$ **then**
  emit($name$ local variable or argument of an function)
else
  $dstVar \leftarrow gdn.findNode(searchRes.name)$
  $dstVarType \leftarrow gdn.findNode(searchRes.type)$

action:

if $dstVar \neq null$ **then**
  /* create USES links */
  $gdn.createLink(type = USES, srcNode = srcFunction, dstNode = dstVar)$
  /* induce direct class- and package-level dependencies */
  $srcUnit \leftarrow gdn.getUnitFor(srcFunction)$
  $srcPack \leftarrow gnd.getPackageFor(srcUnit)$
  $dstUnit \leftarrow gdn.getUnitFor(dstVar)$
  $dstPack \leftarrow gnd.getPackageFor(dstPack)$
  $gdn.createLink(type = REFERENCES, srcNode = srcUnit, dstNode = dstUnit)$
  $gdn.createLink(type = REFERENCES, srcNode = srcPack, dstNode = dstPack)$
  /* induce hidden class- and package-level dependencies */
  if $dstVarType \neq null$ **then**
    $gdn.createLink(type = REFERENCES, srcNode = srcUnit, dstNode = dstVarType)$
    $typePack \rightarrow gdn.getPackageFor(dstVarType)$
    $gdn.createLink(type = REFERENCES, srcNode = srcPack, dstNode = typePack)$

In the case that the function is called on an object it is necessary to determine the type of the object. The symbol space is used to determine the type of the object, but also implicit and explicit castings have to be taken into account due to polymorphism and run-time binding of function calls.

Implicit castings are handled by rapid type analysis (RTA) which assumes that variable y of type Y can be assigned to variable x ($x := y$) of type X, where X is supertype of Y (then Y is subtype of X). Let t denote variable of type T in CUD U on which function f is called, i.e. class-level entity U in one of its functions contains the function call statement "$t.f(...)$". RTA searches for the definition of f in super and subtypes of T which are instantiated in U and all CUDs directly or indirectly coupled to U. Since RTA is flow-insensitive and does not keep per-statement information there can be multiple targets for f after the analysis. In such cases SNEIPL does not create CALLS links in order to prevent Type I errors (creation of non-existent or false positive CALLS links). Supertypes of T are all GDN nodes reachable from GDN node representing T via EXTENDS or IMPLEMENTS GDN

links. Consequently, if GDN node representing T is reachable via EXTENDS or IMPLEMENTS links from GDN node S then S is subtype of T. Another situation relevant to implicit castings occurs after the call to f is matched with the definition of f. Then U references all types present in the list of formal arguments in the definition of f, since arguments in the call of f may be implicitly casted to the types requested by the definition of f.

Every type identifier is marked with the TYPE eCST universal node. Therefore, the explicit cast of variable v to type T is represented by an eCST tree rooted at the NAME universal node which contains two children: (1) token representing the name of variable v, and (2) the TYPE sub-tree containing the name of type T. The explicit cast statement, due to the existence of the TYPE universal node, causes the creation of the explicit class-level dependency between the class-level entity containing the cast statement and the class-level entity representing type T. Additionally, TYPE information in the NAME sub-tree determines the type of variable v when the explicit cast is the part of a function call statement.

In programming languages that support function overloading it is possible that a function call can not be uniquely matched with the definition of called function using only the name and the number of arguments. In such cases SNEIPL tries to determine the type of each argument in order to select the appropriate definition from a set of candidates that are obtained by rapid type analysis. However, this process may result in unresolved types for arguments if an argument itself is the call to a function imported from a library, and consequently not present in analyzed eCSTs. In the case that the successfully resolved types of arguments do not contain enough information to choose the right candidate, CALLS link can not be created. This means that SNEIPL extracts *optimistic call graphs* where non-existent (false positive) CALLS links are not present, but missing (false negative) CALLS links can occur.

SNEIPL currently extracts only direct function calls, i.e. those calls where the name of the function is used to reference the function. Indirect function calls via function pointers or variables of procedural data types are not yet supported. Those features, when they exist in a language, are mostly extensively used in system programming, and have to be taken into account when extracting call graphs for the optimization tasks done by compilers. The aim of SNEIPL is to extract architectural graph representations of software systems that can be used for software engineering purposes. As pointed out in [Murphy et al., 1998], the requirements placed on tools that compute call graphs for software engineering purposes are typically more relaxed than for compilers, and those tools usually ignore rarely used language features which drastically increase the complexity of static code analysis.

3.4.3.6 Applicability of SNEIPL – controlled experiment

We designed and implemented two programs in Java and C# that administer typical students' activities. Programs are semantically and structurally equivalent which means that they have the same hierarchical structure of entities and realize the same functionality. Both programs consist of 25 methods defined in five classes (*Exam*, *Schedule*, *Mark*, *Person* and *Student*) and one interface (*IStudent*), all in package *JavaTest*/*CSharpTest*. This means that software networks, correctly extracted from these programs, have to be structurally equivalent (isomorphic) at the class and method level. At the package level we have a trivial case of isomorphism, one isolated node, because all classes and interfaces are contained in one package.

For both programs, SNEIPL correctly identified all defined software entities. In both cases, extracted GDN contains 33 nodes and 63 links in one weakly connected component. The distribution of

link types is also the same for both programs. Figure 3.7 shows extracted static call graphs and class collaboration networks visualized with Pajek. It can be observed that networks at different levels of abstraction are isomorphic as expected. This result confirms that eCST is a suitable representation for the language-independent extraction of software networks.

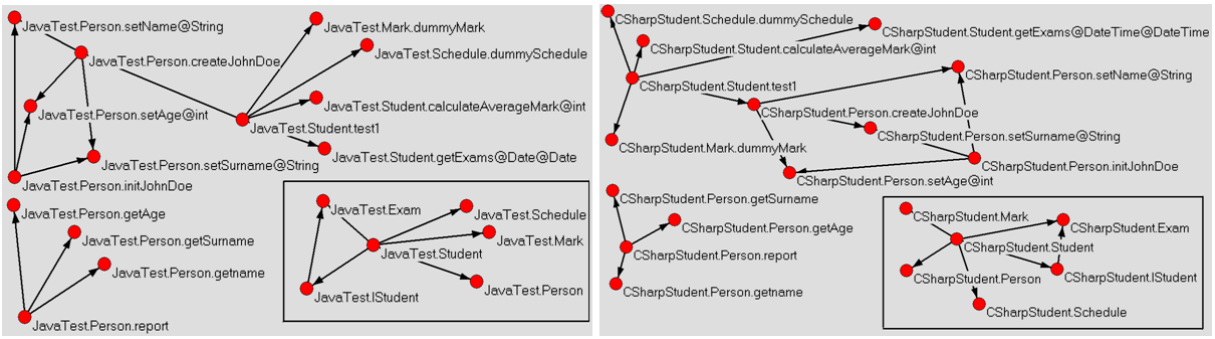


FIGURE 3.7: Extracted static call graphs and class collaboration networks for two identical program written in Java and C#.

3.4.3.7 Applicability of SNEIPL – extraction of software networks from real-world software systems

In this Section we will demonstrate that SNEIPL is able to identify dependencies in real-world software systems written in different programming languages which belong to different language paradigms. We employed SNEIPL to extract software networks from the following software projects:

- Commons-IO[2] (CIO), an open-source Java library of utilities to assist with developing IO functionality,
- Apache Tomcat[3], an open-source web server and Java servlet container written in Java,
- Modula-2 Algebra System[4] (MAS), an open-source computer algebra system written in Modula-2,
- Lumos[5], an open-source operating system written in Modula-2 for a computer called Stride 440,
- Model Scene Editor (MSE)[6], an open-source 3D scene editor written in Delphi,
- A proprietary, database-oriented Delphi application (we will use the term "DelPro" to denote this software).

Table 3.4 summarizes software systems used in the experiment, and for each of them shows the number of lines of code (LOC), the number of eCSTs produced by eCST Generator (this number is equal to the number of compilation units in the source code distribution), and the total number of eCST nodes in produced eCSTs.

Table 3.5 summarizes the properties of extracted General Dependency Networks. Links representing self references are excluded from the counts. The distribution of GDN nodes per type shows us how many nodes will appear in a particular software network.

[2]http://commons.apache.org/io/
[3]http://tomcat.apache.org/
[4]http://krum.rz.uni-mannheim.de/mas/
[5]http://www.uranus.ru/download/lumos.zip
[6]http://mse.sourceforge.net/

TABLE 3.4: The summary of software systems used in the extraction experiment.

Software system	**CIO**	**Tomcat**	**MAS**	**Lumos**	**DelPro**	**MSE**
Version	2.4	7.0.29	1.01	2	-	0.13
Language	Java	Java	Modula-2	Modula-2	Delphi	Delphi
LOC	25663	329924	100546	37250	104438	41858
#eCSTs	103	1083	329	297	491	113
#eCST nodes	88063	1650355	824043	297095	1151923	466061

TABLE 3.5: The number and distribution of nodes and links in extracted General Dependency Networks.

Software system	**CIO**	**Tomcat**	**MAS**	**Lumos**	**DelPro**	**MSE**
#nodes	1518	24287	6857	4104	13721	8359
PD - packages/units	6	97	0	0	491	113
CUD - classes/imp. modules	104	1351	163	193	501	156
IUD - interfaces/def. modules	4	143	166	115	0	22
TD - other user-defined types	0	21	66	293	43	1061
AD - class attrs./global vars.	328	7364	1128	1475	9846	4252
FD - methods/functions	1076	15311	5334	2028	2840	2755
#links	3001	71093	31558	12174	18267	11630
vertical dependencies	1517	24270	6528	3807	13230	8246
horizontal dependencies	1484	46823	25030	8367	5037	3384
package-level dependencies	9	493	0	0	895	268
class-level dependencies	341	13962	3024	1559	965	692
method-level dependencies	1134	32368	22006	6808	3177	2424
calls dependencies	611	20831	17456	3738	1522	772
access dependencies	523	11537	4450	3070	1655	1652

From extracted GDNs SNEIPL form the set of software networks at different levels of abstraction, thus providing different granularity views of the organizational structure of software systems under investigation. Vertical dependencies (CONTAINS links) reflect the hierarchy of software entities, and together with the set of GDN nodes constitute hierarchy tree view of analyzed source code. Characteristics of extracted hierarchy networks are shown in Table 3.6. IN_0 denotes the number of nodes whose in-degree is equal to zero. Those nodes represent software entities which are not contained in other entities. Hierarchy network is a disjoint union of hierarchy trees, and each zero in-degree node is the root of one hierarchy tree. In the case of Java software systems zero in-degree nodes are root packages, in Modula-2 systems they represent non-local (non-nested) modules, while in Delphi systems each unit is one zero in-degree node (Delphi units can not be nested). On the other hand, nodes having out-degree equals to zero (OUT_0) represent entities which do not define other entities. It can be seen that the majority of software entities are nodes with zero out-degree (from 92.49% in CIO to 95.22% in MAS). This is not surprising because global variables (class attributes) and functions (methods) that do not define nested function-level or named class-level entities are entities with zero out-degree in hierarchy networks.

Horizontal dependencies connect software entities appearing at the same level of abstraction. At the highest level of abstraction we have dependencies between package-level entities (packages in Java and units in Delphi). Table 3.7 shows characteristics of extracted package collaboration networks (PCNs) for Java and Delphi systems. Table 3.7 also contains information about packages with the maximal value of Robert Cecil Martin's afferent and efferent coupling metrics (MaxAC and MaxEC, respectively). For example, `Scene` is the most reused Delphi unit in MSE (referenced by 34 other

TABLE 3.6: Characteristics of extracted hierarchy networks: #nodes - the number of nodes, #links - the number of links, IN_0 - the number of nodes without in-coming links, OUT_0 - the number of nodes without out-going links, UPP - the average number of units per package, FPU - the average number of functions per unit, and VPU - the average number of global variables per unit.

Software system	**CIO**	**Tomcat**	**MAS**	**Lumos**	**DelPro**	**MSE**
#nodes	1518	24287	6857	4104	13721	8359
#links	1517	24270	6528	3807	13230	8246
IN_0	1	17	329	297	491	113
OUT_0	1404	22694	6529	3808	12756	7978
UPP	17.166	11.525	0	0	1.02	1.575
FPU	9.962	10.24	16.212	6.584	5.522	7.87
VPU	3.037	4.924	3.428	4.788	18.261	10.702

units), while unit `Main` has the highest degree of aggregation of other units (it references 25 other units).

TABLE 3.7: Characteristics of extracted package collaboration networks: #nodes - the number of nodes, #links - the number of links, #isol - the number of isolated nodes, MaxAC - the highest value of in-degree (afferent coupling), MaxEC - the highest value of out-degree (efferent coupling).

Software system	**CIO**	**Tomcat**	**DelPro**	**MSE**
#nodes	6	97	491	113
#links	9	493	895	268
#isol	0 (0%)	1 (1.03%)	21 (4.28%)	6 (5.31%)
MaxAC	5	58	169	34
MaxAC name	io	juli.logging	AmcCountrySP	Scene
MaxEC	2	30	144	25
MaxEC name	io.output	catalina.core	MainAmcBS	Main

At the middle level of abstraction there are dependencies between class-level entities: classes and interfaces in Java and Delphi, and definition and implementation modules in Modula-2. In class (module) collaboration networks all parallel links denoting different coupling types between two classes (modules) are reduced to one link, i.e. different coupling types between two nodes are recorded as attributes of one REFERENCES link. The characteristics of extracted class collaboration networks for software systems from the corpus are given in Table 3.8. The table also provides the information about the fraction of isolated nodes. It can be observed that for Tomcat, MAS and Lumos the fractions of isolated nodes are very low (less than 2% of the total number of classes/modules), suggesting that in those systems the number of unused ("dead") classes is reduced to the minimum. For other systems, isolated nodes do not necessarily point to unused code. In the case of libraries, isolated nodes can denote simple utility classes directly available to programmers. An example of such isolated class is `io.CopyUtils` from CIO. The mentioned class provides the set of static methods for copying files and relies only on JDK classes from `java.io` package. To the contrary, for standalone user applications, such as DelPro and MSE, it is more likely that isolated classes indicate unused or unfinished code. Examples of such classes in MSE are `TSplashScreen`, `TRegisterDialog` and `THelpIndexDialog`. The mentioned classes declare only Delphi visual components as class attributes without corresponding event handler methods, and their names clearly suggest that they represent non-core features planned to be introduced in one of the future releases.

At the lowest level of abstraction SNEIPL identifies function-level dependencies: CALLS links between functions and ACCESS links between functions and global variables. Table 3.9 presents the

TABLE 3.8: Characteristics of extracted class/module collaboration networks: #nodes - the number of nodes, #links - the number of links, #isol - the fraction of isolated nodes, MaxIn - class/module having the highest in-degree, MaxOut - class/module having the highest out-degree (the exact values of in- and out- degrees are given in brackets).

Software system	#nodes	#links	#isol (%)	MaxIn	MaxOut
CIO	108	174	15.74	AbstractFileFilter (19)	FileFilterUtils (16)
Tomcat	1494	6839	1.67	Log (293)	StandardContext (73)
MAS	329	2054	0.91	MASStor (277)	RqePRRC (36)
Lumos	308	973	1.94	R2SysCalls (78)	L2SysCalls (25)
DelPro	501	770	5.58	TAmcCountry (57)	TFMainAmcBS (145)
MSE	178	343	6.74	TShape (35)	TMainForm (57)

characteristics of extracted static call graphs for software systems from the corpus, while Table 3.10 shows the functions having the highest in- and out-degree.

TABLE 3.9: Characteristics of extracted static call graphs/method collaboration networks.

Software system	CIO	Tomcat	MAS	Lumos	DelPro	MSE
#nodes	1076	15311	5334	2028	2840	2755
#links	611	20831	17456	3738	1522	772
#isolated (%)	43.77	30.49	43.4	37.33	59.05	75.97
#calls resolved (%)	88.47	95.46	100	100	100	99.05
#hard to match (%)	30.67	7.48	0	0	0	0.11
hard to match resolved	132	832	-	-	-	0
hard to match unresolved	102	1494	-	-	-	9

TABLE 3.10: Functions with the maximal values of in- and out- degree in extracted static call graphs.

Software system	MaxIn		MaxOut	
CIO	Charsets.toCharset	24	DirectoryWalker.walk	7
Tomcat	Log.isDebugEnabled	429	StandardContext.startInternal	64
MAS	MASStor.ADV	1132	MASLoadE.InitExternalsE	128
Lumos	R2SysCalls.WriteString	121	L2SysCalls.InitPr	100
DelPro	AmcCountrySP.UpdateSQLWithSchema	257	TFActivityHandler.DoActivity	24
MSE	Misc.SaveStringToStream	16	TSceneData.MouseDown	12

In the case of Modula-2 programs nodes in a SCG represent function declarations in definition modules and function definitions in corresponding implementation modules. Since Modula-2 does not have function overloading and inheritance features, a direct Modula-2 function call is always matched with the definition of called function in the implementation module. In other words, isolated nodes in extracted Modula-2 SCGs represent either function declarations in definition modules, unused functions in implementation modules or functions in implementation modules called exclusively via variables of procedural types. The SCG nodes representing functions from definition modules can be easily pruned from the SCG (they are attached to CONTAINS links emanating from GDN nodes whose type is IUD eCST universal node), thus leaving only unused functions from implementation modules as isolated nodes in the SCG.

For object-oriented languages, due to function overloading and overriding, some function calls may be unresolved by SNEIPL (see Section 3.4.3.5). In such cases the rapid type analysis realized by SNEIPL's function call resolver results in multiple function definitions as destination candidates (targets) for a single function call. All candidates represent functions with the same name and the same number of formal parameters. We call such functions *hard to match*. Table 3.9 shows the fraction of hard to match functions, as well as the fraction of resolved function calls. A function

call is resolved when there is exactly one candidate in the candidate list for the destination function definition after rapid type analysis. Naturally, due to the absence of dynamic binding, there are no hard to match functions in Modula-2 systems, and each direct function call is properly resolved. The number of hard to match functions in DelPro is equal to zero, which means that all functions present in this software are unique up to name and the number of formal parameters. In other systems, due to the existence of hard to match functions, there are unresolved function calls. The upper bound of missing CALLS links in extracted SCGs is equal to the number of unresolved function calls: for CIO unresolved function calls create maximally 102 CALLS links, for Tomcat 1494, and for MSE maximally 9 CALLS links. Table 3.9 also provides information about the number of resolved calls to hard to match functions.

Missing calls links may cause isolated nodes in a SCG. When this is not the case an isolated node in SCG does not necessarily represent unused function. For example, Delphi methods from the user-interface layer in GUI applications are event handlers which are never explicitly called by other methods. Also, isolated nodes can represent methods that are dynamically invoked through reflection mechanisms of a language such as Java Reflection API. Another case is that those methods represent call-back methods used by the standard language library or third-party libraries.

3.4.3.8 Comparative analysis

In order to investigate the correctness and completeness of the dependency extraction procedure realized by SNEIPL, we extracted class collaboration networks representing ten real-world, open-source, and widely used software systems written in Java, and compared them to the class collaboration networks extracted by a language-dependent tool – Dependency Finder version 1.2.1, and a language-independent tool – Doxygen version 1.8.5. The characteristics of software systems used in the comparative analysis are summarized in Table 3.11.

TABLE 3.11: Java software systems used in the comparative analysis.

Software system	Version	LOC	Short description
CommonsIO	2.4	25663	IO library
Forrest	0.9	4683	Web publishing framework
PBeans	2.0.2	8502	Object/relational database mapping framework
Colt	1.2.0	84592	High performance scientific computing library
Lucene	3.6.0	111763	Text search engine library
Log4j	1.2.17	43898	Java logging library
Tomcat	7.0.29	329924	Web server and servlet container
Xerces	2.11.0	216902	XML parser library
Ant	1.9.2	219094	Build tool
JFreeChart	1.0.17	226623	Chart creator

Nodes in a class collaboration network are identified by fully qualified class names. Therefore it is easy to match two nodes from two different networks representing the same class (different in the sense that they are formed by two different tools). Consequently, a link in a class collaboration network is uniquely identified by the fully qualified names of the source and destination class.

Since a network consists of a set of nodes and a set of links, comparing two networks is equivalent to comparing two sets of nodes and two sets of links. The Jaccard coefficient (also Jaccard index or Jaccard similarity measure) is a commonly used measure of the similarity between two sets. In our comparative analysis we use two Jaccard coefficients, one expressing the similarity between two sets of CCNs nodes, and another showing the similarity between two sets of CCNs links. Let

$\mathrm{CCN}_A = (N_A, L_A)$ and $\mathrm{CCN}_B = (N_B, L_B)$ denote two class collaboration networks extracted by tools A and B, respectively. With N_x and L_x are denoted sets of nodes and links, respectively, in the CCN extracted by tool x ($x = A$ or $x = B$). We will also use the following symbols:

$\mathrm{MN}_{A,B}$ – the number of mutual nodes, i.e. nodes that appear in both CCN_A and CCN_B,
UN_A – the number of nodes that are unique to CCN_A and do not appear in CCN_B,
UN_B – the number of nodes that are unique to CCN_B and do not appear in CCN_A,
$\mathrm{ML}_{A,B}$ – the number of mutual links, i.e. links that appear in both CCN_A and CCN_B,
UL_A – the number of links unique to CCN_A and do not appear in CCN_B,
UL_B – the number of links unique to CCN_B and do not appear in CCN_A.

Using the previously introduced notation, the Jaccard coefficient for nodes can be defined as:

$$\mathrm{JN}(A,B) := \frac{\text{number of mutual nodes}}{\text{total number of different nodes}} = \frac{|N_A \cap N_B|}{|N_A \cup N_B|} = \frac{\mathrm{MN}_{A,B}}{\mathrm{MN}_{A,B} + \mathrm{UN}_A + \mathrm{UN}_B}.$$

Similarly, the Jaccard coefficient for links is:

$$\mathrm{JL}(A,B) := \frac{\text{number of mutual links}}{\text{total number of different links}} = \frac{|L_A \cap L_B|}{|L_A \cup L_B|} = \frac{\mathrm{ML}_{A,B}}{\mathrm{ML}_{A,B} + \mathrm{UL}_A + \mathrm{UL}_B}.$$

Let us assume that CCN_B is the 100% correct class collaboration network, i.e. CCN_B contains all classes and all class dependencies present in the corresponding software system. Then JN and JL quantify both the completeness and correctness of the node and link sets obtained by tool A. Namely, UN_B and UL_B represent the number of existent nodes and links respectively, that are not identified by tool A (missing nodes and links). Higher UN_B and UL_B imply lower degree of completeness of results obtained by tool A. On the other hand, UN_A and UL_A represent the number of non-existent nodes and links respectively, that are created by tool A. Therefore, higher UN_A and UL_A imply lower degree of correctness of results obtained by tool A. In our comparative analysis we examine three different dependency extraction approaches where one is language-independent (Dependency Finder). Therefore, under the assumption that the language-dependent tool produces 100% correct results, JN and JL quantify the completeness and correctness of the two other language-independent approaches.

Tables 3.12 and 3.13 summarize differences between class collaboration networks extracted using SNEIPL, Dependency Finder, and Doxygen. Both tables show the number of nodes ($|N_x|$) and links ($|L_x|$) in the CCN formed by tool x, the number of mutual nodes (MN) and mutual links (ML), the number of unique nodes (UN_x) and unique links (UL_x) with respect to tool x, and the values of the Jaccard coefficients for nodes (JN) and links (JL). It can be observed that the CCNs formed by SNEIPL are highly similar to those formed by Dependency Finder: for all analyzed systems, except for Tomcat, we have JN = 1.0 (identical sets of nodes), while JL is always higher than 0.9 implying highly overlapping sets of links (class dependencies). That is not the case with Doxygen where the maximal JL is equal to 0.41. The CCNs extracted by Doxygen are significantly smaller ($|L_B| \ll |L_A|$) proper sub-graphs ($\mathrm{UN}_B = 0 \wedge \mathrm{UL}_B = 0$) or close to proper sub-graphs ($\mathrm{UN}_B \ll \mathrm{UN}_A \wedge \mathrm{UL}_B \ll \mathrm{UL}_A$) of corresponding CCNs formed by Dependency Finder.

In the case of Tomcat all classes identified by Dependency Finder are also identified by SNEIPL ($\mathrm{UN}_B = 0$), but SNEIPL identified seven classes more ($\mathrm{UN}_A = 7$). The same seven classes are also present in the CCN extracted by Doxygen (see UN_B value in Table 3.13). The analysis of the Ant script that is used to build Tomcat revealed that those classes are not the part of the Tomcat binary

distribution, but belong to extra components (JMX Remote Lifecycle Listener and JSR 109 web services support).

From the data presented in Table 3.12, it can be observed that for all examined systems SNEIPL identified a small portion ($UL_A \ll ML$) of class dependency links which are not identified by Dependency Finder. For example, the Forrest CCN formed by DependecyFinder is a sub-graph of the Forrest CCN formed by SNEIPL ($UL_A = 4$, $UL_B = 0$), i.e. all classes and dependencies identified by Dependency Finder are also identified by SNEIPL, but SNEIPL identified 4 dependencies more. Those dependencies are represented by the following links:

locationmap.lm.AbstractNode	→ locationmap.lm.LocationMap
locationmap.RegexpLocationMapMatcher	→ locationmap.lm.LocationMap
locationmap.WildcardLocationMapMatcher	→ locationmap.lm.LocationMap
locationmap.WildcardLocationMapHintMatcher	→ locationmap.lm.LocationMap

Dependency Finder was unable to identify aforementioned dependencies simply because they do not exist in the bytecode. For a final String attribute, or a final attribute of some primitive type, the Java compiler inlines the value of the attribute directly into all client classes, so dependencies to the class which owns the attribute are lost. Another situation observed in our case studies when the translation from source to bytecode can lead to the loss of dependencies occurs when a dependency between two classes is caused solely by the existence of local variables whose type is the dependent class. Type information for local variables is not present in bytecode: the Java compiler validates assignments involving local variables and then discards information about their types.

TABLE 3.12: Similarity between class collaboration networks extracted by A = SNEIPL and B = Dependency Finder.

Software system	$\|N_A\|$	$\|N_B\|$	MN	UN_A	UN_B	**JN**	$\|L_A\|$	$\|L_B\|$	ML	UL_A	UL_B	**JL**
CommonsIO	108	108	108	0	0	**1.0**	174	174	173	1	1	**0.99**
Forrest	35	35	35	0	0	**1.0**	56	52	52	4	0	**0.93**
PBeans	58	58	58	0	0	**1.0**	143	144	140	3	4	**0.95**
Colt	299	299	299	0	0	**1.0**	1272	1280	1254	18	26	**0.97**
Lucene	789	789	789	0	0	**1.0**	3544	3606	3439	105	167	**0.93**
Log4j	251	251	251	0	0	**1.0**	883	853	839	44	14	**0.93**
Tomcat	1494	1487	1487	7	0	**0.99**	6839	6832	6512	327	320	**0.91**
Xerces	876	876	876	0	0	**1.0**	4775	4677	4517	258	160	**0.91**
Ant	1175	1175	1175	0	0	**1.0**	5521	5517	5345	176	172	**0.94**
JFreeChart	624	624	624	0	0	**1.0**	3218	3249	3208	10	41	**0.98**

TABLE 3.13: Similarity between class collaboration networks extracted by A = Dependency Finder and B = Doxygen.

Software system	$\|N_A\|$	$\|N_B\|$	MN	UN_A	UN_B	**JN**	$\|L_A\|$	$\|L_B\|$	ML	UL_A	UL_B	**JL**
CommonsIO	108	100	100	8	0	**0.93**	174	71	71	103	0	**0.41**
Forrest	35	33	33	2	0	**0.94**	52	21	21	31	0	**0.40**
PBeans	58	36	36	22	0	**0.62**	144	19	19	125	0	**0.13**
Colt	299	228	228	71	0	**0.76**	1280	263	263	1017	0	**0.21**
Lucene	789	637	637	152	0	**0.81**	3606	925	907	2699	18	**0.25**
Log4j	251	230	230	21	0	**0.92**	853	246	245	608	1	**0.29**
Tomcat	1487	1310	1303	184	7	**0.87**	6832	1707	1694	5138	13	**0.25**
Xerces	876	813	813	63	0	**0.93**	4677	1494	1494	3183	0	**0.32**
Ant	1175	1055	1055	120	0	**0.90**	5517	1406	1401	4116	5	**0.25**
JFreeChart	624	597	597	27	0	**0.96**	3249	792	792	2457	0	**0.24**

The translation of Java source to Java bytecode can lead to the loss of class dependencies, but also during the compilation new class dependencies that do not exist in the source code may be created. For example, for non-static inner classes the Java compiler always create the synthetic field called

`this$0` which represents the reference to the instance of the outer class. The precise quantification of missing calls links in networks extracted by SNEIPL for software systems used in the comparative analysis is given in Table 3.14.

TABLE 3.14: Quantification of missing CALLS dependencies in networks extracted by SNEIPL: Calls resolved (%) – the fraction of resolved function calls, HTM – the fraction of hard to match functions in the source code, HTM resolved – the number of resolved calls to hard to match function, and HTM unresolved – the number of unresolved calls to hard to match functions.

Software system	**Calls resolved (%)**	**HTM (%)**	**HTM resolved**	**HTM unresolved**
CommonsIO	88.47	30.67	132	102
Forrest	100	0	-	-
PBeans	89.08	7.54	23	84
Colt	94.23	11.58	586	516
Lucene	97.92	8.19	271	206
Log4j	98.47	13.50	162	51
Tomcat	95.46	7.48	832	1494
Xerces	95.96	4.20	535	831
Ant	88.61	4.46	569	2512
JFreeChart	96.28	7.26	483	604

To investigate practical implications of the observed differences between CCNs extracted by SNEIPL, Dependency Finder, and Doxygen, we consider two perspectives: one associated with researchers interested in empirical investigations of design complexity of large-scale software systems, and another with practitioners interested in software metrics.

Researchers interested in the design complexity of real-world, large-scale software systems examine complementary cumulative degree distributions (CCDD) of software networks in order to determine the type of design complexity of studied systems [Hylland-Wood et al., 2006; Myers, 2003; Valverde et al., 2002]. Therefore, we investigated if there are statistically significant differences between CCDDs computed from CCNs formed by SNEIPL, Dependency Finder, and Doxygen. The existence of statistically significant differences between two complementary cumulative distributions can be checked using the two sample Kolmogorov-Smirnov (KS) test [Feller, 1948]. To perform KS tests we used an open-source Java library called JCS (Java Statistical Classes)[7]. The results of KS tests are summarized in Table 3.15. The null hypothesis is accepted if the obtained value of the significance probability (p) is higher than 0.05. It can be observed that for all examined systems there are no statistically significant differences between CCDDs computed from class collaboration networks extracted by SNEIPL and Dependency Finder. In other words, the degree distribution analysis of CCNs obtained by SNEIPL and Dependency Finder would result in the same conclusion about the type of design complexity of corresponding software systems. On the other hand, statistically significant differences between CCDDs computed from CCNs formed by Dependency Finder and Doxygen are present for all examined software systems, except for Forrest.

Since the degree of a node in a CCN is at the same the value of Chidamber-Kemerer CBO metric for the corresponding class, the degree distribution of CCN is at the same time the distribution of CBO values for all classes present in the source code. Therefore, the previous statistical analysis based on KS tests tells us also that there are no statistically significant differences between values of CBO metric when they are computed using CCNs extracted by SNEIPL and Dependency Finder. However, software engineers are usually not interested in the overall statistical properties of metric values, but want to know concrete values of CBO for classes present in a software system. Therefore,

[7] http://www.jsc.nildram.co.uk/

TABLE 3.15: Results of two-sample Kolmogorov-Smirnov tests: D – Kolmogorov-Smirnov statistics, p – the value of the significance probability. "Accepted" denotes if the null hypothesis (no statistically significant differences between distributions) is accepted or not.

	SNEIPL – DependencyFinder			**DependencyFinder – Doxygen**		
Software system	D	p	**Accepted**	D	p	**Accepted**
CommonsIO	0.009	0.99	**yes**	0.45	< 0.01	**no**
Forrest	0.085	0.99	**yes**	0.33	0.057	**yes**
PBeans	0.034	1.00	**yes**	0.72	< 0.01	**no**
Colt	0.013	1.00	**yes**	0.61	< 0.01	**no**
Lucene	0.022	0.98	**yes**	0.52	< 0.01	**no**
Log4j	0.051	0.89	**yes**	0.53	< 0.01	**no**
Tomcat	0.017	0.97	**yes**	0.50	< 0.01	**no**
Xerces	0.042	0.41	**yes**	0.44	< 0.01	**no**
Ant	0.007	1.00	**yes**	0.54	< 0.01	**no**
JFreeChart	0.008	0.99	**yes**	0.52	< 0.01	**no**

we examined the distribution of CBO differences when CBO is computed from CCNs extracted by SNEIPL and Dependency Finder. Results are presented in Table 3.16. As it can be seen, large CBO differences ($\triangle CBO \geq \pm 4$) occur very rarely (for less than 4% of the total number of classes). On the other hand, for more than 65% of the total number of classes in each examined system, the CBO obtained by SNEIPL has the same value as the CBO obtained by Dependency Finder ($\triangle CBO = 0$).

TABLE 3.16: The distribution of CBO differences ($\triangle CBO$) when they are calculated using CCNs extracted by SNEIPL and Dependency Finder.

Software system	0 (%)	± 1 (%)	± 2 (%)	± 3 (%)	$\geq \pm 4$ (%)
CommonsIO	96.3	3.7	-	-	-
Forrest	85.71	11.43	-	-	2.86
PBeans	82.76	13.79	3.45	-	-
Colt	81.61	16.05	0.33	0.67	1.34
Lucene	65.78	26.36	4.69	0.76	2.41
Log4j	76.1	19.92	1.59	0.8	1.59
Tomcat	65.37	23.13	5.45	3.43	2.62
Xerces	64.95	25.23	5.14	1.26	3.42
Ant	72	21.11	3.57	1.45	1.87
JFreeChart	90.54	7.21	1.28	0.64	0.32

Doxygen is not considered in the analysis of CBO differences, because the degree distributions obtained by Doxygen are significantly different from those obtained by Dependency Finder, which automatically implies large CBO differences. It is important to emphasize that CBO calculated from Java source code may be different than CBO calculated from Java bytecode. As pointed earlier, during the compilation some class dependencies may be lost due to inline optimizations, and at the same time new dependencies may be introduced. In other words, the CBO differences presented in Table 3.16 are not caused entirely by two different implementations of CCN extraction, but also by different sources for CCN extraction.

3.5 Analysis of software networks

Complex network theory [Albert and Barabási, 2002; Boccaletti et al., 2006; Newman, 2010, 2003b] provides a set of techniques for statistical analysis and modeling of complex, real-world networks. Those techniques can also be applied to software systems in order to identify and explain connectivity

patterns and evolutionary trends in dependency structures formed by software entities. In Section 3.5.1 we will present an overview of research works dealing with analysis of software systems under the framework of complex network theory. Analysis of software networks extracted using SNEIPL will be covered in subsequent sections.

3.5.1 Related work

Valverde et al. [2002] reported the first empirical evidence of scale-free and small-world properties in software systems. The authors examined the degree distributions, the small-world and clustering coefficients of undirected projections of class collaboration networks associated to JDK (Java Development Kit) in version 1.2 and UbiSoft ProRally (computer game). In their subsequent study Valverde and Solé [2007] investigated the same networks but this time as directed graphs. Moreover, they examined statistical properties of 18 more software networks associated to software systems written in C/C++. The main conclusion of the study is that different software systems exhibit the same pattern of node connectivity characterized by power-law in-degree, out-degree and total-degree distributions.

Myers [2003] examined class collaboration networks representing 3 open source software written in C++ (VTK, DM, AbiWord) and static call graphs representing 3 open source software written in C (Linux, MySQL, XMMS). Analysis of connected components revealed that that all examined networks have a giant weakly connected component (GWCC). For each identified GWCC, Myers computed the complementary cumulative in-degree and out-degree distribution reporting that these distributions have a power-law scaling region. He also investigated degree correlations in GWCCs showing that they exhibit a weak disassortative mixing. Finally, Myers proposed a simple model of software network evolution based on two refactoring techniques: decomposition of large entities into smaller ones and removal of duplicated code. The experimental evaluation of the model showed that it is capable to reproduce empirically observed heavy-tailed in-degree and out-degree distributions.

de Moura et al. [2003] investigated properties of networks associated to four open source software projects written in C/C++ (Linux, XFree86, Mozilla and Gimp). Similarly to previous studies, the authors reported that analyzed networks exhibit scale-free and small-world properties.

Hylland-Wood et al. [2006] analyzed software networks of two Java open source software (Kowari Metastore and JRDF) for fifteen-month period of development. The authors constructed monthly snapshots of the networks at the package, class and method level and investigated properties of their in-degree and out-degree distributions. The results showed that the distributions follow truncated power-laws for each evolutionary snapshot. The study by Jenkins and Kirk [2007] which examined properties of four Java class collaboration networks also indicated that the scale-free property in software systems is persistent across subsequent software releases.

Chatzigeorgiou et al. [2006] showed that class collaboration networks associated to three Java software systems (JUnit, JHotDraw and JRefactory) do not have hub-like cores, i.e. highly coupled classes do not tend to be connected among themselves. The existence of a hub-like core is typically investigated by computing assortativity index, but in the study the authors used an alternative approach based on the S metric proposed by Li et al. [2005]. The authors also demonstrated how various graph-based algorithms can be exploited in OO software engineering.

Puppin and Silvestri [2006] studied the links present among Java classes belonging to various unrelated Java software projects. The authors performed link analysis over the class collaboration network constructed from a set of 49500 classes crawled from Web. The results showed that the in-degree distribution of the network is a power-law curve.

Louridas et al. [2008] analyzed a dataset of nineteen software networks that includes Java class collaboration networks and networks determined by dependencies of Perl packages, libraries in open-source Unix distributions, Windows DLLs, FreeBSD ports, Tex and Metafont modules, and Ruby libraries. They found that in- and out-degree distributions of examined networks can be approximated by power laws, concluding that the scale-free property in software systems appears at various levels of abstraction, in diverse systems and languages. The scale-free property was also reported for class collaboration networks of grid middleware applications [Yuan et al., 2008], agent-oriented applications [Sudeikat and Renz, 2007], inter-package dependency networks for various operating system distributions [Kohring, 2009; Labelle and Wallingford, 2006; Maillart et al., 2008], run-time object collaboration networks [Potanin et al., 2005] and sorting comparison networks [Wen et al., 2009b].

Wheeldon and Counsell [2003] examined statistical properties of software networks that represent different forms of object-oriented (OO) coupling. The networks used in the study were extracted from three Java software systems: Java Development Kit (JDK), Apache Ant and Tomcat. The results of the analysis showed that power-law scaling behavior characterizes different forms of class coupling.

Baxter et al. [2006] extended the study of Wheeldon and Counsell to a larger corpus of software networks associated to 56 Java software systems. In contrast to all previously mentioned studies, the authors considered power-law, log-normal and stretched exponential distributions to model empirically observed degree distributions of class collaboration networks restricted to a particular coupling type. The best fits were obtained using the weighted least square fitting technique. The authors showed that out-degree distributions of software networks restricted to a particular coupling type tend not to have good fits to a power law. On the other side, in-degree distributions have good power law fits.

Concas et al. [2007] presented a comprehensive statistical analysis of an implementation of the Smalltalk system. They analyzed distributions of the following quantities: number of methods per class, number of attributes per class, number of subclasses, number of method calls, size of methods in terms of LOC, size of classes in terms of LOC, in-degree in class collaboration network and out-degree in class collaboration network. Computed complementary cumulative distribution were tested against power-law and log-normal distributions. The parameters of the theoretical distributions were determined using maximum likelihood estimators, while the Pearson's χ^2 test was used to assess the quality of fits. The authors found that the in-degree distributions show a power-law behavior in the tails, while the out-degree distributions exhibit log-normal behavior. In their subsequent study [Concas et al., 2010] the authors examined statistical properties of metrics used in social network analysis (SNA) computed on two software networks. The main conclusion of the study is that empirically observed distributions of SNA metrics, distributions of metrics from the Chidamber-Kemerer (CK) suite, and distribution of node in- and out-degree show power-law behavior in the tails. The authors also reported that SNA metrics and metrics from the CK suite are moderately correlated to the number of defects.

Ichii et al. [2008] analyzed class collaboration networks associated to four open source software written in Java (Ant, JBoss, JDK and Eclipse). The authors reported that the in-degree distributions almost ideally follow a power-law. On the other hand, out-degree distributions do not exhibit power-law behavior through the whole range of out-degree values – the distributions have a "peak" for small values of out-degree followed by a power-law behavior in the tails. The authors also investigated correlations between in-/out-degree and standard software metrics (LOC, two variants of WMC and LCOM). The correlation analysis showed that the out-degree has a high correlation with metrics of internal complexity (LOC and the variants of WMC).

Taube-Schock et al. [2011] examined 97 networks associated to software systems written in Java. The unique characteristic of their work is that the networks were constructed to encompass not only architectural elements as nodes but also statements. The authors examined degree distributions of networks concluding that all of them are heavy-tailed. However, the decision to include statements in analyzed networks is extremely problematic. Statements can not be referenced and typically reference a low number of methods through method calls that are part of the statement. Since the number of statements in any large-scale software is drastically higher than the number architectural elements the resulting network will have a vast majority of nodes having a low degree. Consequently, the average degree will be low and a small number of nodes representing architectural elements will tend to have degree drastically higher than the average. Therefore, a heavy-tailed degree distribution of the network will be practically caused by the existence of nodes representing statements, not by the structure of dependencies among architectural elements. Secondly, the authors computed degree distributions considering only inter-module links (links that constitute coupling among classes) in their heterogeneous networks concluding that they are also heavy-tailed for all examined systems. However, it should be emphasized that the degree distribution of a heterogeneous software network is practically useless since it hides more than it reveals. Namely, a degree distribution shows the probability that a randomly selected node has certain degree. Therefore, relying on the degree distribution of a heterogeneous software network we can know the value of the probability that a randomly selected node has certain degree but we are unable to know what the node represents.

Wen et al. [2009a] analyzed a series of fifty monthly snapshots of the static call graph of the Apache HTTP server. Firstly, the authors observed the presence of the densification law [Leskovec et al., 2005, 2007] in the evolution of the network: the number of links grows super linearly in time with respect to the number of nodes satisfying the power-law of the form $E(t) \sim N(t)^{1.18}$, where $E(t)$ and $N(t)$ are the number of links and the number of nodes at time t, respectively. This can be considered as a bad phenomenon from the software engineering point of view because it means that the average coupling of functions increases as the software evolves. Similarly to Baxter et al. [2006] the authors used the weighted least square fitting technique to examine empirically observed in-degree and out-degree distributions of each snapshot considering power-law, log-normal and stretched exponential distributions as theoretical models. The results of the degree distribution analysis showed that for in-degree distributions power-law provides the best fit for each evolutionary snapshot. For out-degree distributions log-normal provides the best fit, while power-law fits tend to be the worst, even worse than stretched exponential fits.

Wang et al. [2009, 2013] analyzed the evolution of 223 static call graphs corresponding to 223 consecutive versions of the Linux kernel (from version 1.1.0 to 2.4.35). Static call graphs were constructed for each module of the kernel separately, i.e. there was one graph for the file system module, one for the device drivers module, etc. The authors found that the call graphs of the file system, device drivers, kernel, memory management and network module have scale-free and small-world properties. In each evolutionary snapshot the authors identified the 5% nodes with the highest in-degree and the 5% nodes with the highest out-degree. Those nodes were aggregated into so called TDNS (top degree node set). Then they computed so called connecting probability (CP), probability that a node introduced in the next evolutionary snapshot establishes a connection with a node from the TDNS. Formally speaking, CP is the number of links accident to both newly added nodes and nodes from the TDNS normalized by the total number of links accident to newly added nodes. For five kernel modules they obtained a large connecting probabilities indicating that the preferential attachment principle of the Barabási-Albert model can explain the evolution of the network.

Bhattacharya et al. [2012] investigated evolution of static call graphs of eleven open source software systems (Firefox, Blender, VLC, MySQL, Samba, Bind, Sendmail, openSSH, SQLite, Vsftpd) written in C/C++. The authors showed that the average degree for all systems except MySQL increases in time. They also observed that for three systems (Samba, MySQL and Blender) the number of nodes in strongly connected component increases linearly in time. Both of previous two observations can be considered as bad phenomena affecting software quality negatively. All networks, similarly to previous findings, exhibit disassortative mixing. The authors also showed that the page rank metric [Brin and Page, 1998a,b] is a good predictor of bug severity.

Šubelj and Bajec [2012] investigated whether class collaboration networks associated to Java software possess community structure. The study was conducted on 8 networks representing dependencies between classes present in JUnit, JavaMail, Flamingo, Jung, Colt and three namespaces from JDK (org, javax and java). The authors employed three algorithms to detect communities: the Girvan-Newman edge betweenness algorithm [Girvan and Newman, 2002], greedy modularity optimization [Newman, 2004b] and label propagation [Raghavan et al., 2007]. The quality of obtained partitions was assessed using modularity score [Newman and Girvan, 2004]. The results of the experiment showed that examined networks posses a strong community structure with average modularity score in the range [0.55,0.75]. The authors also investigated the correspondence between obtained partitions into communities and the package structure of software systems concluding that identified communities do not exactly correspond to predefined groupings of classes into packages. The study by Fortuna et al. [2011] showed that a strong community structure can be found in networks describing dependencies among packages in the Debian distribution of GNU/Linux operating system. Paymal et al. [2011] investigated the community structure in class collaboration networks extracted from six consecutive versions of JHotDraw software using the greedy modularity optimization technique. The authors observed that two largest communities contain 50% or more of all nodes in each version and that those two communities have continuous and stable growth during software evolution. Finally, Gao et al. [2014] used greedy modularity optimization to identify communities in the static call graph of Linux kernel.

3.5.2 Experimental dataset

Using SNEIPL [Savić et al., 2012, 2014] we formed a dataset of software networks associated to widely used, open-source software systems written in Java. The dataset contains class collaboration networks representing large-scale software systems (more than 10^6 LOC) that were already used in the comparative analysis presented in Section 3.4.3.8. The basic characteristics of the networks are summarized in Table 3.17. Here we investigate the structure of the networks from the experimental dataset. Analysis of package and method collaboration networks will be part of our future work.

TABLE 3.17: Experimental dataset of class collaboration networks. N is the number of nodes, while L is the number of links.

Software system	**Version**	LOC	N	L
Tomcat	7.0.29	329924	1494	6841
Lucene	3.6.0	111763	789	3544
Ant	1.9.2	219094	1175	5521
Xerces	2.11.0	216902	876	4775
JFreeChart	1.0.17	226623	624	3218

3.5.3 Methodological framework

The main characteristic of our study of class collaboration networks is that each node (class or interface) in the network is described by a metric vector. The metric vector contains metrics used in software engineering practice, as well as metrics used in analysis of complex networks. In other words, metric vectors contain both domain-dependent (software metrics) and domain-independent metrics (metrics defined on any directed graph). Employed metrics can be classified into the following categories:

1. Metrics of volume and internal complexity. Those measures reflect how "big" and complex classes are. For each class we computed LOC, CC (cyclomatic complexity), NUMA (the number of attributes in a class) and NUMM (the number of methods in a class).

2. Metrics of coupling (local centrality metrics): in-degree (IN), out-degree (OUT) and total-degree (CBO) of nodes in class collaboration network.

3. Inheritance metrics from the Chidamber-Kemerer metric suite.

4. Metrics of centrality and importance: BET (betweenness centrality) and PR (page rank).

Metrics from the first and third category are domain-dependent metrics. On the other hand, metrics of centrality and importance are domain-independent metrics. Metrics of coupling are also domain-independent metrics but in contrast to the metrics from the fourth category one of coupling metrics (CBO) is widely used in software engineering practice.

The first step in the analysis of complex networks is the identification and characterization of connected components. Class collaboration networks are directed graphs and consequently for each analyzed network we identified both weakly and strongly connected components. To identify weakly connected components we used the breadth first search algorithm. To investigate whether analyzed networks possess the small-world property we measured the small-world and clustering coefficients that are obtained by averaging the values of mentioned metrics for all nodes from a component. The assortativity index is used to measure the extent to which highly connected nodes tend to be directly connected among themselves.

Strongly connected components are identified using the Tarjan algorithm. The tendency of node pairs to form direct mutual dependencies is quantified using the link reciprocity and the normalized link reciprocity measure with respect to a comparable random graph [Garlaschelli and Loffredo, 2004]. We also generalized reciprocity to indirectly connected nodes. Namely, we measured path reciprocity which is the probability that there is a directed path from A to B when there is a directed path from B to A.

In recent years, researchers investigated a variety of real-world software systems revealing the presence of the scale-free (SF) property in software networks (see Section 3.5.1), i.e. indicated that degree distributions of software networks follow power-laws. However, a common point in the majority of those works is that the empirically observed degree distributions were tested only against a power-law. Only in a few studies (Baxter et al. [2006]; Concas et al. [2007]; Wen et al. [2009a]) distributions different than power-law were additionally considered as theoretical models. In this thesis we also investigate statistical properties of degree distributions of class collaboration networks from our experimental corpus testing them against a power-law, exponential and log-normal distributions. In contrast to [Baxter et al., 2006; Concas et al., 2007; Wen et al., 2009a], we employ the power-law test

introduced by Clauset et al. [2009] to determine parameters of theoretical distributions and assess the quality of fits. The mentioned test consists of the three following steps:

1. The scaling parameter of a power-law (α) is determined using the maximum likelihood estimation (MLE) with respect to some lower bound of the power-law behavior in the data (x_m). The MLE for α in the case of a discrete power-law probability distribution is given by

$$\hat{\alpha} = 1 + n \left[\sum_{i=1}^{n} \ln \frac{x_i}{x_m - 0.5} \right]^{-1},$$

   where n is the number of nodes in the network that have degree higher or equal to x_m, and x_i is degree of node i. x_m is determined by the minimization of the weighted Kolmogorov-Smirnov (KS) statistic. The KS statistic is the maximum distance between the cumulative distribution function (CDF) of data and the fitted model. The adjusted version, KS*, adds weights to distances in order to obtain uniform sensitivity across the whole range of values:

$$\mathrm{KS}^* = \max_{x \geq x_m} \frac{|S(x) - P(x)|}{\sqrt{P(x)(1 - P(x))}},$$

   where $S(x)$ is the empirically observed CDF and $P(x)$ is the CDF of the model.

2. A large number of power-law synthetic data is generated using the estimated values of x_m and α in order to compute the goodness of the power-law model. For each synthetic dataset the parameters of the power-law model are determined in the same way as already described in the first step and the value of KS* statistic is recorded. The quality of the power-law fit (p-value) obtained in the first step is the probability that a randomly selected synthetic dataset has higher value of KS* compared to the value KS* statistics obtained in the first step. If the obtained p-value is lower than 0.1 then the power-law hypothesis is rejected, i.e. power-law is not a plausible fit to the empirically observed degree distribution. The number of synthetic dataset has to be higher than 2500 in order to ensure that the p-value is accurate to 2 decimal digits.

3. The parameters of alternative distributions are also determined using the appropriate MLEs. The power-law fit is compared to the fits of alternative distributions using the likelihood ratio test. The null hypothesis of the test is that two theoretical distributions are equally far from an empirically observed distribution.

We can always divide the set of nodes of a class collaboration network into two disjoint groups of classes according to some topological criterion. In this dissertation we introduce the metric-based comparison test that is used to examine difference between two classes of nodes in a class collaboration network where each node in the network is characterized by previously described metric vector. The test is based on the application of the Mann-Whitney U (MWU) test [Mann and Whitney, 1947]. The MWU test is a non-parametric statistical procedure that can be used to check whether values (scores) in one group tend to be greater (or smaller) than the values in the other group when the values of both groups are not normally distributed and groups are not of equal size. Let G_1 and G_2 be two groups of values. The MWU test checks the null hypothesis that the values in G_1 do not tend to be either greater or smaller than the values in G_2. Let g_1 and g_2 be the size of G_1 and G_2, respectively. The test arranges values from both groups into a single ranked series R. The minimal value in R has rank 1 and the maximal value has rank $g_1 + g_2$. In the case that there is a group of tied values then

the average rank of the whole group is assigned to each member of the group[8]. Let U_1 be the number of times a value from G_1 precedes a value from G_2 in R:

$$U_1 = g_1 g_2 + \frac{g_1(g_1+1)}{2} - S_1,$$

where S_1 is the sum of ranks of values from G_1. Similarly we can obtain U_2 which is the number of times a value from G_2 precedes a value from G_1. If the null hypothesis is true then the value of both U_1 and U_2 should be about the half of the total number of comparisons among values which is equal to $g_1 g_2$. Let U be the minimum of U_1 and U_2. Mann and Whitney showed that the limit distribution of U is normal regardless of how g_1 and g_2 approach infinity. Therefore, if the observed standardized value of U is far from the center of the standardized normal distribution $N(0,1)$ the null hypothesis will be rejected.

The effect size of the the Mann-Whitney test can be quantified by so called probability of superiority (PS) [Erceg-Hurn and Mirosevich, 2008]. PS is the probability that a randomly selected value from G_1 is larger than a randomly selected value from G_2. PS can be computed directly from the U statistic:

$$\mathrm{PS} = \frac{U}{g_1 g_2}.$$

However when groups contain tied values PS represents the following probability:

$$\mathrm{PS} = P\{X > Y\} + \frac{1}{2}P\{X = Y\},$$

where X and Y are randomly selected values from G_1 and G_2, respectively. Therefore, we recorded two probabilities of superiority that were computed in a straightforward manner (by comparing each value from G_1 to each value from G_2):

- $\mathrm{PS}_1 = P\{X > Y\}$ – the probability that a randomly selected value from G_1 is larger than a randomly selected value from G_2.
- $\mathrm{PS}_2 = P\{Y > X\}$ – the probability that a randomly selected value from G_2 is larger than a randomly selected value from G_1.

Clearly, $1 - \mathrm{PS}_1 - \mathrm{PS}_2$ is equal to $P\{X = Y\}$.

The metric-based comparison test at the input takes a binary classifier (BC) which separates nodes of a class collaboration network into two classes C_1 and C_2, and an array of metric functions F which map a node to its metric vector. The test at the output produces three sets:

- S_1 - a set of metric descriptors such that for each metric M in the set there is no statistically significant difference between the values of M for classes contained in C_1 and the values of M for classes contained in C_2.
- S_2 - a set of metric descriptors such that for each metric M in the set there is a drastically significant difference between the values of M for classes contained in C_1 and C_2.
- S_3 - a set of metric descriptors that do not belong to S_1 and S_2 which means that for each metric from S_3 a statistically significant difference is observed but it is not too drastic.

[8]Other tie resolving strategies such as the smallest rank, the largest rank or the randomly selected rank from the group are also possible.

The full description of the metric-based comparison procedure is as follows:

- Initialize S_1, S_2 and S_3 to empty sets.
- For each f in F execute the following steps:
  - Initialize G_1 and G_2 to empty sets.
  - Classify each node in the network according to BC. Let m be the value of metric f for a node n. If n belongs to C_1 then add m to G_1, otherwise add m to G_2.
  - Perform the Mann-Whitney U test on G_1 and G_2.
    - If the null hypothesis is accepted add the descriptor of f to S_1.
    - Otherwise if $\mathrm{PS}_1 \geq 0.75$ or $\mathrm{PS}_2 \geq 0.75$ add the descriptor of f to S_2.
    - Otherwise add the descriptor of f to S_3.

3.5.4 Connected component analysis

To identify weakly connected components (WCC) we used the breadth first search algorithm. For each detected WCC we measured the number of nodes, the number of links, diameter, the small-world coefficient, the clustering coefficient and the assortativity index of the component. Obtained values are given in Table 3.18. Each network has a giant WCC which encompasses more than 90% of nodes (in three cases more than 99% of nodes). Moreover, the giant WCCs of Lucene, Xerces and JFreeChart encompass all non-isolated nodes. In each network the fraction of isolated nodes is extremely small (less than 3%) indicating that in examined software systems the degree of unused classes is reduced to a minimum.

TABLE 3.18: Results of the weakly connected component analysis: $\#WCC$ – the number of WCCs, $N(LWCC)$ – the number of nodes in the largest WCC, $L(LWCC)$ – the number of links in the largest WCC, $N(SWCC)$ – the number of nodes in the second largest WCC, I – the number of isolated nodes. All quantities are given in percentages with respect to the total number of nodes (links).

Software system	$\#WCC$	$N(LWCC)$ [%]	$L(LWCC)$ [%]	$N(SWCC)$ [%]	I [%]
Tomcat	32	92.37	96.55	4.82	1.67
Lucene	4	99.62	100	0.12	0.38
Ant	6	99.4	99.96	0.25	0.34
Xerces	3	99.77	100	0.11	0.22
JFreeChart	14	97.92	100	0.16	2.08

The existence of a giant connected component in analyzed networks is not surprising. Namely, the existence of a GWCC means that the vast majority of classes defined in the system work together in order to realize the desired functionality. The existence of two or more large weakly connected components, none of them being giant, would imply large non-interacting software components (sets of classes) realizing unrelated functionalities. In such situations, it is easier for software developers, testers and maintainers to have separate software projects for each weakly connected component. This means that the size of the largest WCC can be used as the estimator of the overall cohesiveness of the software project. Relatively small values of the largest WCC (values below 50%) would definitely indicate one of the following two scenarios:

- The project is an early phase of the development which is conducted in a bottom-up manner, i.e. the product is not growing from a central core of classes.

- The project provides a set of unrelated functionalities.

In the case that the project has a good degree of overall cohesiveness (largest WCC whose size is above 90%) then small size WCCs (including isolated nodes) correspond either to:

- Software components that are in an early phase of development and not yet functionally integrated into the product.
- Deprecated components that are not removed from the source code distribution.

Characteristics of giant weakly connected components are presented in Table 3.19. It can be seen that the small-world coefficients of GWCCs are close to the predictions made by the Erdős-Renyi model of random graphs – $SW \approx SW_r$. Empirically observed clustering coefficients are significantly larger than those of comparable random graphs – $CC \gg CC_{rnd}$. Those two properties imply that GWCCs are small-worlds in the Watts-Strogatz sense [Watts and Strogatz, 1998b]. Finally, GWCCs show slightly disassortative mixing. This means that GWCCs do not possess a hub-like core of nodes, i.e highly coupled classes do not tend be directly coupled to other highly coupled classes.

TABLE 3.19: Characteristics of giant weakly connected components: SW – small-world coefficient, SW_r – the small-world coefficient of comparable random graph, D – diameter, CC – clustering coefficient, CC – clustering coefficient of comparable random graph, A – assortativity index.

Software system	SW	SW_r	D	CC	CC_r	A
Tomcat	3.297	3.258	14	0.239	0.003	-0.102
Lucene	3.753	3.089	14	0.237	0.006	-0.051
Ant	4.680	3.177	12	0.239	0.004	-0.066
Xerces	3.038	2.880	14	0.255	0.006	-0.065
JFreeChart	2.746	2.743	12	0.215	0.009	-0.110

The Tarjan algorithm was employed to identify strongly connected components (SCCs) in GWCCs. The basic characteristics of identified SCCs are shown in Table 3.20. Relatively large size of the largest SCC (ranging from 7% to 25% of the total number of nodes) and the total number of nodes contained in SCCs (ranging from 17% to 36% of the total number of nodes) indicate that examined networks strongly deviate from hierarchical structures. The majority of SCCs are small size components (components that encompasses two or three classes):

- Tomcat has 38 small size SCCs (67.8% of the total number). The largest SCC contains 190 classes[9].
- Lucene has 27 small size SCCs (67.5%). The largest SCC contains 141 classes.
- Ant has 14 small size SCCs (51.85%). The largest SCC contains 286 classes.
- Xerces has 22 small size SCCs (68.7%). The largest SCC contains 121 classes.
- JFreeChart has 13 small size SCCs (68.4%). Two largest SCC contains 44 classes.

The reciprocity of links in examined networks is relatively small ($0.03 < R \leq 0.08$) but still higher than expected by random chance ($R_n > 0$). On the other hand, the path reciprocity is significantly higher than the link reciprocity for Tomcat, Ant and Lucene (at least two times higher). In the context of class collaboration networks, path reciprocity is the conditional probability that class A directly or

[9]Interfaces are also counted as classes.

TABLE 3.20: Characteristics of strongly connected components: $\#SCC$ – the number of SCCs, $LSCC$ – the size of the largest SCC, $N(SCC)$ – the total number of nodes contained in SCCs , R – reciprocity, R_n – normalized reciprocity, R^p – path reciprocity.

Software system	$\#SCC$	$LSCC$ [%]	$N(SCC)$ [%]	R	R_n	R^p
Tomcat	56	12.72	35.74	0.078	0.075	0.179
Lucene	40	17.87	35.23	0.080	0.075	0.162
Ant	27	24.34	35.06	0.046	0.042	0.237
Xerces	32	13.81	32.76	0.078	0.072	0.118
JFreeChart	19	7.05	17.63	0.032	0.024	0.048

indirectly depends on B when B directly or indirectly depends on A. For example, the value of path reciprocity for the Ant class collaboration network is equal to 0.237 which means that nearly quarter of all (both direct and indirect) dependencies among classes present in Ant are cyclic dependencies.

From the software engineering perspective, large cyclic dependencies among classes are undesirable and considered as anti-patterns. As noted by Fowler [2001], circular dependencies among software entities should be avoided in order to prevent possible vicious circle of change propagation. Large SCCs are the source of many problems to software developers because it is hard to isolate any entity in the SCC [Melton and Tempero, 2006]. Anyone wanting to understand one entity in the SCC effectively has to understand every entity in the SCC. Anyone wanting to test one entity in the SCC has to test the whole SCC. A direct implication of previous two observations is an increased cost of maintenance for such entities. Although various software methodologies advise to avoid cyclic dependencies among software entities, our results show that large cycles are present in the structures of the analyzed software systems. Additionally, our results are consistent with the findings of Melton and Tempero [2007], who showed that large cyclic dependencies are common for classes present in Java software systems.

The number of links in a SCC determines the complexity of the component. For example, if we have two SCCs of the same size then the one with the higher number of links is more complex. In general case we can say that component S_1 is more complex than component S_2 if

$$\frac{L(S_1)}{N(S_1)} > \frac{L(S_2)}{N(S_2)},$$

where $L(S)$ and $N(S)$ are the number of links and the number of nodes in SCC S, respectively. Since the minimal number of links in a SCC is equal to the number of nodes in the SCC then $L(S)/N(S)$ actually measures to what extent S deviates from being a pure cycle containing N mutually reachable nodes. The stronger deviation from the pure cycle indicates a more complex SCC regardless of its size. On the other hand, $L(S)/N(S)$ can be viewed as the average intra-component degree – the average number of in-coming/out-going links incident to a node in S considering only links whose both ends are in S and ignoring the rest of links. Therefore, for each detected SCC we determined the set of links contained in the SCC in order to compute its average intra-component degree. Then we investigated whether there is a correlation between SCC size and average intra-component degree. The values of the Sperman rank correlation coefficient for examined software systems are given in Table 3.21. As it can be observed there is a strong Sperman correlation (always higher than 0.8) between the size of SCC and average intra-component degree. This means that the complexity of SCC tends to increase with the size of SCC. In other words, SCCs tend to *densify* with size: larger SCC then it more deviates from being a pure cycle compared to a smaller SCC. From the practical (software engineering) point

of view this result implies that large SCCs are extremely hard to comprehend, test and refactor since the cohesiveness of cyclic dependencies increases with the number of mutually dependent classes.

TABLE 3.21: Densification of strongly connected components. $\rho(N(S), \frac{L(S)}{N(S)})$ – Spearman's rank correlation between size and average intra-component degree of SCCs, α – scaling exponent of empirically observed densification law $L(S) \approx N(S)^{\alpha}$.

Software system	$\rho(N(S), \frac{L(S)}{N(S)})$	α
Tomcat	0.975	1.29
Lucene	0.965	1.27
Ant	0.982	1.27
Xerces	0.977	1.33
JFreeChart	0.825	1.32

A strong correlation between component size and average intra-component degree suggests that the number of links in SCCs grows super-linearly with respect to the number of nodes in SCCs:

$$L(S) \approx N(S)^{\alpha}, \alpha > 1.$$

Higher values of the scaling exponent α indicate a higher degree of densification. For each examined system we made a log-log plot in which each point represents one SCC (see Figure 3.8). The x coordinate of a point corresponds to the number of nodes in the SCC, while the y coordinate corresponds to the number of links. Then we fitted a power-law curve of the form x^{α} using the chi-square minimization technique provided by the MicroCal Origin data analysis software. As it can be observed from Figure 3.8 the relationship between the number of nodes and the number or links in SCC can be very well approximated by a power-law (the coefficient of determination is always higher than 0.9). Obtained power-law exponents (given on the plots but also in Table 3.21) indicate the degree of densification of SCCs in analyzed systems. The scaling exponent of an empirically observed densification law can be viewed as the indicator of design quality. Smaller scaling exponent implies lower complexity of strongly connected components present in the system and thus a better design.

To investigate characteristics of strongly connected components we applied the metric-based comparison test described in Section 3.5.3 where the first class of nodes (C_1) are nodes belonging to SCCs and the second class of nodes (C_2) are nodes that do not belong to any SCC. The results of the test are presented in Table 3.22. For each metric M the table shows:

- $\overline{C_1}$ – the average value of metric M for nodes that are contained in strongly connected components.
- $\overline{C_2}$ – the average value of metric M for nodes that are not contained in strongly connected components.
- U – obtained test statistic used in the Mann-Whitney U test.
- p – obtained significance probability of the Mann-Whitney U test. If this probability is lower than 0.05 then the null hypothesis of the test is rejected.
- PS_1 - the probability that a randomly selected node belonging to a SCC has a higher value of M compared to a randomly selected node not contained in a SCC.

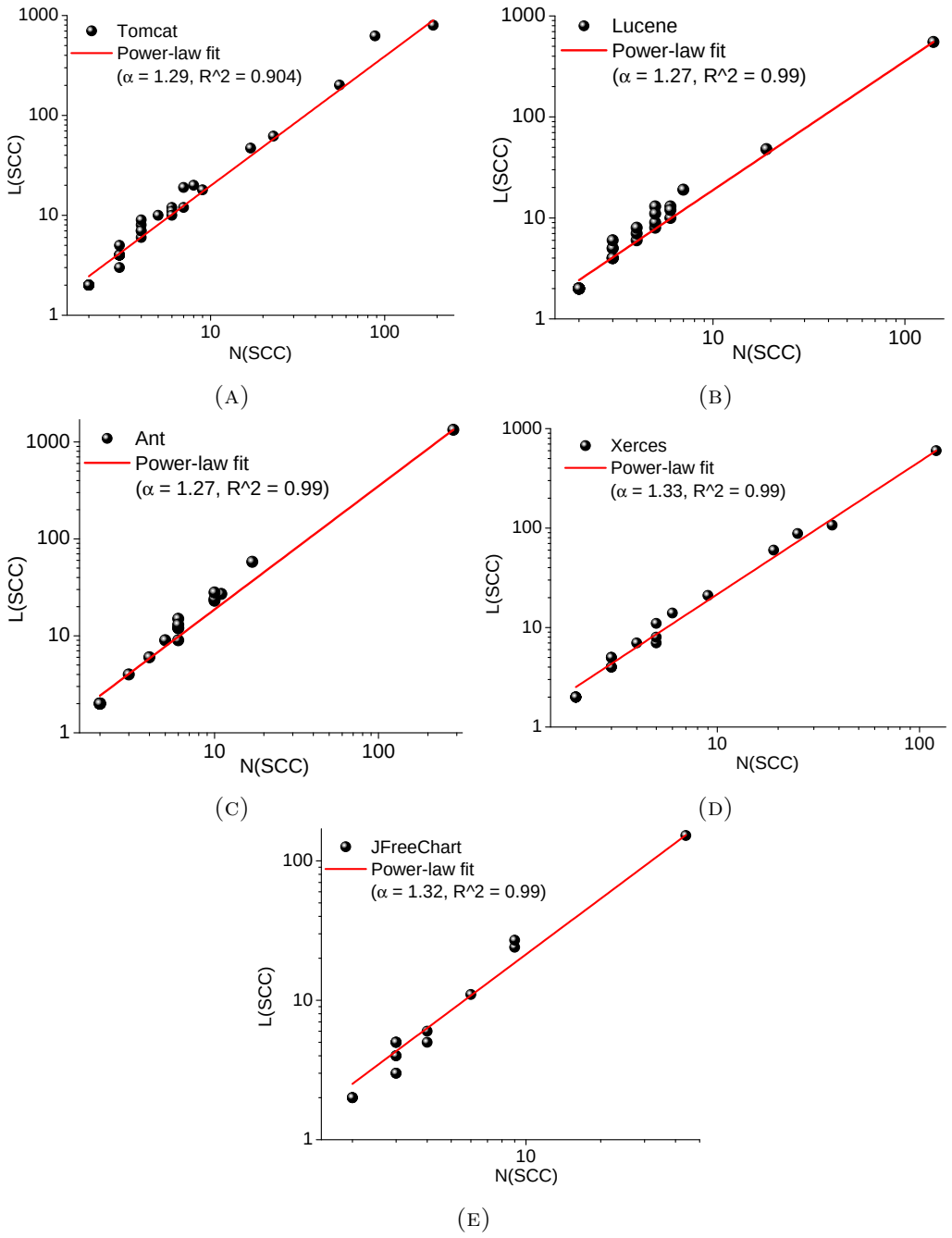


FIGURE 3.8: Densification of strongly connected components in (A) Tomcat, (B) Lucene, (C) Ant, (D) Xerces, and (E) JFreeChart.

- PS_2 - the probability that a randomly selected node not belonging to a SCC has a higher value of M compared to a randomly selected node contained in a SCC.

For example, the first row in Table 3.22 shows that the average LOC of Tomcat's classes involved in cyclic dependencies is 289.17, the average LOC of classes not involved in cyclic dependencies is 161.06, there is statistically significant difference of those two groups of classes considering their volume (the null hypothesis of the MWU test is rejected), and the probability that the LOC of a randomly selected class involved in cyclic dependencies is strictly higher than the LOC of a randomly selected class not involved in cyclic dependencies is 0.56, while the opposite probability is 0.43.

TABLE 3.22: The results of the metric-based comparison test for strongly connected components.

Software system	Metric	$\overline{C_1}$	$\overline{C_2}$	U	p	NullHyp	PS_1	PS_2
Tomcat	LOC	289.17	161.06	291224	$< 10^{-4}$	rejected	0.56	0.43
	CC	27.32	17.24	281184	0.002	rejected	0.48	0.38
	NUMA	5.21	4.77	270537	0.075	**accepted**	0.47	0.41
	NUMM	14.26	8	308247	$< 10^{-4}$	rejected	0.57	0.36
	IN	6.38	3.57	339696	$< 10^{-4}$	rejected	0.56	0.24
	OUT	7.29	3.06	386846	$< 10^{-4}$	rejected	0.72	0.21
	CBO	12.68	6.64	368293	$< 10^{-4}$	rejected	0.68	0.25
	NOC	0.46	0.22	262066	0.472	**accepted**	0.09	0.06
	DIT	0.6	0.42	297607	$< 10^{-4}$	rejected	0.34	0.18
	BET	1847.29	124.63	386286	$< 10^{-4}$	rejected	0.68	0.17
	PR	0.000896	0.000543	341903	$< 10^{-4}$	rejected	0.66	0.33
Lucene	LOC	207.19	94.66	94019	$< 10^{-4}$	rejected	0.66	0.33
	CC	21.18	15.37	96546	$< 10^{-4}$	rejected	0.64	0.27
	NUMA	5.42	2.8	88067	$< 10^{-4}$	rejected	0.56	0.32
	NUMM	10.11	6.1	87275	$< 10^{-4}$	rejected	0.58	0.35
	IN	6.03	3.65	90841	$< 10^{-4}$	rejected	0.55	0.27
	OUT	7.27	2.98	110913	$< 10^{-4}$	rejected	0.74	0.18
	CBO	12.28	6.63	100382	$< 10^{-4}$	rejected	0.68	0.26
	NOC	0.72	0.39	76054	0.1	**accepted**	0.2	0.13
	DIT	0.97	0.72407	81111	0.0009	rejected	0.4	0.26
	BET	1522.37	102.47	111895	$< 10^{-4}$	rejected	0.74	0.16
	PR	0.001415	0.001187	90250	$< 10^{-4}$	rejected	0.63	0.36
Ant	LOC	194.4	162.2	176965	0.00036	rejected	0.56	0.43
	CC	15.69	14.29	174507	0.0018	rejected	0.51	0.39
	NUMA	4.36	5.01	160886	0.504	**accepted**	0.42	0.44
	NUMM	9.86	8.23	172325	0.006	rejected	0.51	0.42
	IN	9.78	1.95	228723	$< 10^{-4}$	rejected	0.64	0.19
	OUT	5.93	4.02	200409	$< 10^{-4}$	rejected	0.59	0.31
	CBO	15.11	5.98	215327	$< 10^{-4}$	rejected	0.65	0.28
	NOC	1.17	0.21	168921	0.0343	rejected	0.16	0.08
	DIT	1.11	1.14	157439	0.9624	**accepted**	0.35	0.35
	BET	3255.16	103.95	253004	$< 10^{-4}$	rejected	**0.76**	0.14
	PR	0.001744	0.000369	242063	$< 10^{-4}$	rejected	**0.77**	0.22
Xerces	LOC	385.88	155.29	105178	$< 10^{-4}$	rejected	0.62	0.37
	CC	53.89	14.47	101519	$< 10^{-4}$	rejected	0.51	0.31
	NUMA	8.16	4.66	100067	$< 10^{-4}$	rejected	0.53	0.34
	NUMM	14.96	8.29	108053	$< 10^{-4}$	rejected	0.61	0.33
	IN	6.49	4.94	108948	$< 10^{-4}$	rejected	0.56	0.27
	OUT	9.19	3.62	125479	$< 10^{-4}$	rejected	0.7	0.21
	CBO	14.39	8.57	113871	$< 10^{-4}$	rejected	0.64	0.29
	NOC	0.74	0.24	92605	0.0214	rejected	0.19	0.09
	DIT	1.56	0.75	104528	$< 10^{-4}$	rejected	0.44	0.2
	BET	1888.39	108.65	131778	$< 10^{-4}$	rejected	0.74	0.18
	PR	0.001643	0.000897	116151	$< 10^{-4}$	rejected	0.69	0.31
JFreeChart	LOC	548.27	245.49	30856	0.13	**accepted**	0.54	0.45
	CC	36.28	15.69	29545	0.46	**accepted**	0.46	0.42
	NUMA	8.06	3.88	30721	0.15	**accepted**	0.48	0.39
	NUMM	25.6	11.56	31787	0.04	rejected	0.54	0.41
	IN	10.23	4.07	44351	$< 10^{-4}$	rejected	**0.75**	0.18
	OUT	7.48	4.66	35014	$< 10^{-4}$	rejected	0.57	0.33
	CBO	16.79	8.73	37484	$< 10^{-4}$	rejected	0.63	0.31
	NOC	1.04	0.33	32150	0.02	rejected	0.22	0.08
	DIT	0.55	0.92	31164	0.09	**accepted**	0.26	0.36
	BET	924.85	38.88	45216	$< 10^{-4}$	rejected	**0.76**	0.15

Continued on next page

Table 3.22 – continued from previous page

Software system	Metric	$\overline{C_1}$	$\overline{C_2}$	U	p	NullHyp	PS_1	PS_2
	PR	0.002716	0.001364	45590	$< 10^{-4}$	rejected	**0.81**	0.19

In the case of Tomcat, And and JFreeChart the null hypothesis of the MWU test was accepted for the NUMA (the number of attributes in a class) metric. This means that classes involved in cyclic dependencies in mentioned software systems do not tend to declare more class attributes compared to classes not involved in cyclic dependencies. The MWU tests for NOC (the number of children, i.e. the number of classes extending a class) and DIT (the depth in inheritance tree) were also accepted for some of examined software systems:

- In the case of Tomcat and Lucene the null hypothesis of the MWU test was accepted for the NOC metric indicating that highly extensible classes present in those software systems do not tend to be involved in cyclic dependencies more than lowly extensible classes.
- In the case of Ant and JFreeChart the null hypothesis of the MWU test was accepted for the DIT metric indicating highly specialized classes present in those software systems do not tend to be involved in cyclic dependencies more than lowly specialized (highly abstract) classes.

In the case of JFreeChart the null hypotheses were also accepted for two metrics of internal complexity – LOC (lines of code) and CC (cyclomatic complexity). It can be seen that the average values of LOC and CC for classes involved in cyclic dependencies are two times higher compared to the average values for the rest of classes. However, those differences in average values are not statistically significant. The probability that a class from a SCC has higher internal complexity (estimated either by LOC or CC) than a class not involved in cyclic dependencies, PS_1, is not drastically higher than the opposite probability of superiority PS_2.

Only for one software systems, Xerces, all null hypotheses were rejected. The probability of superiority of the first class of nodes (classes involved in cyclic dependencies) is higher than than the probability of superiority of the second class of nodes (classes not involved in cyclic dependencies) for all examined metrics, but drastic differences are absent ($PS_1 < 0.75$ for all examined metrics). The largest differences ($PS_1 > 0.7 \wedge PS_2 < 0.25$) are present for OUT (out-degree) and BET (betweenness centrality) which means that classes involved in cyclic dependencies moderately tend to aggregate a high number of other classes and tend to be central classes in the design structure.

Drastic differences between metric values of classes belonging to SCCs and classes that are not involved in cyclic dependencies are present in two systems: Ant and JFreeChart. In both systems classes involved in cyclic dependencies tend to have higher values of BET and PR (Page Rank) implying that there is a strong tendency that strongly connected components contain the most central and the most important classes present in those systems. Additionally, highly reused classes in JFreeChart strongly tend to be involved in cyclic dependencies.

3.5.5 Degree distribution analysis

For each giant weakly connected component of the class collaboration networks from our experimental dataset we computed corresponding in-degree ($P_{in}(k)$), out-degree ($P_{out}(k)$) and total-degree ($P(k)$) distributions. Small, isolated clusters of trivial complexity which increase $P(k)$, $P_{in}(k)$ and $P_{out}(k)$ for small values of k are omitted from analysis, and the analysis itself is focused on structures that

carry the bulk of the complexity of systems under investigation. Table 3.23 shows the basic quantities describing empirically observed degree distributions:

- Mean degree (μ) – the average number of (in/out/total) links incident to a node.
- Standard deviation of the mean degree (σ) which measures the amount of variation from the average degree.
- Coefficient of variation (c_v) is a normalized measure of dispersion defined as the ratio of the standard deviation and the mean degree.
- Skewness (G_1) which is the third standardized moment of the distribution and quantifies its asymmetry. Skew equals to 0 implies that the distribution is perfectly symmetric (e.g. the normal distribution). Negative skewness indicates that the left tail of the distribution is longer than the right tail, while the positive skewness indicates the opposite.
- Maximal degree (M).

TABLE 3.23: The basic characteristics of empirically observed degree distributions.

Software system	Distribution	μ	σ	c_v	G_1	M
Tomcat	Total-degree	9.58	15.93	1.66	8.05	293
	In-degree	4.79	13.86	2.90	11.49	293
	Out-degree	4.79	7.07	1.48	3.36	73
Lucene	Total-degree	9.02	11.58	1.28	5.95	175
	In-degree	4.51	9.84	2.18	7.22	153
	Out-degree	4.51	5.30	1.17	2.93	46
Ant	Total-degree	9.46	24.73	2.62	15.25	534
	In-degree	4.73	23.57	4.99	16.53	533
	Out-degree	4.73	5.46	1.16	2.38	40
Xerces	Total-degree	10.92	14.23	1.30	2.87	106
	In-degree	5.46	10.97	2.01	4.22	105
	Out-degree	5.46	8.71	1.60	3.84	91
JFreeChart	Total-degree	10.54	16.02	1.52	5.78	211
	In-degree	5.27	14.01	2.66	7.95	211
	Out-degree	5.27	7.48	1.42	4.39	93

From the data presented in Table 3.23 it can be observed that the average total degrees are in the range $[9, 11]$ which means that an average class present in examined software systems is coupled to approximately 10 other classes. However, the values of the coefficient of variation, skewness and maximal degree indicate the presence of hubs in the networks – classes whose in/out/total degree are significantly higher than the average degree. For example, the most coupled class defined in Ant has the total degree that is 56 times higher than the average, the most internally reused class has in-degree that is 112 times higher than the average, while the class that references the largest number of classes has out-degree that is 8.5 times higher than the average. The absence of hubs means that all nodes in the network have in/out/total degree that is close to the average, i.e. extreme values of degree are absent and consequently the network can be modeled using the Erdős-Renyi model of random graphs. The basic characteristic of large-scale, finite size Erdős-Renyi random graphs is that their degree distributions can be well approximated by Poisson distribution. The Poisson distribution has coefficient of variation and skewness that are equal to $1/\sqrt{\mu}$. For random graphs comparable to examined networks $1/\sqrt{\mu}$ is always smaller than 1 since $\mu > 1$. As it can be observed the coefficient

of variation of empirically observed degree distributions is always higher than 1, while the skewness is drastically higher than 1 indicating the strong of deviation from the Erdős-Renyi model of random graphs and the presence of strong hubness in the networks.

Using the power-law test introduced in [Clauset et al., 2009] (whose implementation is provided by the poweRlaw R package[10]) we investigated whether the degree distributions of examined software systems follow power-laws and consequently exhibit the scale-free property. Additionally, the test compares the best power-law fit to the best fits of exponential probability mass function (PMF) and log-normal probability mass function in obtained power-law scaling region. The results of the test are summarized in Table 3.24, while the plots are given in Appendix A. As it can be observed power-laws are plausible models for all examined degree distributions since obtained $p-$values are always higher than 0.1. Obtained values of the lower bound of power-law scaling region (denoted by x_m in Table 3.24) are always higher than 1 which means that the distributions exhibit power-law behavior in the tails. However, for all out-degree distributions and the total degree distribution of the Xerces CCN corresponding x_m value is considerably high and consequently the sizes of the power-law scaling regions (denoted by L in Table 3.24, $L = \max(x) - x_m$) are relatively small.

TABLE 3.24: The results of the power-law test.

Software system	Distribution	x_m	L	α	p-value	R_{ln}	$p(R_{ln})$	R_e	$p(R_e)$
Tomcat	Total-degree	17	276	2.8	0.99	-0.68	0.49	1.95	0.05
	In-degree	4	289	2.07	0.74	-1.84	0.06	3.32	0.0009
	Out-degree	19	54	3.69	0.46	-1.07	0.28	-0.75	0.45
Lucene	Total-degree	10	165	2.68	0.64	-1.39	0.16	1.24	0.21
	In-degree	3	150	1.95	0.17	-2.76	0.005	1.75	0.08
	Out-degree	11	35	3.72	0.84	-0.29	0.76	1.28	0.19
Ant	Total-degree	11	523	2.6	1.00	0.41	0.68	2.77	0.005
	In-degree	7	526	2.08	0.78	-0.38	0.7	3.36	0.0007
	Out-degree	14	26	4.17	0.42	-0.94	0.34	-0.29	0.76
Xerces	Total-degree	49	57	4.65	0.83	-0.75	0.45	-0.85	0.39
	In-degree	6	99	2.1	0.60	-2.45	0.01	1.18	0.23
	Out-degree	27	64	4.29	0.66	-0.22	0.81	0.53	0.59
JFreeChart	Total-degree	14	197	2.65	0.88	-0.47	0.63	2.38	0.01
	In-degree	9	202	2.31	0.94	-0.4	0.68	2.31	0.02
	Out-degree	14	79	3.79	0.14	-0.07	0.94	1.16	0.24

The results of the likelihood ratio tests which compare the power-law fits to the best fits of the log-normal and exponential PMFs in obtained power-law scaling regions are also shown in Table 3.24. The value of the log likelihood ratio is denoted by R_d in Table 3.24, where d is the alternative distribution ("ln" – log-normal, "e" – exponential). Positive and statistically significant R_d ($R_d > 0, p(R_d) < 0.1$) indicates that the power-law fit is favored over the best fit of the alternative distribution d. On the other side, negative and statistically significant R_d ($R_d < 0, p(R_d) < 0.1$) implies that the alternative distribution better fits the tail of the distribution. If R_d is not statistically significant ($p(R_d) \geq 0.1$) then the best fits of both theoretical distributions are equally far from the empirically observed tail where the power-law fit is plausible. It can be seen that for all distributions that have a small power-law scaling region (all out-degree distributions and the total degree distribution of Xerces class collaboration network) all considered theoretical distributions are equally plausible models. Moreover, all considered theoretical distributions are equally plausible models for the tail of the total degree

[10] http://cran.r-project.org/web/packages/poweRlaw/index.html

distribution of the Lucene class collaboration network. For the rest of distributions it can be concluded that:

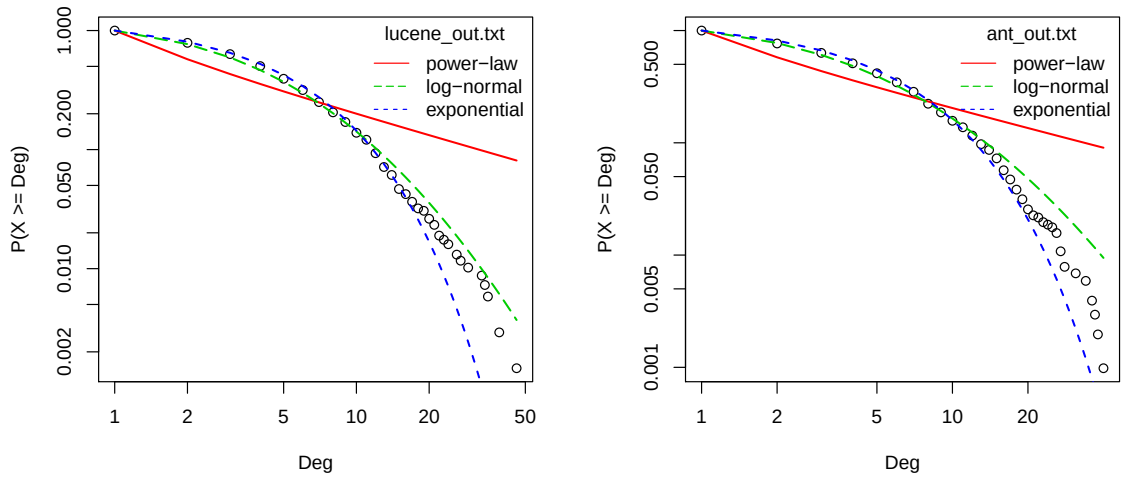
- The best power-law fit is always better than the exponential fit except for the tail of the Xerces in-degree distribution where the log-normal PMF is the most plausible model.
- The power-law fit is never preferred over the log-normal fit. To the contrary, log-normal PMF is the better model for the tail of the Ant in-degree distribution, the Lucene in-degree distribution, and the Xerces in-degree distribution.

In other words, our degree distribution analysis showed that there is a moderate support for the power-law behavior in the tails of degree distributions of examined class collaboration networks. Power-laws are plausible fits in the tails but alternative distributions are either equally plausible models or even provide better fits. Moreover, for some distributions the power-law scaling regions are relatively small suggesting that in such cases the power-law is poor model for the data. Therefore, using the likelihood ratio test we also investigated which theoretical distribution provides the best fit considering the whole range of degree values ($x_m = 1$). The results are summarized in Table 3.25. The value of the log likelihood ratio is denoted by $R\left(\frac{d_1}{d_2}\right)$, where d_1 and d_2 are two theoretical distributions ("pw" – power-law, "ln" – log-normal, "e" – exponential). As it can be seen log-normal distribution provides the best to all distributions except for the out-degree distribution of Lucene and Ant (two distributions with the lowest coefficient of variation and skewness, see Table 3.23) where log-normal and exponential distributions are equally plausible. The complementary cumulative out-degree distribution for Lucene and Ant with accompanying power-law, log-normal and exponential fits through the whole range of out-degree values are shown in Figure 3.9. As it can be visually observed the values in the tails are higher compared to the exponential fits and consequently we can conclude that those distributions are heavy-tailed. Others distributions are also heavy-tailed since their tails are not exponentially bounded: the distributions have higher coefficient of variation and skewness than exponential distribution ($c_v(\text{EXP}) = 1$, $G_1(\text{EXP}) = 2$) and the log-normal fits through the whole range of values are preferred over the exponential fits.

TABLE 3.25: The results of the power-law test through the whole range of values ($x_m = 1$).

Software system	Distribution	$R\left(\frac{\text{pw}}{\text{ln}}\right)$	p	$R\left(\frac{\text{pw}}{\text{e}}\right)$	p	$R\left(\frac{\text{ln}}{\text{e}}\right)$	p	Best fit
Tomcat	Total-degree	-19.56	$< 10^{-4}$	-7.80	$< 10^{-4}$	3.81	0.0001	LN
	In-degree	-5.04	$< 10^{-4}$	5.77	$< 10^{-4}$	6.64	$< 10^{-4}$	LN
	Out-degree	-12.88	$< 10^{-4}$	-6.23	$< 10^{-4}$	3.78	0.0002	LN
Lucene	Total-degree	-19.66	$< 10^{-4}$	-14.44	$< 10^{-4}$	3.86	0.0001	LN
	In-degree	-4.68	$< 10^{-4}$	4.70	$< 10^{-4}$	6.12	$< 10^{-4}$	LN
	Out-degree	-11.88	$< 10^{-4}$	-10.12	$< 10^{-4}$	0.31	0.7578	LN/EXP
Ant	Total-degree	-18.89	$< 10^{-4}$	-4.24	$< 10^{-4}$	3.26	0.0011	LN
	In-degree	-3.06	0.0022	4.24	$< 10^{-4}$	4.61	$< 10^{-4}$	LN
	Out-degree	-13.63	$< 10^{-4}$	-11.27	$< 10^{-4}$	-0.88	0.3807	LN/EXP
Xerces	Total-degree	-18.48	$< 10^{-4}$	-10.86	$< 10^{-4}$	5.82	$< 10^{-4}$	LN
	In-degree	-5.08	$< 10^{-4}$	7.00	$< 10^{-4}$	9.15	$< 10^{-4}$	LN
	Out-degree	-10.19	$< 10^{-4}$	-2.06	0.0397	5.37	$< 10^{-4}$	LN
JFreeChart	Total-degree	-16.20	$< 10^{-4}$	-9.28	$< 10^{-4}$	4.07	$< 10^{-4}$	LN
	In-degree	-4.51	$< 10^{-4}$	3.57	0.0004	4.94	$< 10^{-4}$	LN
	Out-degree	-8.95	$< 10^{-4}$	-3.40	0.0007	2.49	0.0127	LN

Having in mind the definition of scale-free networks we can conclude that examined class collaboration networks are not scale-free for two reasons:



(A) (B)

FIGURE 3.9: Complementary cumulative out-degree distribution for (A) Lucene and (B) Ant.

- Log-normal distribution provides better fit to empirically observed degree distribution through the whole range of degree values compared to power-law.
- The tails of the distribution can be modeled by power-laws but alternative distributions are either equally plausible models or even provide better fits.

However, the main characteristic of scale-free networks is that they contain hubs – highly connected nodes whose degrees are significantly higher than the average degree. The same holds for examined class collaboration networks which also contains hubs due to heavy-tailed degree distributions. The existence of hubs and emergence of power-law scaling behavior in complex networks is usually explained by the preferential attachment principle of the Barabási-Albert model [Barabasi and Albert, 1999]. The evolution of the network governed by the preferential attachment principle is based on the observation that new nodes tend to attach to hubs increasing their degree and consequently their hubness. Formally speaking, the probability Π that a new node establishes link to an old node is directly proportional to the degree of the old node k:

$$\Pi(k) \propto k.$$

On the other side, log-normal degree distributions arise from the nearly-linear preferential attachment [Redner, 2005] of the form

$$\Pi_{nl}(k) \propto \frac{k}{1 + a \ln k}.$$

Therefore, both log-normal and power-law degree distributions can be explained by some form of preferential attachment with the difference that the preferential attachment probability in scale-free networks is slightly higher compared to networks with log-normal degree distributions.

In software development practice, it is desirable to keep class coupling as low as possible. Heavy-tailed total degree distributions implies that coupling among classes has no characteristic scale: average class coupling is relatively small, but there is a statistically significant number of highly coupled classes whose degree of coupling is extremely large. From the software engineering perspective, this

phenomenon is considered to be bad, because highly coupled entities can cause difficulties in software maintenance, testing and comprehension. Therefore, in the next Section of the dissertation we will investigate in detail the characteristics of highly coupled classes (hubs in class collaboration networks) in order to provide a deeper understanding of the high coupling phenomenon in object-oriented software systems.

3.5.6 Characteristics of highly coupled classes

As emphasized in the previous Section total degree distributions of class collaboration networks from our experimental dataset are heavy-tailed implying that they contain hubs – classes with extremely high value of the Chidamber-Kemerer CBO metric. Let C denote the set of nodes in a class collaboration network. We split C into two disjoint sets H and O ($C = H \cup O, H \cap O = \emptyset$) where H contains hubs and O contains those classes that are not highly coupled. H will be the *minimal* set satisfying the following condition

$$\sum_{h \in H} \text{degree}(h) > \sum_{o \in O} \text{degree}(o).$$

In other words, H is the minimal set of highly coupled classes whose total CBO is higher than the total CBO of the rest of the classes contained in the system. Table 3.26 shows the fraction of hubs in examined systems and their minimal total degree (CBO). For example, 13.41% classes defined in Tomcat (classes with CBO higher or equal to 18) have the total CBO higher than the total CBO of the rest of classes.

TABLE 3.26: The fraction of highly coupled classes (H) and the minimal total degree (CBO) of highly coupled classes (H_d).

Software system	H [%]	H_d
Tomcat	13.41	18
Lucene	17.3	14
Ant	12.92	15
Xerces	13.73	23
JfreeChart	15.06	17

Looking to the data presented in Table 3.23 we can observe that the coefficients of variation, skewness and maximal degree of in-degree distributions are (drastically) higher than the same quantities describing out-degree distributions. This means that the tails of the in-degree distributions are more longer than the tails of the out-degree distributions. The previous observation suggests that highly coupled classes, classes contained in tails of the total-degree distributions, tend to have higher in-degree than out-degree. For example, the class with the highest total degree in Tomcat has the total degree equal to 293 and at the same time this is the class with the highest in-coming degree which is also equal to 293. This means that the coupling of the most coupled class in Tomcat is entirely caused by internal reuse since this class does not reference any other class defined in Tomcat. The class with the highest total degree in Ant has total degree equal to 534, in-coming degree equal to 533 and out-going degree equal to 1. In other words, basic statistical quantities describing empirically observed in-degree and out-degree distributions of examined class collaboration network suggest that there is some kind of disbalance between in-degree and out-degree of highly coupled classes where in-degree tends to be significantly higher. In order to give a precise quantification of the disbalance phenomena we define the following two metrics:

1. C_k which is the average ratio of in-degree to total-degree (CBO) for classes whose total-degree is higher or equal to k. C_k is a normalized measure taking values in the range $[0, 1]$ since in-degree is always smaller or equal to total-degree. High values of C_k for large k implies that high coupling is dominantly caused by internal reuse, not by internal aggregation.

2. P_k which is the the probability that a randomly selected class whose total degree is higher or equal to k has two times higher in-degree than out-degree.

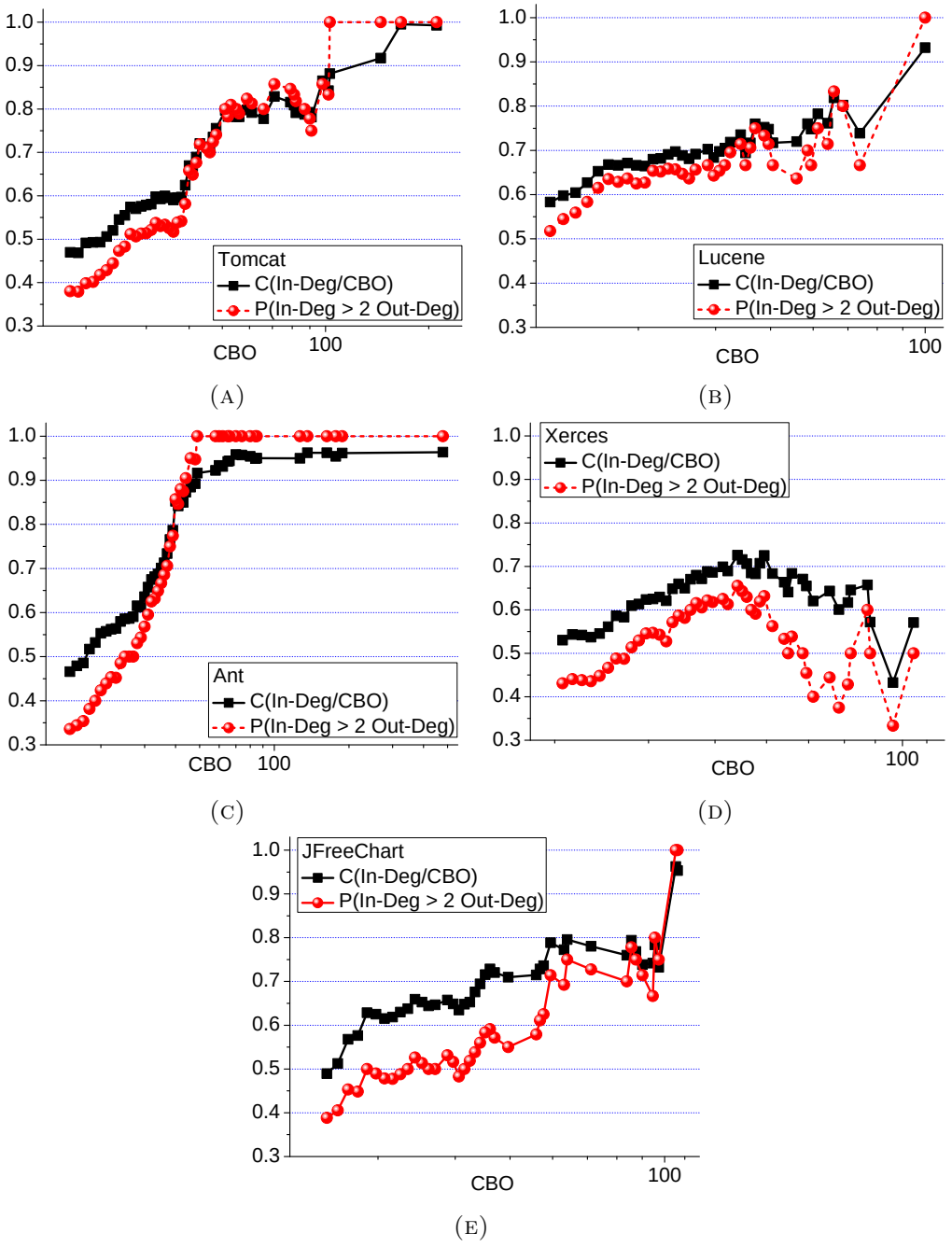


FIGURE 3.10: In-Out degree disbalance for (A) Tomcat, (B) Lucene, (C) Ant, (D) Xerces, and (E) JFreeChart.

Figure 3.10 shows the value of C_k and P_k for $k \geq H_d$ where H_d is the minimal total degree in the set of highly coupled nodes. As it can be observed for all software systems except for Xerces both C_k

and P_k tend to increase with k. This means that the disbalance between in- and out-degree becomes more drastic with higher values of CBO metric.

In the case of Xerces C_k starts to decrease from 0.71 at $k = 50$ to 0.43 at $k = 96$. Extremely highly coupled classes (CBO > 50) in Xerces are dominantly caused by internal reuse ($C_{50} = 0.71$ and $P_{50} = 0.67$) but the magnitude of in-degree dominance decreased for higher values of CBO. This means that Xerces contains a significant portion of extremely highly coupled classes that are either dominantly caused by internal aggregation or both aggregation and reuse significantly contribute to total coupling. The top ten most coupled classes in Xerces are shown in Table 3.27. As it can be observed the list contains:

- Two classes (XSDHandler and XMLSchemaValidator) whose coupling is dominantly caused by the internal aggregation of a large number of other classes.
- Four classes (XNIException, QName, SymbolTable and Constants) whose coupling is entirely or almost entirely caused by their excessive internal reuse.
- Four classes where both internal aggregation and internal reuse significantly contribute to total coupling.

In contrast to Xerces, the coupling of the top ten most coupled classes defined in Ant is either almost entirely (out-degree equal or close to zero) or strongly (in-degree $\gg$ out-degree) caused by internal reuse (see Table 3.28).

TABLE 3.27: The top ten most coupled classes in Xerces.

Class	In-degree	Out-degree	Total-degree (CBO)
org.apache.xerces.impl.xs.traversers.XSDHandler	15	91	106
org.apache.xerces.xni.XNIException	105	0	105
org.apache.xerces.impl.xs.XMLSchemaValidator	15	81	96
org.apache.xerces.util.SymbolTable	86	1	87
org.apache.xerces.xni.QName	86	0	86
org.apache.xerces.dom.CoreDocumentImpl	47	33	80
org.apache.xerces.impl.xs.SchemaGrammar	35	44	79
org.apache.wml.dom.WMLDocumentImpl	37	39	76
org.apache.xerces.impl.Constants	72	1	73
org.apache.xerces.impl.XMLEntityManager	28	40	68

TABLE 3.28: The top ten most coupled classes in Ant.

Class	In-degree	Out-degree	Total-degree (CBO)
org.apache.tools.ant.BuildException	533	1	534
org.apache.tools.ant.Project	445	34	479
org.apache.tools.ant.Task	180	8	188
org.apache.tools.ant.util.FileUtils	165	12	177
org.apache.tools.ant.Location	162	1	163
org.apache.tools.ant.types.Resource	131	5	136
org.apache.tools.ant.types.Path	111	16	127
org.apache.tools.ant.types.Commandline.Argument	81	4	85
org.apache.tools.ant.types.Commandline	79	5	84
org.apache.tools.ant.types.ResourceCollection	80	0	80

Our analysis of the relationship between in-degree and out-degree for highly coupled classes in five Java software systems showed the origin of their high coupling, which is considered as an indicator

of poor software design, is in extensive internal reuse, which is, to the contrary, considered desirable in software development practice. This seems to be a paradox. However, high coupling caused by extensive internal reuse can indicate only the negative aspects of extensive internal reuse (high criticality), not the negative aspects of extensive internal aggregation (understandability and error-proneness). In the case when highly reused classes tend to be simple (and thus problem-free) or when they are extensively tested (or validated) in early phases of software development and do not tend to cause problems during software evolution, then we can consider high coupling caused by extensive internal reuse as an indicator of good rather than poor modularization. In such situations, high coupling means low redundancy of code and proper abstraction of highly reused classes. However, in the case when a highly reused class tends to be unstable during software evolution, in the sense that its modification forces modifications in a large number of classes that depend on it, then we have to control its coupling/internal reuse and keep it as low as possible.

We also applied the metric-based comparison test in order examine differences between hubs and "ordinary" classes (the O set). The results are summarized in Table 3.29. As it can be observed for four software systems (all except Lucene) the null hypothesis of the Mann-Whitney U test is accepted only for the DIT (depth in inheritance tree) metric. This means that highly coupled classes tend to exhibit the same degree of specialization as loosely coupled classes. In all other aspects (voluminosity, internal complexity, degree of reuse and aggregation, centrality and importance) the differences between hubs and non-hubs are statistically significant. The drastic differences in all cases studies are present for the following metrics:

- LOC. Highly-coupled classes tend to be drastically more voluminous than ordinary classes. In all cases studies the average LOC of hubs (denoted by $\overline{C_1}$ in Table 3.29) is at least three times higher than the average LOC of ordinary classes (denoted by $\overline{C_2}$ in Table 3.29) and the probability that a randomly selected hub contains more lines of code than a randomly selected ordinary class is in the range $[0.77, 0.86]$.

- BET. Highly coupled classes tend to have drastically higher betweenness centrality than loosely coupled classes. In all cases studies the average BET of hubs is at least 13 times higher than the average BET of ordinary classes (for Ant 30 times higher, Tomcat 28 times higher). The probability that a randomly selected hub have higher BET than a randomly selected ordinary class is in the range $[0.76, 0.83]$. This means that hubs tend to occupy central positions in examined class collaboration networks indicating their vital role to the overall functionality of corresponding software projects.

- IN. Highly-coupled classes tend to have drastically higher in-degree than ordinary classes. However this result is expected since high coupling in examined software systems is dominantly caused by extensive internal reuse.

For Tomcat, Ant and JFreeChart, the drastic differences between hubs and ordinary classes can also be observed with respect to the NUMM (number of methods) metric. This means that hubs tend to define drastically more methods than ordinary classes. However, only for one system, Ant, there is drastic difference between the cyclomatic complexity of hubs and the cyclomatic complexity of ordinary classes. Having in mind that for all examined systems there is drastic difference with respect to the LOC metric we can conclude that hubs tend to drastically more voluminous than non-hubs, but not drastically more internally complex.

TABLE 3.29: The results of the metric-based comparison test for hubs.

Software system	Metric	$\overline{C_1}$	$\overline{C_2}$	U	p	NullHyp	PS_1	PS_2
Tomcat	LOC	699.03	137.29	204480	$< 10^{-4}$	rejected	**0.84**	0.15
	CC	71.47	13.69	176951	$< 10^{-4}$	rejected	0.68	0.22
	NUMA	12.18	3.89	178509	$< 10^{-4}$	rejected	0.7	0.22
	NUMM	31.68	7.21	206308	$< 10^{-4}$	rejected	**0.84**	0.13
	IN	20.36	2.35	198789	$< 10^{-4}$	rejected	**0.78**	0.14
	OUT	15.42	3.04	199959	$< 10^{-4}$	rejected	**0.8**	0.15
	NOC	1.25	0.18	136294	0.006	rejected	0.18	0.05
	DIT	0.63	0.47	126201	0.35	**accepted**	0.27	0.23
	BET	4777.76	169.75	208000	$< 10^{-4}$	rejected	**0.83**	0.11
	PR	0.002055	0.000474	179470	$< 10^{-4}$	rejected	0.74	0.26
Lucene	LOC	332.5	93.03	69044	$< 10^{-4}$	rejected	**0.78**	0.22
	CC	37.11	13.32	62822	$< 10^{-4}$	rejected	0.66	0.25
	NUMA	7.52	2.93	59358	$< 10^{-4}$	rejected	0.62	0.28
	NUMM	15.17	5.91	65950	$< 10^{-4}$	rejected	0.72	0.23
	IN	16.26	2.04	76210	$< 10^{-4}$	rejected	**0.82**	0.11
	OUT	10.05	3.33	64601	$< 10^{-4}$	rejected	0.69	0.24
	NOC	1.68	0.26	55865	$< 10^{-4}$	rejected	0.35	0.09
	DIT	0.46	0.88	53421	0.0002	rejected	0.2	0.4
	BET	2828.94	139.12	71292	$< 10^{-4}$	rejected	**0.76**	0.15
	PR	0.00347	0.000809	70588	$< 10^{-4}$	rejected	**0.79**	0.2
Ant	LOC	574.78	114.32	132568	$< 10^{-4}$	rejected	**0.86**	0.14
	CC	49.67	9.63	124300	$< 10^{-4}$	rejected	**0.78**	0.17
	NUMA	12.37	3.66	117880	$< 10^{-4}$	rejected	0.73	0.2
	NUMM	26.08	6.25	133043	$< 10^{-4}$	rejected	**0.85**	0.13
	IN	24.36	1.79	126563	$< 10^{-4}$	rejected	**0.78**	0.14
	OUT	13.30	3.42	129639	$< 10^{-4}$	rejected	**0.82**	0.14
	NOC	2.88	0.2	95863	$< 10^{-4}$	rejected	0.31	0.07
	DIT	1.28	1.11	83411	0.11	**accepted**	0.39	0.32
	BET	7715.53	249.41	129761	$< 10^{-4}$	rejected	**0.8**	0.12
	PR	0.003835	0.000411	117779	$< 10^{-4}$	rejected	**0.75**	0.23
Xerces	LOC	766.36	145.84	71757	$< 10^{-4}$	rejected	**0.79**	0.21
	CC	100.54	15.78	64222	$< 10^{-4}$	rejected	0.64	0.22
	NUMA	22.35	3.18	66545	$< 10^{-4}$	rejected	0.69	0.22
	NUMM	24.35	8.26	67912	$< 10^{-4}$	rejected	0.73	0.23
	IN	22.72	2.71	77836	$< 10^{-4}$	rejected	**0.83**	0.11
	OUT	17.67	3.51	66649	$< 10^{-4}$	rejected	0.7	0.23
	NOC	1.46	0.23	53140	0.003	rejected	0.26	0.09
	DIT	0.68	1.07	50199	0.06	**accepted**	0.23	0.33
	BET	3897.61	182.87	72166	$< 10^{-4}$	rejected	**0.76**	0.16
	PR	0.003367	0.000788	67360	$< 10^{-4}$	rejected	0.74	0.25
JFreeChart	LOC	854.12	202.85	37753	$< 10^{-4}$	rejected	**0.77**	0.23
	CC	59.83	12.32	36467	$< 10^{-4}$	rejected	0.71	0.22
	NUMA	11.97	3.35	33972	$< 10^{-4}$	rejected	0.65	0.27
	NUMM	37.25	10.02	38177	$< 10^{-4}$	rejected	**0.77**	0.21
	IN	21.45	2.34	39365	$< 10^{-4}$	rejected	**0.76**	0.16
	OUT	14.38	3.56	37118	$< 10^{-4}$	rejected	0.72	0.21
	NOC	1.77	0.23	31594	$< 10^{-4}$	rejected	0.35	0.06
	DIT	1	0.83	27197	0.08	**accepted**	0.39	0.27
	BET	931.58	67.69	39678	$< 10^{-4}$	rejected	**0.77**	0.14
	PR	0.004705	0.001066	35461	$< 10^{-4}$	rejected	0.71	0.26

3.6 Summary and future work

Real-world software systems are characterized by complex interactions among source code entities. Those interactions can be modeled in terms of software networks which show dependencies between software entities. In order to understand the complexity of dependency structures of software systems, to compute metrics associated with software design, or to recover system architecture from the source code, networks representing software systems have to be extracted. In this thesis we have presented SNEIPL, the language-independent software networks extractor based on the enriched Concrete Syntax Tree (eCST) representation of the source code. The eCST representation extends parse trees with so called universal nodes which are predefined semantic markers of syntactical constructions. The set of eCST universal nodes contains nodes that mark definitions of software entities which appear as nodes in software networks, as well as universal nodes such as that serve as the starting point to recover horizontal dependencies between software entities. From the hierarchy of universal nodes in eCSTs different types of horizontal dependencies can be deduced, which means that SNEIPL is able to extract software networks at different levels of abstraction.

The applicability of SNEIPL was shown by the extraction of software networks associated with real-world, medium to large-scale software systems written in different programming languages (Java, Modula-2, and Delphi). To investigate the correctness and completeness of the extraction algorithm, we compared class collaboration networks extracted from ten Java software systems with networks extracted using Dependency Finder (language-dependent software networks extractor) and Doxygen (language-independent documentation generator tool). Obtained results showed that networks extracted by SNEIPL and Dependency Finder are highly similar, and that the eCST-based approach to language-independent extraction of dependencies provides far more precise results compared to the unified fuzzy parsing approach realized by Doxygen. Since SNEIPL operates on the language-independent representation of the source code, this result can be generalized to networks representing software systems written in other languages.

In contrast to other widely used language-independent reverse engineering tools and frameworks, SNEIPL provides both language-independent fact extraction and language-independent representation of extracted facts. This means that besides language-independent network based analysis of software systems and language-independent computation of software design metrics, SNEIPL can be used to provide language-independent extraction of fact bases for reverse engineering, architecture recovery, and software comprehension activities.

In this thesis we introduced the idea of applying graph clustering evaluation (GCE) metrics to graphs representing software systems in order to evaluate cohesiveness of software entities. In contrast to standard cohesion metrics, GCE metrics do not ignore external references. They are based on the idea that reducing coupling between an entity and the rest of the system increases cohesion of the elements contained in the entity. Using the theoretical framework introduced by Briand et al. [1996, 1998] we investigated the properties of graph clustering evaluation metrics. This analysis showed that GCE metrics are theoretically sound with respect to the most important properties of software cohesion metrics (monotonicity and merge property), but also showed that they possess certain limitations we should be aware of when using GCE metrics as software metrics. Our future work will extend the present work with an empirical investigation of the following research questions:

1. Do GCE metrics correlate to standard software cohesion metrics (LCOMs, TCC, LCC, etc.) and to what extent?

2. Each software entity can be described by a numerical vector containing metrics of internal complexity and metric of design complexity. Each of these vectors can be, according to the degree of cohesion, classified as Radicchi strong (strongly cohesive), Radicchi weak (weakly cohesive) or poorly cohesive (entity that is neither Radicchi strong nor Radicchi weak). Therefore, our second research question will be: are there any differences in internal and design complexity between strongly, weakly and poorly cohesive software entities?

3. Possibility to automatically remodularize software system using simple refactorings such as move class/method in order to improve the degree of cohesion of the overall system (to increase the number of Raddichi strong clusters, to minimize the average conductance, etc.).

In this thesis we analyzed class collaboration networks representing five large-scale software systems written in Java (Tomcat, Lucene, Ant, Xerces and JFreeChart). The networks were extracted using SNEIPL. One of the distinctive characteristic of our analysis is that each node in the network was described by a metric vector that contains both domain-dependent (software metrics) and domain-independent metrics (metrics used in analysis in complex networks and defined on any directed graph). Moreover, we introduced the metric-based comparison test based on the Mann-Whitney U test to examine characteristics of a certain subset of nodes in the networks.

Analysis of connected components showed that each analyzed network has a giant weakly connected component (GWCC) that exhibit the small-world property in the Watts-Strogatz sense and a weak disassortative mixing. Moreover, we observed that there are large cyclic dependencies among classes in GWCCs reflected by the existence of a relatively large strongly connected components (SCCs). The analysis of SCCs showed that there is a strong Spearman correlation between the size of SCC and the average intra-component degree which means that SCCs tend to densify with size. The densification of SCCs can be modeled by power-laws whose scaling exponents can be used as indicators of design quality. The application of the metric-based comparison test showed that in two software systems (Ant and JFreeChart) there is a strong tendency that strongly connected components encompasses the most central and the most important classes.

In contrast to similar studies, degree distributions of giant weakly connected components are tested against power law, exponential and log-normal probability mass functions using the power-law test introduced by Clauset et al. [2009]. Such analysis revealed that the tails of the empirically observed degree distributions can be described by power-laws but alternative distributions are either equally plausible or even provide better fits. Moreover, log-normal distribution provides better fits to the distributions considering the whole range of degree values compared to power-law indicating that the nearly-linear preferential attachment governs the evolution examined class collaboration networks.

Due to heavy-tailed degree distributions all examined networks contains hubs – highly connected nodes whose in-, out- and total-degree (CBO) are high above the average values. For extremely highly coupled classes we showed that there is the disbalance between in-degree and out-degree where in-degree strongly dominates over out-degree. Moreover, for four of examined systems (all except Xerces) the extent of in-degree domination over out-degree increases with CBO. This result implies that extremely highly coupled classes in real software systems are caused dominantly by internal reuse and consequently that high coupling can indicate only negative aspects of internal reuse, not negative aspect of internal aggregation. The application of the metric-based comparison test showed that highly coupled classes in all examined systems tend to be drastically more voluminous, internally reused and centrally positioned in the networks compared to the rest of classes. On the other side, in

the majority of examined systems (all except Lucene) highly coupled classes do not tend to be more or less specialized than lowly coupled classes.

In this thesis we investigated the structure of class collaboration network focusing on the properties of connected components, degree distributions and hubs. In our future work we will use the same methodological framework to investigate the structure of software networks at different levels of abstraction such as package and method collaboration networks. We will also examine the evolution of software networks to investigate how weakly and especially strongly connected component evolve. The evolutionary analysis can also be focused on the properties of hubs. From the evolutionary perspective, the set of nodes in a class collaboration network can be divided into two disjoint subsets according to the following two binary classifiers:

1. If node is referenced by one or more newly created nodes then it belongs to the positive class, otherwise it is in the negative class.

2. If node references one or more newly created nodes then it belongs to the positive class, otherwise it is in the negative class.

Therefore, the application of the metric-based comparison test in the evolutionary setting (a sequence of networks corresponding to different versions) can additionally reveal how new nodes (classes) integrate into the network as corresponding software evolves.

Chapter 4

Ontology networks

Ontology networks describe the structure of ontological entities contained in an ontological description. This chapter of the dissertation is devoted to the extraction and analysis of ontology networks. Necessary preliminaries and definitions are given in Section 4.1. The next Section 4.2 presents an overview of existing ontology design metrics. In the previous chapter of the dissertation we argued that graph clustering evaluation (GCE) metrics can be used to evaluate cohesiveness of software modules. In Section 4.3 we expand that idea showing that GCE metrics can also be viewed as ontology metrics. The extraction of ontology networks is discussed in Section 4.4. In the same Section we present a novel approach to the extraction of ontology networks whose nodes are attributed with a rich set of metrics. In Section 4.5, after giving an overview of previous research works related to analysis of ontology networks, we present results of the analysis of ontology networks representing a real-world, widely used modularized ontology. Finally, Section 4.6 summarizes the chapter and gives possible directions for future work.

4.1 Preliminaries and definitions

The Web Ontology Language (OWL) is an ontology language for semantic web applications that is designed and recommended by W3C (World Wide Web Consortium). The latest version of the language is called OWL2 and dates from December 2012. OWL extends capabilities of the Resource Description Framework (RDF) which is a language for describing web accessible resources and relationships between them. OWL is a knowledge representation language based on description logic. This means that knowledge expressed in OWL is amenable to decision procedures that can be used to check its consistency and derive implicit knowledge.

OWL2 ontologies and ontological entities are identified using Internationalized Resource Identifiers (IRIs) which have a global scope. Ontological entities can be classified as:

- Classes (concepts) - sets of objects,
- Objects (individuals) - instances of classes,
- Data types - sets of data values (literals) such as strings and numbers,
- Object properties (roles) - relations between objects,
- Data properties (attributes) - relations between objects and literals,

- Annotation properties - relations between ontological entities and annotation values (IRIs, literals and anonymous objects – objects that do not have IRI).

We can distinguish between two drastically different types of ontology design: monolithic and modularized. The content of a monolithic ontology is given in a single document (one owl file). The OWL2 language contains import construct which enables ontology modularization. Ontology A can import ontology B in order to gain access to all ontological entities and axioms present in B. Ontologies present in a modularized ontology we also call *ontology modules* (or just modules). There are many advantages of the modularized approach to ontology engineering including easier reuse, better reasoning performance, efficient ontology management and change handling. Similarly as for software systems, the principle of low coupling and high cohesion of ontology modules emphasizes a good modularization.

Every OWL2 ontology besides import statements and annotations (human-readable descriptions/-comments) also contains a set of axioms that can be divided into the following categories:

- Declaration axioms state the existence of ontological entities and associate them to particular types.
- Class axioms which describe relations (equivalence, subsumption and disjointness) between classes/class expressions. A class expression (complex concept, class description, anonymous class or class constructor) describes a set of objects relying on existing classes, objects, object and data properties through the usage of set operations, enumerations and restrictions.
- Object property axioms describe relations between object properties, relations between object properties and classes, or characterize object properties (specify whether an object property is reflexive, symmetric, transitive or functional). An object property is considered as functional if it connects an object to at most one other object.
- Data property axioms describe relations between data properties, relations between data properties and classes, relations between data properties and data types or characterize data properties as functional.
- Assertion axioms, also called *facts*, describe relations between objects (same object, different objects), state that two objects are associated by an object property, associate objects and literals by data properties and state that an object is instance of a particular class.
- Annotation axioms describe relations between annotation properties and associations between annotation properties and other ontological entities.
- Data type definition axioms maps data types to data ranges.

The first dilemma when constructing an ontology graph is whether it should represent explicitly stated relations between ontological entities or relations obtained after ontology normalization. Ontology normalization means that ontological inference was previously applied and new axioms are derived from explicitly stated axioms. The issue of ontology normalization is especially discussed in the literature related to ontology metrics [Vrandečić and Sure, 2007]. Namely, we can distinguish between two types of ontology metrics:

- Semantic-aware or semantic stable metrics which perform ontology normalization,

- Structural metrics that are computed on the graph representation of an ontology obtained without normalization or where only certain normalization steps are performed.

The fact that two different onotological descriptions expressing the same knowledge have different ontology graph representations is commonly used as an argument for the "superiority" of semantic stable metrics [Ma et al., 2011]. However, the main point here is that semantic stable metric and structural metrics measure two different things: the former measure complexity and quality aspects of the knowledge inferred from an ontological description, while the latter reflect the complexity or quality of the description itself (explicitly stated knowledge). As pointed out by Zhang et al. [2010] ontology normalization may drastically change the shape of an ontological description which means that the normalized ontology graph may not faithfully represent explicitly stated knowledge and consequently semantic stable metric may not faithfully express its complexity or quality.

Another relevant point not discussed in the literature is that semantic stable ontology metrics can be "unstable" with respect to the predefined modularization of an ontology. The essence of ontology normalization is the materialization of subsumption hierarchies [Ma et al., 2011; Vrandečić and Sure, 2007], i.e. the identification and removal of cyclic dependencies in a hierarchy of concepts. Here we provide a simple example which demonstrates how this normalization step "breaks" predefined modularization. Let us suppose that we have a modularized ontology that consists of two ontology modules named A and B. The descriptions of A and B are as follows:

```
(Ontology <A>
   Import(<B>)
   SubClassOf(<A#a> <B#b>)
)

(Ontology <B>
   SubClassOf(<B#b> <A#a>))
)
```

From the semantic point of view ontology A "knows" that classes a and b are equivalent. On the other side, ontology B "knows" just that b is subclass of a because B does not import A, and consequently the SubClassOf axiom contained in A is not visible in B and cannot be used to infer the equivalency of a and b. If we normalize the subsumption hierarchy then the cyclic subsumption dependency between a and b will be detected, both SubClassOf axioms will be removed, and a new axiom EquivalentClass($A\#a$, $B\#b$) will be stated. Obviously the newly introduced axiom is out of the predefined modular structure of the whole ontology: who states ("owns") the axiom that is obtained by the normalization, ontology A or ontology B? If we put the new axiom in ontology A then the knowledge encapsulated in A remains unchanged, but then ontology B does not provide any knowledge. If we put the axiom in B then again the knowledge in A remains unchanged, but then B contains more knowledge than before the normalization. Generally speaking, each normalization that affects entities from different ontology modules can introduce inconsistencies in the modular structure of the whole ontology, i.e. the structure has to be remodularized in order be consistent with the effect of the normalization. If we apply remodularization to predefined design of modules produced by knowledge engineers then we are unable to measure and evaluate the quality of their work. Therefore, in this dissertation we consider minimal semantic normalizations which preserve

predefined modularization. Namely, we only replace complex class expressions by anonymous classes and assemble domain and range axioms into associations.

Having in mind the previous discussion we define an ontology graph as follows.

Definition 4.1 (**Ontology graph).** Ontology graph G representing modularized ontology M is a directed, typed graph where both nodes and links have types. The nodes of G represent different types of ontological entities: ontologies, classes, objects, object properties, annotation properties, data properties, data types and literals. There are seven types of links in G:

- SUB links represent subsumption relations between two ontological entities of the same type. They can connect classes (as stated by SubClassOf axioms), object properties (as stated by SubObjectPropertyOf axioms), data properties (as stated by SubDataPropertyOf axioms) and annotation properties (as stated by SubAnnotationPropertyOf axioms). In other words the subgraph of G induced by SUB links represents hierarchies (taxonomies) present in M.

- ASSERTION links represent associations between ontological entities that are induced by assertion axioms.

- EQUIVALENT links denote that two ontological entities are equivalent. Those links can be established between two classes (as stated by EquivalentClasses axioms), two objects (as stated by SameIndividuals axioms) and two object properties (as stated by EquivalentObjectProperties axioms). EQUIVALENT links are reciprocal, i.e. if a points to b by an EQUIVALENT link then b also points to a by another EQUIVALENT link.

- DISJOINT links denote that two ontological entities are not equivalent. Similarly as EQUIVALENT links they connect two classes, two objects or two object properties, and they are reciprocal.

- REFERENCES links connect anonymous classes with named classes. Let e be a complex class expression. Then e is represented by an anonymous class A in G. A is connected by REFERENCES links to all named classes involved in e.

- CONTAINS links associate ontology modules to other ontological entities. Contains link $O \rightarrow e$ states that ontological entity e is part of ontology module O.

- IMPORTS links denote dependencies between ontology modules. Two ontologies A and B are connected by the IMPORT link $A \rightarrow B$ if and only if ontology A imports ontology B.

- Links that represent user-defined associations among classes determined by relevant pairs of ObjectPropertyDomain and ObjectPropertyRange axioms. This category of links also includes user-defined association stated by assertion axioms involving object, data and annotation properties.

OWL ontologies are primarily exchanged as RDF documents. A RDF description is a sequence of triplets which specify associations between pairs of terms. Therefore, every OWL ontology can be mapped to an appropriate RDF graph [Peter F. Patel-Schneider and Bors Motik (Editors)]. However, it should be emphasized that the ontology graph representation is not same as the RDF graph representation. The first difference is that user-defined associations (concrete object, data and annotation properties) in ontology graph exist as both nodes and links, while in the RDF graph representation

they exist only as nodes. This difference between the RDF graph representation and ontology graph representation of a simple ontology is illustrated in Figure 4.1. Another difference is that nodes of RDF graphs do not have type, but type of nodes is modeled using predefined nodes and links. For example, RDF nodes representing concepts are connected to the node "owl:Class" by links whose type is "rdf:type". Finally, nodes and links of RDF graphs represent also axioms that do not state associations among ontological entities but their individual characteristics (e.g. transitivity of a property).

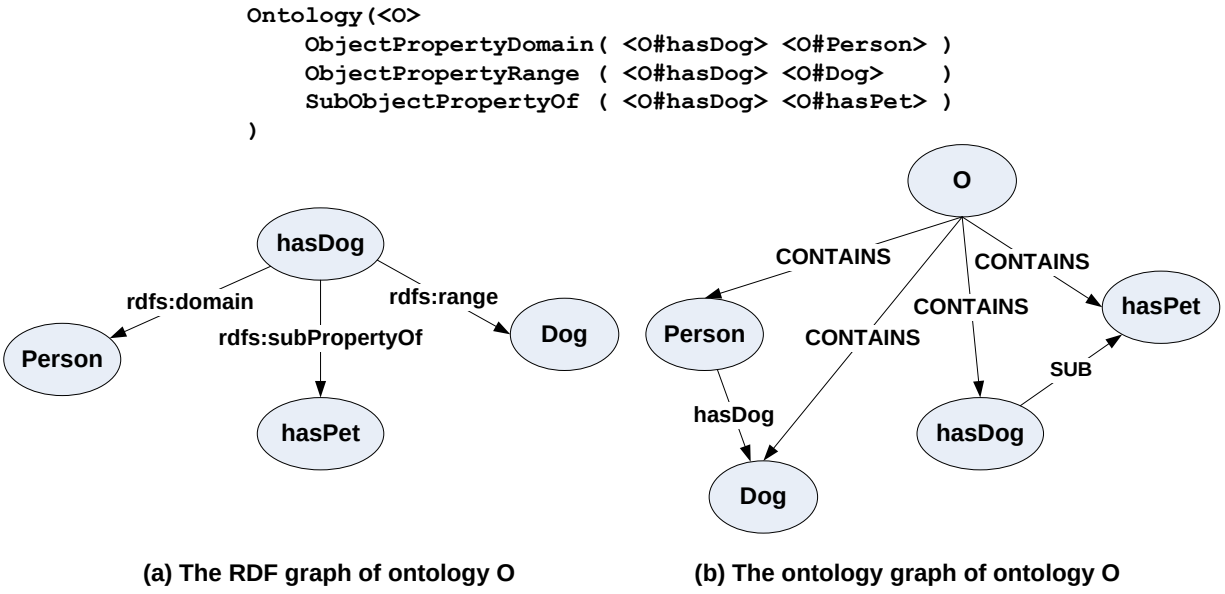


FIGURE 4.1: Simple ontology O with its RDF and ontology graph representations.

Class expressions are represented by anonymous classes in ontology graph. An example is given in Figure 4.2. Ontology O contains just one axiom which states that two classes are equivalent. This axiom has class expression as the second argument (ObjectIntersectionOf) that is represented by "AnonymousClass" node in the ontology graph. "AnoymousClass" references all named classes that are part of the class expression (classes Child and Man). Therefore, there is an indirect coupling between the first argument of the axiom (class Boy) and the classes that are involved in the class expression. The ObjectIntersetionOf class expression is contained in ontology O and consequently there is CONTAINS link between O and the node that represents the class expression.

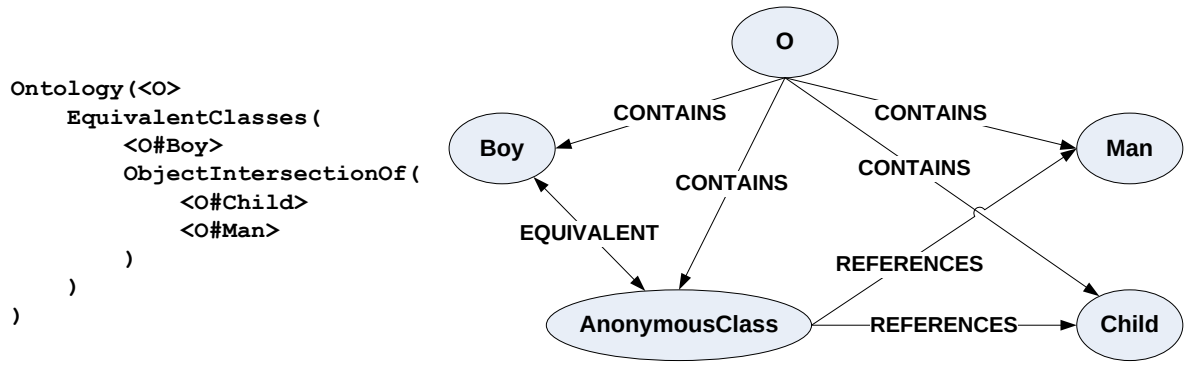


FIGURE 4.2: Normalization of complex class expressions.

From ontology graph a variety of ontology networks can be formed.

Definition 4.2 (**Ontology module network, OMN).** The ontology module network of a modularized ontology O is a subgraph of the ontology graph that contains all nodes representing ontologies (modules) and existing IMPORTS links.

Definition 4.3 (**Ontology class network, OCN).** The ontology class network of a (modularized) ontology O is a subgraph of the ontology graph induced by nodes representing classes (including anonymous classes too) without restrictions on link types. In other words, the ontology class network shows dependencies between concepts present in the domain described by O.

Definition 4.4 (**Ontology subsumption network, OSN).** The ontology subsumption network of a (modularized) ontology O is a subgraph of the ontology graph induced by classes (including anonymous classes too) and SUB links. This network represent taxonomy of concepts present in the ontology which is commonly considered as its backbone.

Definition 4.5 (**Ontology object network, OON).** The ontology object network of a (modularized) ontology O is a subgraph of the ontology graph induced by objects without restrictions on link types. In other words, this network shows associations among concrete objects from the domain.

There is one important difference between ontology object networks and other above defined ontology networks. An OON describes a particular state of affairs in a domain. Other networks are oriented toward the knowledge about the domain: OCN and OSN depict the structure of conceptualization, while OMN describes the structure of knowledge modularization.

4.2 Ontology metrics

Ontology networks are closely related to ontology metrics that are used to evaluate various aspects of ontology design. Several metrics based on the ontology graph representation (or some specific ontology network that is a sub-graph of ontology graph) were proposed in recent years in order to evaluate the design complexity and quality of ontological descriptions.

Inspired by the Chidamber-Kemerer object-oriented metric suite Yao et al. [2005] proposed three ontology cohesion metrics. An ontology has a high cohesion value if its entities are strongly related. The authors emphasized that the backbone of every ontology is the taxonomy of concepts present in the domain. Therefore, they defined three metrics that quantify the complexity of ontology subsumption networks (see Definition 4.4):

- The number of root classes. A root class has no super-class which means that it is not a specialization of any other concept present in the domain. Thus, the number of root classes is the number of nodes in the ontology subsumption network whose out-degree is equal to zero.
- The number of leaf classes. A leaf class has no sub-class which means that there is no concept which is its specialization. Thus, the number of leaf classes is the number of nodes in the ontology subsumption network whose in-degree is equal to zero.
- The average depth of inheritance tree of leaf nodes. Each leaf class is directly or indirectly connected to one or more root classes. Therefore this metric represent the average distance from a leaf class to the furthest root class.

As it can be observed from the definitions of metrics their small values indicate more compact and consequently more cohesive taxonomy of concepts. However, those metrics cannot be used to evaluate

cohesiveness of ontology modules in a modularized ontology since the taxonomy of concepts transcends the modularization of the ontology.

Tartir et al. [2005] introduced OntoQA which is perhaps the most voluminous metric suite for ontology quality evaluation. The metrics from OntoQA can be classified either as schema or instance metrics. Schema metric indicate the richness of an ontology schema. Ontology schema is a part of ontology devoted to concepts present in the domain. Three metrics from OntoQA belong to this category of metrics:

- *relationship richness* that reflects the diversity of relations in the ontology,
- *attribute richness* which is the average number of attributes (data properties) per class,
- *inheritance richness* which is the average number of subclasses per class.

On the other hand, instance metrics quantify the effectiveness of ontology design and the amount of facts contained in the ontology. The metrics from this category can be further divided into two sub-categories: knowledge-base metrics and class metrics. Knowledge-base metrics from OntoQA are:

- *class richness* – the fraction of classes that have instances,
- *average population* – the average number of instances per class,
- *cohesion of instances* – the number of weakly connected components in ontology object network (see Definition 4.5).

Class metric introduced in OntoQA are:

- *importance* – the fraction of instances that belong to the sub-tree rooted at a class in the ontology subsumption network (see Definition 4.4),
- *inheritance richness* – the average number of subclasses per class in the sub-tree rooted at a class in the ontology subsumption network,
- *relationship richness* that reflect the actual utilization of user-defined associations,
- *connectivity* – the number of instances of other classes connected to instances of a class.

As it can be observed all ontological metrics from the OntoQA metrics suite can be easily obtained from the ontology graph or specific ontology network (ontology subsumption network and ontology object network).

Orme et al. [2006] proposed three ontology coupling metrics that quantify the strength of efferent coupling. Those metric are the number of external classes (NEC), reference to external classes (REC), and referenced includes (RI). Let O be an ontology or ontology module in a modularized ontology. Let C denote the set of classes used in O. This set can be divided into two disjoint subsets C_i and C_o where C_i are classes defined in O and C_o classes defined outside of O. Then NEC of O is the cardinality of C_o and REC of O is the number of links connecting classes from C_i to classes from C_o. Both metric can be computed from the ontology graph of O since the graph has CONTAINS links which determine the ontology that define a concept. RI is the number of import statements in O which is the out-degree of the node representing O in the ontology module network.

Zhang et al. [2010] proposed a metric suite for measuring design complexity of ontologies that includes both ontology-level and class-level metrics. The ontology-level metrics introduced in the suite are:

- *The size of vocabulary* of an ontology which is the number of named entities in the ontology. In other words this metric is equivalent to the number of nodes representing named classes, individuals and user-defined associations in a ontology graph.
- *Edge node ratio* which is the ratio of the number of links to the number of nodes in ontology graph.
- *Tree impurity* which measures the extent to which an ontology subsumption network deviates from being a tree.
- *Entropy of graph* which measures the regularity of degree sequence of an ontology graph – the minimal value of this metric is achieved when all nodes in the ontology graph have the same total degree.

Two of four class-level metrics in the suite are adoption of object-oriented software metrics introduced in the Chidamber-Kemerer suite. NOC (number of children) and DIT (depth in inheritance tree) computed on ontology subsumption network can be used to quantify reuse of concepts through inheritance and the degree of specialization of concepts, respectively. Other two metrics are in-degree and out-degree of a class in the ontology class network (see Definition 4.3) which reflect afferent and efferent coupling of the class, respectively.

Žontar and Heričko [2012] analyzed software metrics from the Lorenz-Kidd, Chidamber-Kemerer and Abreu metric suites in order to determine which of them can be adopted for ontologies. The results of their study showed that graph-based software metrics (design software metrics) can be adopted for ontology evaluation.

Oh et al. [2011] proposed a metric suite for evaluating coupling and cohesion of modularized ontologies. The authors introduced one cohesion metric and two coupling metrics. Following the idea that the hierarchy of concepts in the backbone of an ontology, the authors made the explicit distinction between hierarchical relations determined by subsumption axioms and non-hierarchical relations. The coupling metrics in the suite are the number of separated hierarchical relations and the number of separated non-hierarchical relations in a module. A relation is considered as separated if it relates concepts from different ontology modules. On the other side, cohesion of a module is measured as the sum of strength of relations among concepts in the module normalized by the number of all possible relations. The strength between two concepts is inversely proportional to the distance between them in ontology class network.

4.3 Graph clustering evaluation metrics as ontology metrics

In the previous chapter on software networks we presented the idea of using graph clustering evaluation metrics as software metrics. The same idea can be also applied for ontologies: graph clustering evaluation metrics can be used to evaluate cohesiveness of modules in a modularized ontology. We can distinguish between two categories of horizontal links in an ontology graph: inter-module and intra-module links. Intra-module link $A \to B$ connects ontological entities belonging to the same ontology:

$$\text{Intra-link}(A \to B) \Leftrightarrow (\exists O)\ \text{CONTAINS}(O \to A) \wedge \text{CONTAINS}(O \to B).$$

Inter-module link associates ontological entities that belong to different ontologies:

$$\text{Inter-link}(A \to B) \Leftrightarrow (\exists O_1, O_2)\ O_1 \neq O_2 \wedge \text{CONTAINS}(O_1 \to A) \wedge \text{CONTAINS}(O_2 \to B).$$

For example, SUB link that connects class B from ontology P to class C from ontology Q is the only inter-module link for the modularized ontology shown in Figure 4.3, while the other two SUB links are intra-module links.

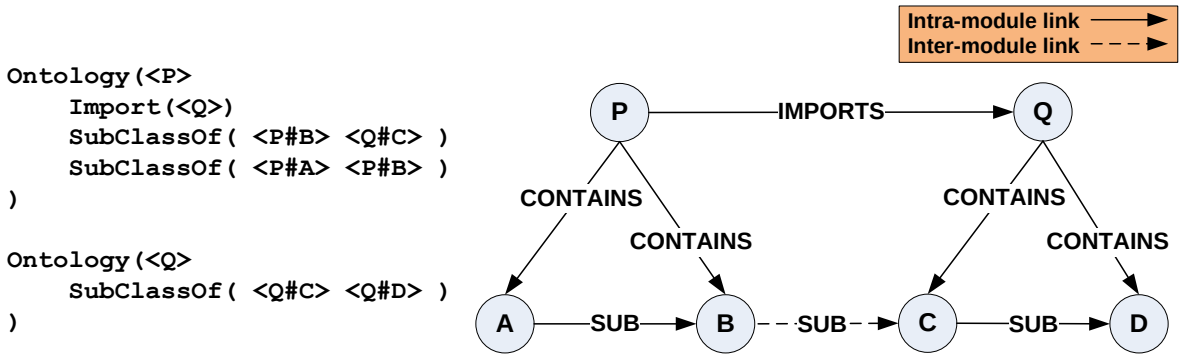


FIGURE 4.3: Ontology graph of a simple modularized ontology.

The distinction between intra-module and inter-module links enables us to compute

- Cut-based graph clustering evaluation metrics (conductance, expansion and cut-ratio). The edge cut of a module constitute all inter-module links emanating from ontological entities contained in the module.
- ODF (out-degree fraction) family of cluster quality measures since for each ontological entity we are able measure intra-module and inter-module out-degrees. Consequently, each ontology module can be classified either as Radicchi strong, Radicchi weak and poorly cohesive cluster (cluster that is neither Radicchi strong nor Radicchi weak). This is another advantage of GCE metrics since they have well defined thresholds that enable us to classify modules according to their degree of cohesion.

4.4 Extraction of ontology networks

Similarly to computer programs, ontological descriptions are given in a formal language. Therefore, ontology networks can be obtained by parsing ontological descriptions. However, ontology networks are much more easier to extract compared to software networks. Firstly, ontology languages are declarative and much easier to parse than imperative programming languages. Secondly, named ontological entities are "scope-free" in the sense that they are uniquely and globally determined by their IRIs. In other words, each identifier introduced in an ontology description always denote the same thing/resource and there are no name "conflicts" characteristic to imperative programming languages where one symbol may denote different things in different scopes. Thirdly and most importantly, the axioms given in an ontological description explicitly state relations between ontological entities where the type of an relation is determined by the type of the corresponding axiom. Moreover, the type of the axiom itself determines types of ontological entities contained in the axiom. For example, in the axiom `SubClassOf(:A :B)` we know that both A and B are ontological entities representing classes.

Ontology networks can be easily extracted using existing OWL libraries such as Apache Jena [Carroll et al., 2004; McBride, 2002] or OWL API [Bechhofer et al., 2003; Horridge and Bechhofer, 2011] which provide classes and interfaces that can be utilized to walk through the structure of an ontology. Moreover, mentioned libraries support different OWL syntaxes. For example, Listing 1 shows

the source code of full working extractor that is able to form ontology subsumption networks (see Definition 4.4) of monolithic onotologies. The extractor is based on the OWL API library and uses the JUNG (Java Universal Graph) library for graph-based operations.

Listing 1. Extractor of class subsumption networks for monolithic ontologies realized using the OWL API and JUNG libraries.

```
import java.util.Iterator;
import org.semanticweb.owlapi.model.*;
import edu.uci.ics.jung.graph.*;

public class ExtractSubsumptionGraph {
    private OWLOntology ontology;
    private int anonymousCounter;
    private DirectedSparseGraph<String, String> graph;

    public ExtractSubsumptionGraph(OWLOntology ontology) {
        this.ontology = ontology;
        anonymousCounter = 0;
    }

    public DirectedSparseGraph<String, String> extract() {
        graph = new DirectedSparseGraph<String, String>();
        Iterator<OWLAxiom> io = ontology.getAxioms().iterator();
        while (io.hasNext()) {
            OWLAxiom axiom = io.next();
            if (axiom.getAxiomType() == AxiomType.SUBCLASS_OF) {
                OWLSubClassOfAxiom scAxiom = (OWLSubClassOfAxiom) axiom;
                OWLClassExpression subClass = scAxiom.getSubClass();
                OWLClassExpression superClass = scAxiom.getSuperClass();
                String src = createNode(subClass);
                String dst = createNode(superClass);
                String linkName = "Subsumption_" + src + " --> " + dst;
                graph.addEdge(linkName, src, dst);
            }
        }
        return graph;
    }

    private String createNode(OWLClassExpression classExpression) {
        String src = null;
        if (classExpression.isAnonymous()) {
            src = "anonymous_" + anonymousCounter++;
        } else {
            src = classExpression.asOWLClass().getIRI().toString();
        }

        if (!graph.containsVertex(src)) {
            graph.addVertex(src);
        }

        return src;
    }
}
```

In this thesis as one of the original contributions we will present a tool called ONGRAM (**ON**tology

GRaphs **A**nd **M**etrics). This tool is able to extract ontology networks and compute different types of metrics reflecting complexity of modularized ontological descriptions. The specificity of the tool is that it is realized as one of the back-ends of the SSQSA framework [Budimac et al., 2012] after SSQSA was extended to support the OWL2 language in the functional-style syntax [Savić et al., 2013]. In the rest of the chapter it will be always assumed the functional-style syntax when referencing to the OWL2 language if some other syntax is not explicitly mentioned. The extension of SSQSA with OWL2 provides the enriched Concrete Syntax Tree (eCST) representation [Budimac et al., 2012; Rakić and Budimac, 2011a] of semantic web ontologies. The main benefit of the extension is that the eCST representation enables us to define new and adopt existing software metrics that reflect lexical and syntactical complexity of ontological descriptions. From the eCST representation ONGRAM extracts underlying graph representation of a modularized ontology and use it to compute graph-based ontology metrics. Therefore, ONGRAM provides an ontology metric suite that encompasses both metrics of internal and design complexity of individual ontologies contained in a modularized ontology.

4.4.1 Integration of OWL2 into SSQSA

In order to integrate OWL2 into the SSQSA framework the SSQSA front-end has to be extended to support OWL2 language. The SSQSA front-end also known under the term eCST generator instantiates ANTLR based parsers which produce the eCST representation. In order to integrate a new language in the SSQSA framework, an ANTLR grammar of that language has to be made. The integration of OWL2 into SSQSA consisted of the following three steps:

1. Realization of ANTLR grammar which describes the OWL2 language,
2. Identification of OWL2 language constructs that corresponds to existing eCST universal nodes,
3. Incorporation of eCST universal nodes into tree-rewrite rules of the grammar in order to obtain eCST representation of parsed text.

Step 1. The formal specification of OWL2 FSS in the Extended Backus-Naur form (EBNF) can be found in the official W3C OWL2 language specification [Boris Motik, Peter F. Patel-Schneider and Bijan Parsia (Editors)]. The ANTLR grammar notation closely follows EBNF, thus the grammar given in [Boris Motik, Peter F. Patel-Schneider and Bijan Parsia (Editors)] can be easily adopted for ANTLR. At this stage of the integration, the realized grammar is tested using ten ontologies from the TONES[1] repository which are previously converted into OWL2 FSS using Protégé[2]. The results are summarized in Table 4.1. It can be seen that the parser generated from the grammar successfully parsed more than 1.4 millions of lines of real-world ontological axioms in less than three minutes.

Step 2. The OWL2 language contains four types of tokens: keywords, separators, identifiers (IRIs) and constants. For each of mentioned categories there are lexical-level eCST universal nodes (KEYWORD, SEPARATOR, NAME and CONSTANT). Axioms are marked with the STMT eCST universal node which is used to mark statements in imperative programming languages. Elements of an axiom are also marked with eCST universal nodes. The ARGUMENT_LIST universal node is used to mark the list of arguments in an axiom, while the ARGUMENT universal node stands for one argument in the argument list. The TYPE eCST universal node is used to denote the type of ontological entity that is referenced in the axiom. Class and data range expressions are marked with

[1] http://owl.cs.manchester.ac.uk/repository/
[2] http://protege.stanford.edu/

Table 4.1: The results of testing of the OWL2 grammar testing using ontologies from the TONES repository.

Ontology	LOC	Parse time [sec]
CTON (Cell Type Ontology)	144252	23
FMA (Foundational Model of Anatomy)	316101	41
Gene Ontology Edit	233608	22
Human Disease	476111	59
Teleost Taxonomy	182656	17
GEO Skills	20506	2
Matr Mineral	46	0.04
OBO Relation Ontology	25412	2
SC Ontology	23707	2
Software Ontology	5931	0.5

the EXPR universal node which is in imperative programming languages used to mark expressions. Import statements are marked with the IMPORT_DECL universal node.

Declarations of ontological entities are also marked with eCST universal nodes. The COMPILATION_UNIT universal node encompasses the whole ontology document, i.e. this node is the root node in the eCST representation of ontology. The PACKAGE_DECL universal node denotes that entities declared in an eCST sub-tree rooted at this node are mutually visible. Therefore, PACKAGE_DECL corresponds to the declaration of ontology. The CONCRETE_UNIT_DECL universal node is used to denote definitions of classes in object-oriented (OO) programming languages. However, there is a big difference between declarations of ontological classes and definitions of OO classes. When the existence of an OO class is stated in the source code then its structure (attributes and methods) is specified. On the other hand, ontological classes are atomic entities, i.e. entities that do not have internal structure. Therefore, declarations of ontology classes are marked with the ATTRIBUTE_DECL universal node which is used to denote declarations of global variables in imperative programming languages. With the same node we also marked other types of ontological declarations: data type declarations, object, data and annotation property declarations, and declarations of named individuals.

OWL2 is a declarative, domain-specific language. Before the integration of OWL2, SSQSA supported several programming languages none of them being declarative or domain-specific. OWL2 axioms represent explicitly stated associations among ontological entities. Therefore, we introduced three new universal nodes that denote different types of explicitly stated relations in general:

1. BINARY_RELATION (BR) mark binary associations,
2. SYMMETRIC_RELATION (SR) mark symmetric n-ary relations,
3. PARTIALLY_KNOWN_BINARY_RELATION (PKBR) mark binary associations in which one of the argument is not known at the moment, i.e. it is not specified in the axiom marked by PKBR. If the first argument of PKBR is known then it is marked with the SRC universal node. The DST universal node denotes that the second argument of association is present. Both SRC and DST are also newly introduced universal nodes.

With BR are marked OWL2 axioms that state subsumptions and assertions. The SR universal node is associated to axioms that indicate the equivalent and disjoints classes, same and different individuals, and equivalent and disjoint object properties. The PKBR universal node marks domain and range axioms for object, data and annotation properties.

Step 3. Once the correspondence between constructs of a concrete language and eCST universal nodes is identified, it is pretty straightforward to incorporate eCST universal nodes into tree rewrite rules of the grammar. For example, it has been identified in Step 2 that ontology declarations correspond to the PACKAGE_DECL universal node. Therefore, PACKAGE_DECL universal node will be incorporated in the tree rewrite rule of the production that describes ontology declarations as the following excerpt from the OWL2 grammar shows:

```
ontology : 'Ontology'  '('
               (ontologyIRI versionIRI?)?
               importo* annotation* axiom*
           ')'
           -> ^(PACKAGE_DECL
                    ^(KEYWORD 'Ontology')
                    ^(SEPARATOR '(')
                    (ontologyIRI versionIRI?)?
                    importo* annotation* axiom*
                    ^(SEPARATOR ')')
           );
```

We can also see that, besides the PACKAGE_DECL universal node, two other universal nodes are also incorporated in the rule: KEYWORD and SEPARATOR to mark keywords and separators present in ontology declarations, respectively.

Figure 4.4 shows how a simple ontology named "PL" looks in the eCST representation. This ontology contains exactly one axiom, `SubClassOf(:C :CPP)`, which states that each program written in the programming language C is at the same time valid C++ program.

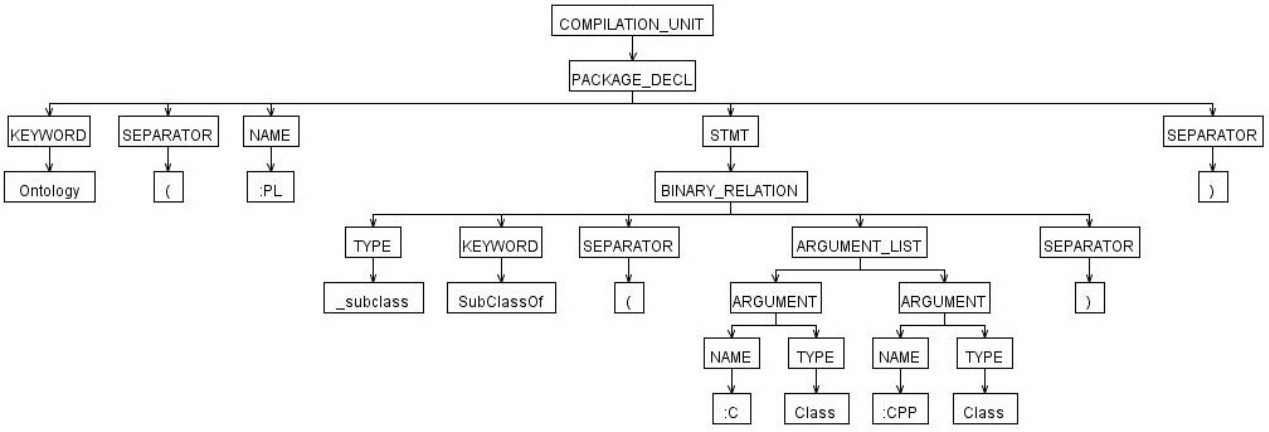


FIGURE 4.4: The eCST representation of a simple ontology.

Another advantage of ANTLR tree-rewrite rules is that they can be used to remove syntactic sugar. For example, there are three basic types of associations among classes: subsumption, equivalency and disjointness. However, OWL2 also contains the DisjointUnion construct of the form

$$\text{DisjointUnion}(A\ B_1\ B_2\ \ldots\ B_n),$$

where A is a named class, while B_i are class expressions that form disjoint classes. DisjointUnion states that A is union of disjoint classes determined by B_i which means that each of B_i is a subclass of

A. Therefore, DisjointUnion can be reduced to n SubClassOf axioms and one DisjointClasses axiom. The transformation can be described by the following ANTLR tree-rewrite rule:

```
disjointUnion : 'DisjointUnion' '(' annotation* classEntity classExpression+ ')'
-> ^(STMT
        ^(KEYWORD 'DisjointUnion')
        ^(SEPARATOR '(')
        annotation*
        (
            ^(BINARY_RELATION
                ^(TYPE EDITABLE_TOKEN["_subclass"])
                ^(ARGUMENT_LIST
                    ^(ARGUMENT classExpression ^(TYPE EDITABLE_TOKEN["Class"]))
                    ^(ARGUMENT classEntity ^(TYPE EDITABLE_TOKEN["Class"]))
                )
            )
        )+
        ^(SYMMETRIC_RELATION
            ^(TYPE EDITABLE_TOKEN["_disjoint_classes"])
            ^(ARGUMENT_LIST
                (
                    ^(ARGUMENT classExpression ^(TYPE EDITABLE_TOKEN["Class"]))
                )+
             )
         )
         ^(SEPARATOR ')')
    );
```

4.4.1.1 Benefits of the eCST representation of an ontology

Metrics that reflect complexity of a description written in a programming or formal language can be classified as follows:

1. Metrics of internal complexity that reflect lexical or syntactical complexity of the description or some of its parts. Lexical complexity measures are derived from the lexical elements of the language and reflect the complexity that is related to the volume of the description. Representative software metrics which belong to this category are the LOC family of metrics and Halstead's [Halstead, 1977] complexity measures. Syntactical complexity is related to the compositional (structural) complexity of concrete language constructs. Cyclomatic complexity [McCabe, 1976] is an example of widely used software metrics of syntactical complexity.

2. Metrics of design complexity reflect the complexity of dependency structures among identifiers introduced in the description. Those metrics quantify coupling, cohesion and inheritance among entities represented by the identifiers. Representative examples are metrics from the Chidamber-Kemerer object-oriented metrics suite [Chidamber and Kemerer, 1994].

3. Hybrid metrics which combine metrics of internal and design complexity. Examples are the WMC metric from the Chidamber-Kemerer suite and the Henry-Kafura complexity [Henry and Kafura, 1981].

As it can be seen from the review of related works on ontology metrics (see Section 4.2), the complexity of an ontological description is viewed as some measure of complexity of the underlying graph representation. In other words, already introduced ontology metrics belong to the category of design complexity metrics. The eCST representation provides us with the syntax-tree view of ontology. This view can be used to define (or adopt) and compute metrics of internal complexity, which is not possible in the graph-based view. For example, Halstead complexity metrics adopted for ontologies can be calculated in the same way as for software systems.

The ATTRIBUTE_DECL universal node can be used to recognize the declaration of ontological concepts, roles and named individuals. Relations among those entities can be identified by the analysis of eCST sub-trees rooted at BR, SR and PKBR universal nodes. This means that the graph representation of ontology can be extracted from the eCST representation of ontology. Therefore, an ontology metrics tool based on the eCST representation could be able to extract ontology graph and compute metrics of design complexity. Finally, metrics of internal and metrics of design complexity can be combined to obtain hybrid complexity metrics.

4.4.1.2 New metrics to evaluate ontologies

As already emphasized, the eCST view of ontology can be used to adopt existing software metrics of internal complexity. Halstead's metrics [Halstead, 1977] are measures of lexical complexity of source code. Halstead's idea is to view a computer program as a sequence of statements in the form "Operator Operands". To derive several measures of Halstead's complexity (volume, difficulty, effort), one has to count the total number of operator and operands, as well as the number of unique operator and operands. In order to adopt Halstead metrics for ontologies each token in an ontological description has to be classified as either operator or operand. We used the following idea: operator tokens are those tokens that are introduced by OWL2 language designers (keywords and separators), while operands are tokens introduced by knowledge engineers (names of ontological entities – IRIs, and constants). Operator tokens are marked with the KEYWORD and SEPARATOR eCST universal nodes, while operand tokens are marked with the NAME and CONSTANT eCST universal nodes. Then, Halstead's measures can be computed by counting and identifying unique operator and operand tokens.

We analyzed OWL2 FSS grammar and came to the conclusion that only two language constructs, class and data range expressions, are syntactically recursive. This means that class/data range expressions can be nested as arguments of other class/data range expressions. The effort to understand and modify an ontological description is directly affected by the syntactical complexity of its class and data range expressions, since other constructs are syntactically linear (do not contain complex arguments) and easy to comprehend. In this thesis we introduce a new ontology metric of internal complexity called *expression complexity*. Expression complexity is calculated by the following formula:

$$\mathrm{EXPC}(R(a_1\ a_2 \ldots\ a_n) = 1 + \sum_{i=1}^{n} \mathrm{EXPC}(a_i),$$

where R denotes a class/data range expression, while a_i are arguments of R. The expression complexity of identifiers (IRIs) and constants is equal to zero. For example, the following class expression

ObjectSomeValuesFrom(:hasChild ObjectUnionOf(:Boy :Girl))

has the expression complexity that is equal to 2. The expression complexity of an axiom is equal to the sum of expression complexities of all class/data range expressions contained in the axiom. In other words, the expression complexity of the axiom can be obtained by counting EXPR universal nodes contained in the eCST sub-tree representing the axiom. Similarly, the expression complexity of an ontology is the sum of expression complexities of all axioms contained in the ontology.

Henry and Kafura [Henry and Kafura, 1981] introduced a software complexity metric based on information flow. The complexity of a software entity is estimated using both a metric that reflect internal complexity (LOC) and two metrics of design complexity (Fan-In – the number of entities that reference the entity, and Fan-out – the number of entities referenced by the entity). The Henry-Kafura complexity of an entity E is calculated as

$$\text{HK}(E) = \text{LOC}(E)(\text{Fan-In}(E) \cdot \text{Fan-Out}(E))^2.$$

Therefore, Henry-Kafura complexity can be adopted for modularized ontologies as

$$\text{HK}(O) = \text{INT}(O)(\text{Fan-In}(O) \cdot \text{Fan-Out}(O))^2,$$

where $\text{INT}(O)$ is some measure of internal complexity of ontology module O (LOC, Halstead's metrics, etc.), while $\text{Fan-In}(O)$ and $\text{Fan-Out}(O)$ are the number of incoming links (in-degree) and the number of outgoing links (out-degree) for the node representing ontology O in appropriate ontology module network (see Definition 4.2). As it can be seen HK complexity increases with internal complexity as well as with afferent and efferent coupling of ontology modules.

4.4.2 ONGRAM tool

The extraction of the graph representation of an ontology is quite a different problem than the extraction of software networks due to the structural difference between ontological and software entities. All ontological entities except ontology modules are structurally atomic which means that they are not composed of other ontological entities. To the contrary, all software entities except variables are not structurally atomic: the definition of a software entity A associates the name of A to a body that contains the structure of A. Horizontal dependencies between software entity A and other entities are contained in the body of A. From the eCST point of view this means that horizontal dependencies between A and other entities are obtained by analysis of eCST sub-tree rooted at the universal node which marks the definition of A. On the other hand, horizontal dependencies between ontological entity A and other entities are contained in independent axioms of the ontology, i.e. in multiple sub-trees rooted at STMT universal nodes in the eCST representation. This means that the SNEIPL back-end of SSQSA (see Section 3.4.3) cannot be used to identify vertical dependencies among ontological entities. Therefore, we designed a new SSQSA back-end called ONGRAM which extracts ontology graph and computes ontology metrics.

ONGRAM uses the eCST representation of ontology to extract ontology networks and compute metrics reflecting the internal and design complexity of ontologies, as well as metric that express a mixture of those two complexity aspects (hybrid metrics). One eCST represents one ontology in the case of monolithic ontology design or one ontology module in the case of modularized ontology design. Figure 4.5 shows the architectural diagram of ONGRAM. It can be observed that ONGRAM consists of four components

1. OGE (Ontology Graph Extractor) extracts the ontology graph from an input set of eCSTs.

2. ICMetrics computes metrics of internal complexity: LOC, the number of axioms, Halstead's metrics and expression complexity. Mentioned metric are computed for each eCST in the input set and stored as attributes of nodes representing ontology modules. Expression complexity is computed for the whole ontology module as well as for individual axioms contained in the module in order to obtain the average expression complexity per axiom.

3. DMetrics calculates metrics of design complexity (both pure ontology metrics and software metrics adopted for ontologies) and domain-independent metrics (metrics of centrality and importance defined on any directed graph) using ontology graph that is previously formed by the OGE component of ONGRAM. Those metrics are also stored as attributes of nodes in the ontology graph.

4. Exporter filters the ontology graph in order to form specific subgraphs of extracted ontology graph (networks given by Definitions 4.2, 4.3, 4.4 and 4.5) and exports them in various file network formats (Pajek and GraphML network file formats).

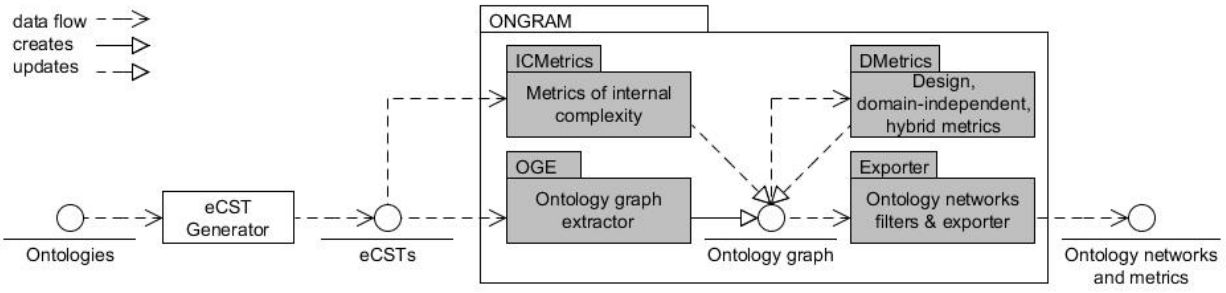


FIGURE 4.5: ONGRAM architecture.

The extraction of an ontology graph is done in two phases: in Phase 1 nodes corresponding to explicitly declared ontological entities are created, while in Phase 2 dependencies are identified and links in ontology graph are established. Phase 1 relies on the ATTRIBUTE_DECL and PACKAGE_DECL universal nodes that are used to recognize declared ontological entities. Both universal nodes have the NAME universal node in the sub-tree which determines the name of newly created node in ontology graph. Each ATTRIBUTE_DECL universal node has the TYPE universal node in the sub-tree which determines the type of created node (the type of corresponding ontological entity). The PACKAGE_DECL universal node always marks the definition of an ontology module.

Let A and B be two nodes in an ontology graph where A represents an ontology module and B represents ontological entity that is not ontology module. A and B are connected by the CONTAINS link $A \rightarrow B$ if the name (IRI) of B is of the form $\alpha S \beta$ where α is the name of A and S is the last occurrence of "#" in the name of B, i.e. the name of B is the name of A expanded by an IRI fragment which starts with the IRI fragment separator "#". Therefore, when a new node in ontology graph is created it is checked whether there is a node representing ontology module satisfying the previous condition. If such module exists then CONTAINS link is created.

Phase 2 identifies horizontal dependencies among ontological entities. However, new nodes can be also created in this phase because ontological entities have not to be explicitly declared. Also,

class expressions constituting anonymous classes are represented by nodes in ontology graph. Phase 2 traverses each eCST in the input set level by level looking for the following universal nodes:

- IMPORT_DECL (ID). ID marks an import statement. Therefore, for each ID in an eCST one IMPORTS link in ontology graph is created. The source node for an IMPORTS link is determined by the content of the NAME sub-tree located under the PACKAGE_DECL universal node in the eCST, while the destination node is determined by the content of the NAME sub-tree located under the IMPORT_DECL universal node.

- BINARY_RELATION (BR). This universal node marks statements that express explicitly stated asymmetric relations between two entities A and B. This means that for each BR universal node a link $A \rightarrow B$ will be created. The type of the relation is determined by the content of the TYPE sub-tree located under the BR universal node. BR also has a child which is the ARGUMENT_LIST universal node that marks arguments of the association. The ARGUMENT_LIST universal has two child ARGUMENT universal nodes. Naturally, the sub-tree rooted at the first ARGUMENT describes A, while the sub-tree rooted at the second argument describes B. Let T be an ARGUMENT sub-tree. Two types of T are possible with regard to universal nodes that are contained at the first level:

  1. If T has two child nodes then T represents a named ontological entity N. The name and type of N are determined by the NAME and TYPE sub-trees located under the ARGUMENT universal node, respectively. If the ontology graph does not contain the node corresponding to N then the node representing N is created which means that N was not explicitly declared.
  2. If T has one child EXPR universal node then T represents a class expression. In this case a new node N representing an anonymous class is created. For each NAME universal node located in the sub-tree it is checked whether it represents a named class C and REFERENCES link $N \rightarrow C$ is created.

- SYMMETRIC_RELATION (SR). This universal node marks statements that express explicitly stated symmetric relations of the same type between n entities A_1, A_2, .., and A_n. For each SR universal node $n(n-1)/2$ symmetric links $A_i \leftrightarrow A_j, 1 \leq i < j \leq n$ are created where the type of the relation is determined by the content of the TYPE sub-tree located under the BR universal node. The ARGUMENT sub-trees located under the SR universal node are processed in the same way as for the BR universal node.

- PARTIALLY_KNOWN_BINARY_RELATION (PKBR). This universal node marks statements that express domain and range asymmetric relations between two ontological entities A and B where one of them is a property (user-defined association) and the other is not. When a PKBR is observed then the pair (A, B) is added to a list of pairs L. After all eCSTs are processed then the list L is analyzed in order to assemble appropriate domain and range relations. A directed link $A \rightarrow C$ whose type B is created if and only if L contains pairs (A, B) and (B, C).

The ICMetrics component computes LOC metric in the same way as the SMIILE back-end [Rakić and Budimac, 2011a] does. Each token in the eCST representation is attributed with the number of line where it appears in the source code, thus the LOC of one ontology module is equal to the line attribute of the last token. In order to calculate Halstead metrics for one ontology module ONGRAM

traverse the corresponding eCST and counts all operators (KEYWORD and SEPARATOR universal nodes) and operands (NAME and CONST universal nodes). Each lexical universal node has exactly one token in its eCST sub-tree. Those tokens are put in the operator and operand sets in order to count the number of unique operator and operand tokens.

The number of axioms and expression complexity are calculated by counting the number of specific universal nodes. A similar algorithm is also used to calculate cyclomatic complexity in the SMIILE back-end [Rakić and Budimac, 2011a]. The number of axioms is the number of STMT universal nodes, while expression complexity is the number of EXPR universal nodes in the eCST corresponding to an ontology module.

The DMetrics component uses extracted ontology graph to compute graph-based and hybrid metrics. ONGRAM computes various metrics at the ontology, class and object level. Domain-independent metrics of centrality and importance (in-degree, out-degree, total-degree, betweenness centrality and page rank) are computed at all three levels. At the class level ONGRAM also computes

- NOC (number of children) and DIT (depth in inheritance tree), the inheritance metrics from the Chidamber-Kemerer object-oriented metric suite adopted for ontologies [Zhang et al., 2010].
- The number of instances which is the number of objects of a class. Let C be an arbitrary class. Then the number of instances of C is the number of links pointing to C whose source nodes correspond to objects.

The largest number of metrics is computed at the ontology level. Besides metrics of internal complexity computed by the ICMetrics component and already mentioned domain-independent graph-based metrics, ONGRAM at the ontology level also computes:

- The number of classes and objects contained in an ontology module,
- Coupling metrics introduced by Orme et al. [2006],
- Population and richness metrics introduced by Tartir et al. [2005],
- Henry-Kafura complexity adopted for ontologies (see Section 4.4.1.2),
- Graph clustering evaluation metrics adopted for ontologies (see Section 4.3).

4.5 Analysis of ontology networks

Analysis of ontology networks helps us to understand the complexity of ontology designs. In this Section we will investigate ontology networks associated to one modularized ontology which describes domain-specific terminology. The special attention will be given to its ontology module network since we want to understand how some real knowledge is actually modularized. We will follow the same methodology that was used to investigate the structure of software networks. In contrast to related studies that are presented in Section 4.5.1, our analysis is not pure topological, but enriched with a rich set of both ontology and domain-independent metrics that we were able to compute using ONGRAM.

4.5.1 Related work

The first notable work in the field of analysis of ontology networks was conducted by Gil and García [2006]. The authors combined 282 ontologies contained in the DAML ontology library into a single RDF graph and analyzed its structure. The results of the analysis showed that the network possesses scale-free and small-world properties.

The paper by Hoser et al. [2006] illustrated the benefits of applying domain-independent metrics of centrality to ontology networks. The authors used degree, betweenness and eigenvector centralities to identify the core concepts and roles in the SWRC (Semantic Web for Research Communities) and SUMO (Suggested Upper Merged Ontology) ontologies. The mentioned metrics were computed on ontology graphs that were constructed using an ontology graph extractor based on the KAON Ontology API.

Theoharis et al. [2008b] analyzed ontology subsumption networks and ontology class networks without subsumption links (this type of network in the paper is called a property graph) for 83 ontologies each of them containing more than 100 classes. The main findings of the study is that total-degree distributions of property graphs and in-degree distributions of subsumption networks can be very well approximated by power-laws for the majority of examined ontologies. The authors also showed that classes with high degrees in property graphs tend to be highly abstract (located at the higher levels in ontology subsumption networks). In their subsequent study [Theoharis et al., 2008a], the authors exploited the results of the analysis to generate synthetic ontologies that can be used for the benchmarking of ontology repositories and query languages.

The study by Cheng and Qu [2008] examined the ontology graph restricted to classes and properties (in the paper named as a *term dependence network*) representing 3090 unrelated semantic web ontologies collected by the Falcons semantic web search engine. The network was built by an extractor based on the Apache Jena library and contains more than one million of nodes and seven million of links. The authors analyzed degree distributions of the network, reachability and connectivity of nodes. The main findings of the study are:

- The in-degree distribution of the network follows a power-law.
- About half of nodes have small eccentricity (less than 6), but there is a relatively large portion of nodes (about 10%) which have relatively high eccentricity (higher than 25). Nodes with high eccentricity tend to be highly specialized (positioned at the bottom of the class hierarchy).
- The network contains relatively large strongly connected components implying the presence of large cyclic dependencies among classes.

The authors also examined the structure of the ontology module network (the authors use the term *vocabulary* instead of ontology, so this type of ontology network is in the paper denoted as a *vocabulary dependence network*). The main observation is that the network becomes extremely fragmented when only 4 language-level ontologies are removed from the network suggesting that the semantic web is still far away from a web of interlinked ontologies.

Zhang [2008] analyzed total degree distributions of seven ontology graphs representing monolithic semantic web ontologies. The analysis showed that all examined network possess the scale-free property. The same result was also obtained for two TCMLS (Traditional Chinese Medical Language System) ontologies [Ma and Chen, 2007]. Moreover, the TCMLS ontologies exhibit the small-world property. Zhang et al. [2010] analyzed the in-degree and out-degree distributions of three ontology graphs showing that they, similarly to previous findings, follow power-laws.

Ge et al. [2010] analyzed ontology object network encompassing instances present in all ontologies crawled by the Falcons search engine. The authors found that the network has a giant connected component and exhibits scale-free and small-world properties. Moreover, they investigated the structural evolution of the network by comparing two snapshots of the network, the first considering ontologies collected until 2008 (inclusive) and the latter considering ontologies collected until 2009 (again inclusive). The authors observed that the average degree of the network increased implying that the connections among objects tend to densify as the network evolves. Also they noticed that the diameter of the network shrinked indicating that the growth of the network caused a stronger small-world effect.

Ding et al. [2010] examined the structure of so called SameAs network extracted from the Web of Data. A SameAs network is the subnetwork of object ontology network restricted to equivalence relations among objects. In other words, each connected component in the SameAs network contains equivalent objects. The authors observed that the vast majority of connected components in the network have a small size concentrated around 2.4 objects. However, the network contains a small fraction of an unusually large components each of them encompassing more than one hundred objects and even more there are two components with more than thousand nodes. Secondly, the degree distribution of the network follows a power-law implying that it exhibits the scale-free property.

4.5.2 Case study

NASA's Semantic Web for Earth and Environmental Terminology (SWEET) is a collection of ontologies for describing earth science data and knowledge [Raskin and Pan, 2005]. SWEET is a modularized ontology consisting of more than 200 ontology modules. In this thesis we study the structure of networks representing the SWEET ontology in version 2.2. The networks and associated metrics were computed using the ONGRAM tool. The Protégé tool was used to collect ontologies[3] and convert them into the OWL2 functional-style syntax. After that, we transformed SWEET ontology modules into the eCST representation using the SSQSA front-end. The basic facts describing this conversion process are summarized in Table 4.2.

Table 4.2: Conversion of SWEET ontologies to the eCST representation.

Number of compilation units	204
Total LOC (excluding empty lines)	21449
Parse time (sec.)	8
Ontology size (MB)	3.03
eCST representation size (MB)	25.1

The SWEET ontology graph consists of 10873 nodes and 26725 links. Fourteen nodes in the graph are external nodes – ontological entities not belonging to SWEET ontology modules. Those external entities are connected to internal nodes by 154 links which is 0.57% of the total number links in the graph. The distribution of nodes and links of the graph is given in Table 4.3. As it can be observed from the data presented in the table the majority of entities in SWEET are classes, while the most dominant type of horizontal links are subsumption associations.

In this thesis we study the structure of three networks isolated from the SWEET ontology graph: ontology module network, ontology class network and ontology subsumption network. The sizes (the

[3]SWEET has an umbrella ontology called "sweetAll.owl" that imports all SWEET ontology modules and can be used to retrieve the whole ontology.

TABLE 4.3: The distribution of nodes and links in the SWEET ontology graph.

Distribution of nodes			Distribution of links		
Type	Total	[%]	Type	Total	[%]
Ontology modules	204	1.88	SUB	6532	24.44
Named classes	4416	40.61	ASSERTION	2325	8.70
Anonymous classes	1958	18.01	EQUIVALENT	1478	5.53
Objects	2245	20.65	DISJOINT	340	1.27
Object property	564	5.19	REFERENCES	1236	4.62
Data property	41	0.38	CONTAINS	10655	39.87
Anotation property	1	0.01	IMPORTS	1340	5.01
Literals	1444	13.28	User defined	2819	10.55

number of nodes and links) of examined networks are shown in Table 4.4. It is important to mention that the ontology called "sweetAll.owl" is deleted from the ontology module network since the only purpose of that module is to enable retrieval of the whole SWEET ontology. Namely, "sweetAll.owl" imports all ontology modules contained in the SWEET ontology and does not state any axiom.

TABLE 4.4: The number of nodes and links in the SWEET ontology networks.

Ontology network	The number of nodes	The number of links
Ontology module network	203	1138
Ontology class network	6374	8483
Ontology subsumption network	6003	6202

4.5.2.1 Connected component analysis

Ontology networks are directed graphs, thus we firstly identified weakly connected components (WCCs) in the networks from the experimental dataset and investigated their properties. The results are summarized in Table 4.5. It can be seen that the ontology module network and ontology class network consist of a single weakly connected component. On the other side, ontology subsumption network has 36 WCCs where one WCC is a giant WCC occupying more than 90% of nodes and 90% of links in the network. The existence of a single/giant weakly connected component in examined ontology networks is quite natural: SWEET is a set of related ontological modules which describe the domain-specific terminology for earth and environmental sciences. This property is also not surprising from the theoretical point of view: all examined network have the average degree greater than one (see Table 4.4 for the number of nodes and links; the average degree is equal to $2L/N$ where N and L and the number of nodes and links, respectively) which is the critical threshold for the emergence of a giant connected component in the Erdős-Renyi model of random graphs. Table 4.5 also shows the small-world coefficient, clustering coefficient and assortativity index of examined networks. For the ontology subsumption network mentioned metrics are computed considering only the giant WCC. It can be observed that all three networks are small-worlds in the Watts-Strogatz sense [Watts and Strogatz, 1998b]: their small-world coefficients are close to the predictions made by the Erdős-Renyi model of random graphs (SW $\approx$ SW-rnd) and at the same time the clustering coefficients are drastically higher than the clustering coefficients of comparable random graphs (CC $\gg$ CC-rnd). It is interesting to observe that at different levels of abstraction there are different assortativity mixing patterns. The ontology module network shows weak assortative mixing. On the other hand, the networks at the

class level exhibit significant disassortative mixing which means that they do not have a hub-like core reflected by mutually connected highly-coupled ontological concepts.

TABLE 4.5: Weakly connected components of the SWEET ontology networks. OMN denotes the ontology module network, OCN the ontology class network and OSN ontology subsumption network. #WCC – the number of weakly connected components (WCCs), LWCCN – the fraction of nodes in the largest WCC, LWCCL – the fraction of links in the largest WCC, SW – the small-world coefficient, SW-rnd – the small-world coefficient of a comparable random graph, CC – the clustering coefficient, CC-rnd – the clustering coefficient of a comparable random graph, A – assortativity index.

Network	#WCC	LWCCN [%]	LWCCL [%]	SW	SW-rnd	CC	CC-rnd	A
OMN	1	100	100	2.55	2.22	0.15	0.028	0.023
OCN	1	100	100	9.51	9.74	0.007	0.00021	-0.158
OSN	36	93.35	94.11	11.8	11.74	0.001	0.00017	-0.171

Table 4.6 provides the basic quantities describing identified strongly connected components (SCCs). It can be observed that each network possess a unique pattern regarding the strong connectivity of nodes:

- Ontology module network has a small number of SCCs, but the largest SCC can be considered as giant since it encompasses more than the half of nodes and the half of links. The second largest SCC contains 4 nodes connected by 6 links, while the smallest has two nodes connected by 2 links. The reciprocity of links (denoted by R in Table 4.6) is relatively small implying that there is a small portion of ontology modules (4% of the total number) that are mutually directly dependent. However, the reciprocity of IMPORTS links is still higher than it could be expected by random chance ($R_n > 0$). In contrast to the link reciprocity, the path reciprocity is extremely high: more than 60% of all dependencies among ontological modules (both direct and indirect) are cyclic dependencies. Large cyclic dependencies between ontology modules can be considered as the indicator of poor modularization with respect to the effort needed to comprehend ontology design. When there are no cyclic dependencies between modules then it is possible to make a topological sort of modules in order to obtain the hierarchical view of dependencies between modules. However, the design of ontology modules in SWEET is highly cyclic and strongly deviates from a layered design.

- Ontology class network has a large number of SCCs, but all SCCs are relatively small. The largest SCC encompasses 11 nodes connected by 17 links. Moreover, the reciprocity of links is significantly higher than the reciprocity of paths implying that the majority of existing cyclic dependencies among concepts are direct dependencies stated by EquivalentClasses and DisjointClasses axioms. Since large indirect cyclic dependencies are absent SCCs in the network can be easily comprehend by ontology engineers. Moreover, the majority of strongly connected components (more than 80% of the total number) are of trivial complexity which means that the number of nodes in a component is equal to the number of links in the component, i.e. the component is a pure circle of nodes.

- Ontology subsumption network has only one SCC of the minimal size. Only two classes ("RadiativeForcing" and "RadiantFlux" from "quanEnergyFlux.owl") are mutually related by the

subsumption association[4]. Therefore, we can conclude that the hierarchy of concepts in SWEET has a layered design.

TABLE 4.6: Strongly connected components of the SWEET ontology networks. #SCC – the number of strongly connected components (SCCs), LSCCN – the fraction of nodes in the largest SCC, LSCCL – the fraction of links in the largest SCC, S – the number of nodes contained in all SCCs, R – link reciprocity, R_n – normalized link reciprocity, R_p – path reciprocity, C – the number of SCCs of trivial complexity.

Network	#SCC	LSCCN [%]	LSCCL [%]	S [%]	R	R_n	R_p	C [%]
OMN	3	61.57	60.63	64.53	0.0545	0.0275	0.608	33.33
OCN	410	0.17	0.20	15.05	0.1214	0.1212	0.0136	80.24
OSN	1	0.03	0.03	0.03	0.0004	0.0003	0.0001	100

We applied the metric based comparison test in order to determine characteristics of the giant strongly connected component (GSCC) in the ontology module network. The results are shown in Table 4.7. For a list of ontology metrics, the table shows the average value of the metric M for modules in the GSCC (denoted by Avg(GSCC) in the table), the average value of the metric M for modules that are not in the GSCC (Avg(Rest)), the value of the Mann-Whitney U statistic, the obtained significance probability of the MWU test, whether the null hypothesis of the test was accepted or not, and two probabilities of superiority. PS_1 denotes the probability that a randomly selected module from the GSCC has strictly larger value of the metric M than a randomly selected module not belonging to the GSCC, while PS_2 denotes the opposite probability of superiority. The null hypothesis of the MWU test (no statistically significant difference between two sets of metric values) is accepted for the following group of metrics:

- Two metrics of internal complexity, total expression complexity (TEXPR) and average expression complexity (AEXPR, expression complexity per axiom). This means that the complexity of axioms contained in modules that are involved in cyclic dependencies is similar to the complexity of axioms contained in modules that are not involved in cyclic dependencies.

- The OUT metric which is the number of out-going links for a module in the network. An ontology module contained in the GSCC does not tend to aggregate significantly more (or less) modules compared to a module not belonging to the GSCC.

- Population and richness metrics from the Tartir et al. [2005] metric suite (AP – average population, CR – class richness, RR – relationship richness), as well as the NINST metric which is the number of instances belonging to a module. In other words, there are no statistically significant differences between the GSCC and the rest of the modules with respect to the actual instantiation of classes and diversity of associations.

- The strength of coupling metrics from the Orme et al. [2006] metric suite (NEC – the number of external classes, REC – references to external classes).

- Graph clustering evaluation metrics reflecting cohesion of ontology modules (CON – conductance, EXP – expansion).

[4]This means that "RadiativeForcing" and "RadiantFlux" are equivalent concepts but that was not explicitly stated in the ontological description.

TABLE 4.7: Characteristics of the largest strongly connected component in the SWEET ontology module network.

Metric	Avg(GSCC)	Avg(Rest)	U	p	NullHyp	PS_1	PS_2
LOC	106.8	101.1	6029	0.0045	rejected	0.61	0.38
TEXPR	5.26	4.11	5412	0.1872	**accepted**	0.5	0.39
AEXPR	0.068	0.071	5058	0.6522	**accepted**	0.49	0.45
AXM	92.8	87.3	6033	0.0044	rejected	0.61	0.38
HVOL	2905	2855.5	6048	0.0039	rejected	0.62	0.38
HDIF	20.5	17.7	6376	0.0002	rejected	0.65	0.35
NCLASS	34.06	27.04	5773	0.0274	rejected	0.58	0.4
NINST	9.24	13.97	5007	0.7448	**accepted**	0.24	0.27
IN	8.34	1.22	9110	$< 10^{-4}$	rejected	**0.91**	0.04
OUT	5.77	5.35	5002	0.7541	**accepted**	0.47	0.46
TOT	14.1	6.55	8057	$< 10^{-4}$	rejected	**0.81**	0.15
PR	0.0066	0.0022	8971	$< 10^{-4}$	rejected	**0.92**	0.08
BET	870.8	20.6	8781	$< 10^{-4}$	rejected	**0.89**	0.09
HK	717545.28	7888.64	8959	$< 10^{-4}$	rejected	**0.92**	0.08
AP	1.74	1.28	4892	0.9666	**accepted**	0.24	0.23
CR	0.11	0.09	5104	0.5729	**accepted**	0.3	0.25
RR	0.23	0.23	5025	0.7125	**accepted**	0.5	0.47
NEC	5.12	4.68	4962	0.8298	**accepted**	0.46	0.44
REC	9.49	8.76	4981	0.7946	**accepted**	0.47	0.49
CON	0.21	0.22	5470	0.1438	**accepted**	0.44	0.56
EXP	0.29	0.31	5498	0.1259	**accepted**	0.43	0.56

A high value of one of the probabilities of superiority indicate a drastic difference between the GSCC and the rest of ontology modules in SWEET. As it can be observed from the data presented in Table 4.7 there is a large difference between in-degrees of modules in the GSCC and in-degrees of the rest of modules: probability that the in-degree of a randomly selected module from the GSCC is strictly higher than in-degree of a module which is not in the GSCC is extremely high (0.91). The average in-degree for modules in the GSCC is 9.34, while the average in-degree for the rest of the network is four times lower. This means that modules from the GSCC possess a higher degree of internal reuse. Large differences in total-degrees and Henry-Kafura complexities are the consequence of large differences in in-degrees since those metrics incorporate in-degree as the constituent factor. The metric-based analysis also revealed that there are statistically significant differences in the metrics of global centrality and importance (page rank and betweenness centrality) for compared groups of modules. This means that the GSCC encompasses the most important modules from the design point of view and modules which are located in the core the network. Therefore, it can be concluded that SWEET ontology module network possesses a *strongly connected core* containing most important and most reused modules.

4.5.2.2 Degree distribution analysis

The existence of scale-free scaling behavior in ontology networks was reported in several studies [Cheng and Qu, 2008; Ding et al., 2010; Ge et al., 2010; Ma and Chen, 2007; Theoharis et al., 2008b; Zhang, 2008; Zhang et al., 2010] reviewed in Section 4.5.1. However, the main characteristic (which can be considered as a weakness) of mentioned studies is that the authors considered only power-laws

to model empirically observed degree distributions, while other theoretical distributions were not taken into account. In this thesis the power-law test introduced in [Clauset et al., 2009] is used to investigate weather ontology networks from the experimental dataset exhibit the scale-free property. Three theoretical distributions are taken into account: power-law, exponential and log-normal. The results of the test are summarized in Table 4.8, where x_m denotes the lower bound of obtained power-law scaling region, M is the maximal value of degree, α is the power-law scaling exponent, and p-value is the statistical significance of the power-law fit. p-value smaller than 0.1 implies that a power-law model is not plausible for an empirically observed distributions [Clauset et al., 2009]. Table 4.8 also shows the comparison between power-law and alternative theoretical distributions when a power-law model is plausible fit to data. The value of the log likelihood ratio is denoted by R_d, where d is the alternative distribution ("ln" – log-normal, "e" – exponential). Positive and statistically significant R_d ($R_d > 0, p(R_d) < 0.1$) indicates that the power-law fit is favored over the best fit of the alternative distribution d, while the negative and statistically significant R_d ($R_d < 0, p(R_d) < 0.1$) indicates that the alternative distribution is better. As it can be seen from the data presented in Table 4.8 the power-law hypothesis is rejected for the majority of empirically observed distributions. For three distributions, the total-degree and in-degree distributions of the SWEET ontology module network (OMN) and the out-degree distribution of the SWEET ontology class network (OCN), obtained power-law is the plausible model in the tails of the distributions ($x_m > 1$). However, for the tails of the total-degree and in-degree distributions of the SWEET OMN the log-normal distribution provides better fits to data compared to power-laws ($R_{ln} < 0$, $p(R_{ln}) < 0.1$). For the out-degree distribution of the SWEET OCN all three theoretical distributions are plausible models in the tail. However, the size of the tail exhibiting the power-law scaling is extremely small suggesting that the power-law is poor model for the data since basically every theoretical distribution can be very well fitted to a such small range of values. Therefore, we can conclude that examined ontology networks do not possess the scale-free property.

TABLE 4.8: The results of the power-law test for degree distributions of the SWEET ontology networks. OMN, OCN and OSN denote ontology module network, ontology class network and ontology subsumption network, respectively.

Network	Distribution	x_m	M	α	p-value	R_{ln}	$p(R_{ln})$	R_e	$p(R_e)$
OMN	Total-degree	11	56	3.24	**0.31**	-4.09	$< 10^{-4}$	9.18	$< 10^{-4}$
	In-degree	8	51	2.78	**0.44**	-8.19	$< 10^{-4}$	15.95	$< 10^{-4}$
	Out-degree	8	17	4.69	0.01				
OCN	Total-degree	2	54	2.54	$< 10^{-4}$				
	In-degree	1	53	2.3	$< 10^{-4}$				
	Out-degree	5	17	6.26	**0.99**	-0.22	0.82	1.01	0.31
OSN	Total-degree	3	54	2.72	$< 10^{-4}$				
	In-degree	1	53	2.39	$< 10^{-4}$				
	Out-degree	3	17	5.67	$< 10^{-4}$				

Do ontology networks from the experimental data set contains hubs – highly connected nodes whose total-degree is significantly higher than the average? In order to provide an answer to this question we compared the best fits of log-normal, exponential and Poisson distribution to the empirically observed total degree distributions through the whole range of degree values. If the Poisson distribution provides the best fit to a total-degree distribution then the corresponding network possesses the connectivity pattern characteristic to Erdős-Renyi random graphs and consequently does not contain hubs. The results of the comparisons are showed in Table 4.9. The values of log likelihood ratios are denoted

by $R\left(\frac{d_1}{d_2}\right)$, where d_1 and d_2 are two theoretical distributions ("ps" – Poisson, "ln" – log-normal, "e" – exponential). Positive and statistically significant R ($R\left(\frac{d_1}{d_2}\right) > 0$, $p < 0.1$) indicates that d_1 is preferred over d_2, while a negative and significant value of R indicates exactly the opposite. In the case that the value of R is not statistically significant ($p \geq 0.1$) then d_1 and d_2 are equally plausible. As it can be seen from the data presented in Table 4.9 the Poisson distribution is never preferred over the alternatives and the log-normal distribution always provide the best fit to the data. Moreover, the coefficient of variation and skewness of the empirically observed total-degree distributions are at least two times higher than the same quantities of the best fits of the Poisson distributions (see Table 4.10). Therefore, it can be concluded that examined networks contain hubs.

TABLE 4.9: The comparison of the best Poisson, log-normal and exponential fits to the empirically observed total-degree distributions.

Network	$R\left(\frac{\text{ps}}{\text{ln}}\right)$	p	$R\left(\frac{\text{ps}}{\text{e}}\right)$	p	$R\left(\frac{\text{ln}}{\text{e}}\right)$	p	Best fit
OMN	-3.74	0.0001	-3.39	0.0007	2.59	0.009	log-normal
OCN	-6.65	$< 10^{-4}$	-6.11	$< 10^{-4}$	10.07	$< 10^{-4}$	log-normal
OSN	-4.84	$< 10^{-4}$	-1.57	0.11	15.11	$< 10^{-4}$	log-normal

TABLE 4.10: The coefficient of variation (c_v), skewness (G_1) and the average value (μ) of the total-degree sequence for examined ontology networks. The coefficient of variation and skewness of the Poisson fit are equal to $\mu^{-0.5}$.

Network	c_v	G_1	μ	$\mu^{-0.5}$
OMN	0.81	2.68	11.2	0.3
OCN	1.21	5.32	2.66	0.61
OSN	1.38	5.92	2.08	0.69

4.5.2.3 Characteristics of hubs

As shown in the Section 4.5.2.2 ontology networks from the experimental dataset do not exhibit the scale-free property but contain hubs – ontology modules and ontology classes whose total degrees are significantly higher than the average values. Similarly as in the analysis of software networks, a node in an ontology network is considered as hub if it belongs to a minimal subset of nodes H such that the sum of total degrees of nodes in H is higher than the sum of total degrees of the rest of nodes ($N \setminus H$, where N is the set of nodes in the network). The size of the H set and the minimal total degree of nodes in H for SWEET networks are shown in Table 4.11. It can be observed that the minimal total degree of hub classes is relatively low – more than 70% of classes contained in the SWEET OCN and OSN are classes that have total degree less than 3 which means that the vast majority of SWEET classes are loosely coupled.

TABLE 4.11: The fraction of highly coupled nodes (the size of the H set) and the minimal total degree of nodes contained in H for SWEET networks.

Network	$\lvert H\rvert$ [%]	H_d
Ontology module network	25.62	14
Ontology class network	29.26	3
Ontology subsumption network	22.36	3

Table 4.12 presents the top five highest coupled hubs in each network from the experimental dataset showing their total-degree, in-degree and out-degree. As it can be seen in-degree values significantly dominate over out-degree values indicating that highly coupled ontology modules and concepts are, similarly to highly coupled classes in object-oriented software systems examined in the previous chapter of the dissertation, caused by internal reuse. Therefore, in the same way as for software networks, we investigated the degree of disbalance between in-degree and out-degree for hubs by measuring two quantities:

1. C_k which is the average ratio of in-degree to total-degree for nodes whose total-degree is higher or equal to k, and

2. P_k which is the the probability that a randomly selected node whose total degree is higher or equal to k has two times higher in-degree than out-degree.

TABLE 4.12: The top five highest coupled nodes in the SWEET ontology networks. Total, In and Out denote total-degree, in-degree and out-degree, respectively.

Network	Node name	Total	In	Out
OMN	realm.owl	56	48	8
	quan.owl	54	46	8
	matr.owl	53	51	2
	phen.owl	51	42	9
	reprMath.owl	43	40	3
OCN	phenAtmoWindMesoscale.owl#MesoscaleWind	54	53	1
	human.owl#HumanActivity	46	46	0
	quanTemperature.owl#Temperature	41	36	6
	realm.owl#Ocean	36	31	5
	quanPressure.owl#Pressure	36	28	8
OSN	phenAtmoWindMesoscale.owl#MesoscaleWind	54	53	1
	human.owl#HumanActivity	44	44	0
	quanFraction.owl#FractionalProperty	33	30	3
	phenAtmoCloud.owl#Cloud	32	27	5
	phenAtmo.owl#MeteorologicalPhenomena	31	30	1

Figure 4.6 shows the values of C_k and P_k for the SWEET module, class and subsumption ontology networks. It can be observed that both C_k and P_k increase with k. This means that the disbalance between in-degree and out-degree becomes more drastic with higher values of total degree. In other words, highly coupled modules and classes are dominantly caused by internal reuse not by internal aggregation. Therefore, metrics measuring module or class coupling (such as adopted CBO from the Chidamber-Kemerer metric suite) can indicate only negative aspects of excessive internal reuse, not negative aspects of excessive internal aggregation.

In order to determine characteristics of hub modules we applied the metric-based comparison test. The results are shown in Table 4.13. It can be seen that hub modules tend to have significantly higher values of the LOC (lines of code), AXM (the number of axiom), HVOL (Halstead volume), NCLASS (the number of classes), IN (in-degree), PR (page rank), BET (betweenness centrality), and HK (Henry-Kafura complexity) metrics compared to non-hub modules. This means that highly coupled ontological modules tend to be more voluminous, more central and more important (considering the whole ontology design) compared to non-hub modules. On the other side, the null-hypothesis of the MWU test was accepted for the following metrics: AEXPR (average expression complexity per axiom), NINST (the number of instances), AP (average population), CR (class richness), CON (conductance)

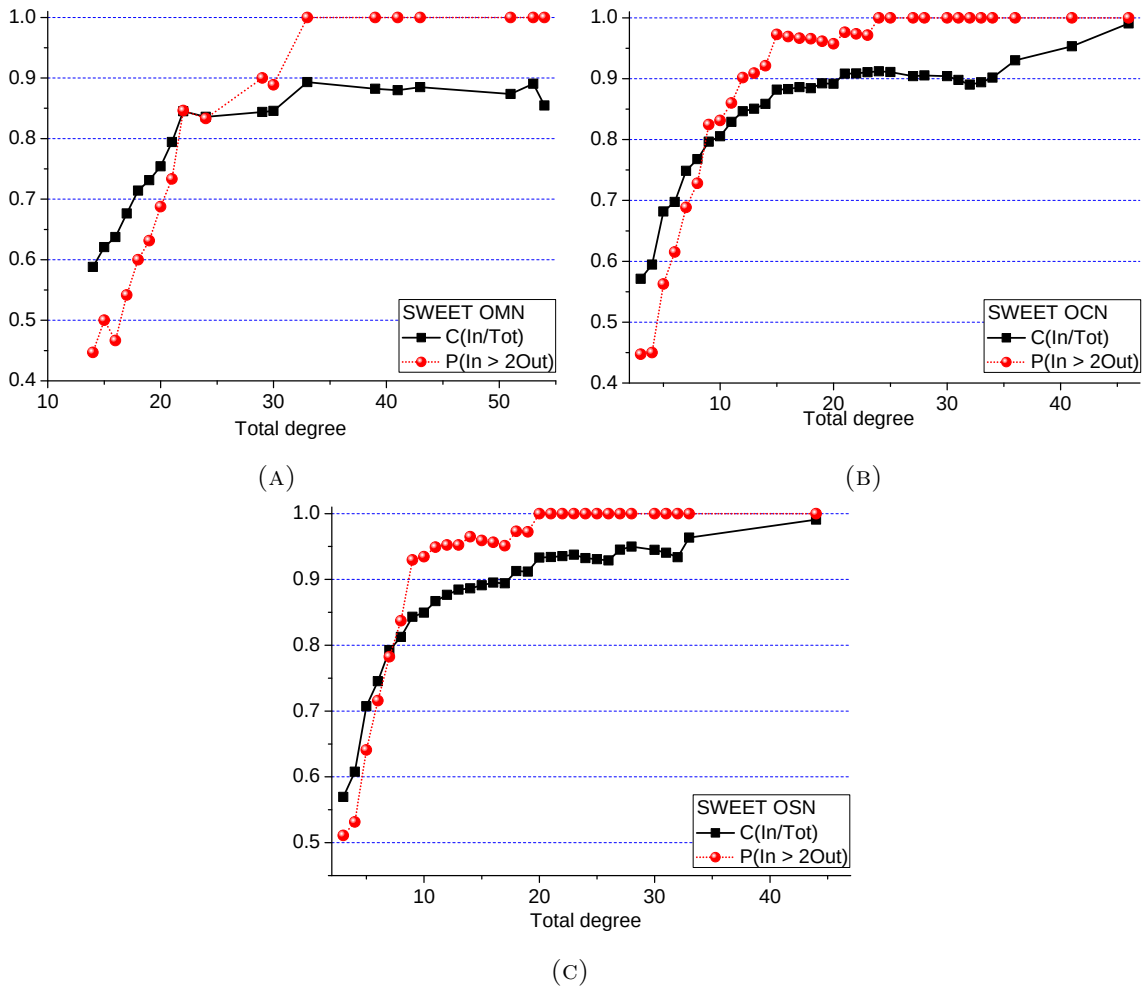


FIGURE 4.6: In-out degree disbalance for the SWEET ontology networks: (A) ontology module network, (B) ontology class network and (D) ontology subsumption network.

and EXP (expansion). This means that there are no statistically significant differences between hub modules and non-hub modules regarding the cohesion of their constituent elements, the number of objects and actual instantiation of classes, and complexity of individual axioms that are contained in them.

4.5.2.4 Cohesion of ontology modules

For each SWEET module we computed graph clustering evaluation (GCE) metrics in order to empirically evaluate our theoretical proposal presented in Section 4.3.

Firstly, we investigated whether there are strong correlations among GCE metrics by computing the Spearman correlation coefficient. The results are shown in Table 4.14 where CON denotes conductance, EXP expansion, CUTR cut-ratio, AODF average-ODF, MODF maximum-ODF, and FODF flake-ODF. It can be seen that there are strong correlations among all GCE metrics with one exception: the MODF metric weakly correlates with the rest of GCE metrics. The examination of obtained MODF metric values showed that this metric possess a small degree of discrimination. 184 SWEET modules (90.64% of the total number) have MODF equal to 1 which is the maximal possible value of the metric. This means that the vast majority of SWEET modules contain at least one ontological entity which does not reference any other entity from the same module but exclusively entities from

TABLE 4.13: Characteristics of hub modules in the SWEET ontology module network.

Metric	Avg(Hubs)	Avg(Rest)	U	p	NullHyp	PS_1	PS_2
LOC	138.4	93	6185	$< 10^{-4}$	rejected	**0.79**	0.21
TEXPR	7.6	3.9	5449	$< 10^{-4}$	rejected	0.66	0.27
AEXPR	0.076	0.068	4579	0.07	**accepted**	0.57	0.4
AXM	122.5	79.7	6097	$< 10^{-4}$	rejected	**0.77**	0.22
HVOL	3931.3	2526.1	6237	$< 10^{-4}$	rejected	**0.79**	0.21
HDIF	23.1	18.2	5797	$< 10^{-4}$	rejected	0.74	0.26
NCLASS	46.6	26.1	5947	$< 10^{-4}$	rejected	**0.75**	0.24
NINST	8.8	11.8	4316	0.28	**accepted**	0.19	0.29
IN	14.7	2.5	7214	$< 10^{-4}$	rejected	**0.9**	0.06
OUT	7.4	5	5175	0.0006	rejected	0.62	0.31
PR	0.0128	0.0022	6737	$< 10^{-4}$	rejected	**0.86**	0.14
BET	1438.7	236.01	6815	$< 10^{-4}$	rejected	**0.87**	0.13
HK	1681429.9	19033.9	7688	$< 10^{-4}$	rejected	**0.98**	0.02
AP	0.83	1.82	4198	0.45	**accepted**	0.19	0.26
CR	0.07	0.11	3926	0.99	**accepted**	0.29	0.28
RR	0.28	0.22	4845	0.01	rejected	0.61	0.38
NEC	7	4.2	5257	0.0003	rejected	0.63	0.29
REC	13.2	7.8	5019	0.003	rejected	0.62	0.34
CON	0.19	0.22	4450	0.15	**accepted**	0.44	0.57
EXP	0.26	0.31	4409	0.18	**accepted**	0.44	0.56

other modules. It is important to recall that the FODF metric measures cohesion while other metrics reflect the lack of cohesion. Thus, there are strong negative correlations between FODF and cut-based GCE metrics, as well as FODF and AODF.

TABLE 4.14: Spearman correlations between GCE metrics.

	EXP	CON	CUTR	AODF	MODF
CON	0.978				
CUTR	1	0.978			
AODF	0.924	0.944	0.924		
MODF	0.174	0.187	0.174	0.31	
FODF	-0.903	-0.91	-0.903	-0.942	-0.276

GCE metrics take into account external dependencies (links connecting ontological entities from different modules) when evaluating cohesiveness of modules. Cohesiveness of ontological entities can be also evaluated relying only of internal dependencies (links connecting ontological entities from the same module). The later is the main characteristic of widely used software engineering cohesion metrics that can be also adopted for ontologies. Let m be an ontology module, let e denotes the set of ontological entities contained in m and let G_m be the graph whose set of nodes is e and the set of links corresponds to existing relations among the elements of e. Then we can compute two basic cohesion measures that consider only internal dependencies:

- Density of G_m which is the actual number of links in G_m divided by the total number of all possible links in G_m, i.e

$$\text{DEN} = \frac{|L|}{|N||N-1|}.$$

- Connectedness of G_m which is the number of weakly connected components in G_m.

Table 4.15 shows the values of the Spearman correlation between GCE metrics and metrics of internal density and connectedness. As it can be observed there are no strong correlations between those two types of cohesion measures indicating usefulness of GCE metrics. Namely, cohesion metrics that consider only internal dependencies cannot point to ontology modules whose constituent entities form either good or bad clusters in the ontology graph.

TABLE 4.15: Spearman correlations between GCE metrics and metrics of internal density (DEN) and connectedness (COMP).

	EXP	CON	CUTR	AODF	FODF
DEN	-0.035	-0.056	-0.038	-0.109	0.076
COMP	0.174	0.265	0.175	0.253	-0.235

From the perspective of GCE metrics, an ontology module can be classified either as Radicchi strong, Radicchi weak or poorly cohesive cluster. SWEET consists of 203 ontologies where 18 of them (12.25%) are Radicchi strong, while 196 of them (96.08%) are Raddichi weak. Other 8 SWEET ontology modules that are not Raddichi weak are listed in Table 4.16. The small percentage of Radicchi non-weak modules in SWEET suggests that the knowledge contained in this ontology is well modularized with respect to the well-known principle of high cohesion.

TABLE 4.16: Poorly cohesive modules in SWEET. *A* denotes the average value of metrics considering all SWEET ontology modules. Ontology modules are sorted by conductance.

Module	LOC	TEXPR	IN	OUT	PR	BET	CON
stateSpaceConfiguration.owl	106	0	1	2	0.0016	12	0.75
stateTimeFrequency.owl	72	0	2	7	0.0021	260	0.75
quanTimeAverage.owl	89	1	3	8	0.0012	451	0.74
stateSpace.owl	70	0	0	5	0.0010	0	0.65
realmAtmoWeather.owl	61	4	0	7	0.0010	0	0.65
reprSpaceDirection.owl	97	0	9	2	0.0045	16	0.61
phenOcean.owl	15	1	2	2	0.0014	13	0.6
stateTime.owl	83	5	3	5	0.0015	7	0.5
A	104.6	4.82	5.6	5.6	0.0049	544	0.22

SWEET ontology modules listed in Table 4.16 are candidates for ontology refactoring, i.e. ontology engineers should examine them in order to see if there is a better way to integrate parts of those modules into other modules in order to reduce the number of Radicchi non-weak modules. It can be observed that poorly cohesive modules have at least two times higher conductance compared to the average conductance. Those modules tend to be smaller in size and less internally complex than an average module: the majority of poorly cohesive modules have LOC and expression complexity smaller than the average value. Additionally, the majority of poorly cohesive modules are located at the periphery of the ontology module network (betweenness centrality equal to zero or significantly less than the average) and consequently have smaller importance according to the page rank measure.

4.5.2.5 Correlations between ontology metrics

A recent study by Sicilia et al. [2012] investigated correlations between ontology metrics when they are computed from randomly sampled ontologies. In this thesis we use a quite different instrumentation

of correlations. Namely, we investigate correlations between ontology metrics that are computed from related ontologies, i.e. ontologies engineered in the same project which form one large morularized ontology. As we will see correlations computed in this way can reveal interesting patterns that can help us to understand how knowledge from a certain domain is modularized.

Table 4.17 shows the values of the Spearman correlation coefficient for metrics of internal complexity. The average expression complexity per axiom is excluded from the analysis since it is not a cumulative, but normalized measure of internal complexity. It can be observed that LOC strongly correlates ($\rho > 0.7$) with both Halstead's measures. A strong correlation between LOC and expression complexity is absent implying that highly complex class expressions can be found in both small-size and large-size modules.

Table 4.17: Spearman correlations between metrics of internal complexity.

	LOC	HVOL	HDIFF
HVOL	0.994		
HDIFF	0.727	0.722	
TEXPR	0.266	0.338	0.143

Strong Sperman correlations ($|\rho| > 0.7$) between metrics of design complexity are listed in Table 4.18. It can be observed that in-degree strongly correlates with page rank and betweenness centrality implying that the most reused ontological modules tend to be located in the center of the SWEET ontology module network and be the most important considering the whole ontology design. Consequently, total degree strongly correlates with metrics of centrality and importance since high coupling in SWEET is caused by internal reuse (see Section 4.5.2.3). There are also strong correlations between metrics from the Orme et al. [2006] metrics suite that reflect the strength of coupling of an ontology module to other ontology modules (NEC and REC). Out-degree strongly correlates with NEC (correlation with REC is equal to 0.68) which means that the strength of coupling of a module increases with the number of aggregated modules.

Table 4.18: Spearman correlations between metrics of design complexity.

Metric 1	Metric 2	Spearman correlation coefficient
IN	PR	0.95
NEC	REC	0.94
IN	TOT	0.75
IN	BET	0.75
OUT	NEC	0.72
PR	BET	0.71
TOT	BET	0.70

Finally, we investigated correlations between metrics of internal and design complexity. In this analysis we also included Henry-Kafura complexity (HKC) which is a hybrid complexity metric that combines metrics of internal and design complexity. Table 4.19 shows metric pairs for which a strong correlation ($|\rho| > 0.7$) is observed. It can be seen that HKC strongly correlates with in-degree (and consequently with page-rank and betweeness centrality due to strong correlation between those metrics and in-degree). However, the strong correlation between HK and in-degree is not surprising since in-degree is one of the constituent component of HK. On the other hand, HK complexity does not strongly correlate with other two constituent metrics, LOC and out-degree, indicating that in-degree

dominates over them. Expression complexity strongly correlates with both metrics that quantify the strength of module coupling. This means that modules which contains a large amount of complex class expressions tend to establish strong connections to other modules.

TABLE 4.19: Spearman correlations between metrics of internal and design complexity.

Metric 1	Metric 2	Spearman correlation coefficient
HK	IN	0.87
HK	TOT	0.85
HK	BET	0.85
HK	PR	0.78
TEXPR	REC	0.77
TEXPR	NEC	0.74

4.5.2.6 Final remark on SWEET modularization quality

As emphasized in Section 4.1 a good modularization of an ontology description should conform to the principle of low coupling and high cohesion. Another important aspect of ontology modularization is to enable reuse of ontological modules and better understandability of the whole ontology description. Does the SWEET ontology possess a good degree modularization? The analysis presented in previous sections can provide an answer to the previously posed question. Firstly, nearly all modules in SWEET tend to be Radicchi weak clusters which means that they possess a satisfactory degree of cohesion. Regarding coupling, we showed that SWEET contains highly coupled modules. However, those highly coupled modules are dominantly caused by internal reuse. Internal reuse can be considered as a good ontology engineering practice and cannot cause problems if an ontology module that is being internally reused was previously verified to be consistent and coherent[5]. Having in mind everything previously said it can be concluded that SWEET possesses a good degree of modularization. However, the SWEET modularization is not easily comprehensible and does not enable efficient external reuse of SWEET modules. As we showed in Section 4.5.2.1, the SWEET ontology module network contains a strongly connected core that encompasses more than half of SWEET modules. Circular structures are hard to comprehend since they cannot be decomposed into layers which can be "consumed" separately in a hierarchical order – if we want to understand one module from the SWEET strongly connected core we have to be fully aware of all modules in the core. Secondly, if we want to externally reuse one module from the core in some other ontology then the import of that module will indirectly import all modules from the core. Another highly interesting point is that the SWEET strongly connected core contains the most internally reused modules indicating that internal reuse actually caused the existence of the core. Therefore, we can conclude that extensive internal reuse actually made external reuse inefficient.

4.6 Summary and future work

In this chapter of the dissertation we presented ONGRAM (Ontology Graphs and Metrics), a tool that extracts networks representing (modularized) semantic web ontologies and computes ontology metrics. The tool is realized as one of the back-ends of the SSQSA framework after SSQSA was

[5] An ontology is incoherent if it contains as least one concept that cannot have any instances. An ontology is inconsistent if it contains contradictory axioms.

extended to support the OWL2 language. The eCST representation of ontological descriptions, produced by the SSQSA front-end, enabled us to define new and adopt existing software engineering metrics reflecting internal (lexical and syntactical) complexity of ontology modules. We introduced a new ontology measure called expression complexity and adopted Halstead metrics and Henry-Kafura complexity for ontology evaluation. From the graph representation of modularized ontologies ONGRAM computes both domain-dependent and domain-independent design metrics at the ontology and class level. Moreover, we showed that graph clustering evaluation (GCE) metrics, similarly as for software systems, can be used to evaluate cohesiveness of ontology modules. GCE metrics are also included in the metric suite computed by ONGRAM. Therefore, ONGRAM is able to form networks representing ontological descriptions where nodes are attributed with a rich set of metrics.

Using ONGRAM we extracted and studied ontology networks associated to SWEET, a publicly available modularized ontology created by NASA, which describes terminology for earth and environmental sciences. We investigated properties of the SWEET ontology module, class and subsumption networks. Firstly, we showed that all three networks have either single or giant weakly connected component exhibiting the small-world property. However, at the different levels of abstraction SWEET networks possess different patterns of assortativity and strong connectivity. Using the metric-based comparison test we showed that the SWEET ontology module network contains a strongly connected core encompassing the most important and the most internally reused modules.

Secondly, the results of the degree distribution analysis revealed that examined networks are not scale-free. However, analyzed networks contain hubs – highly coupled ontology modules and classes. Similarly as for software systems, highly coupled ontological entities in SWEET are caused by internal reuse, not by internal aggregation. In order to determine characteristics of hub modules we again employed the metric-based comparison test. The results of the test revealed that highly coupled modules tend to be more voluminous, more central and more important compared to non-hub modules.

Thirdly, we investigated correlations between ontology metrics. We showed that there are no strong correlations between GCE metrics and metrics of internal density and connectedness that evaluate cohesiveness of ontology modules relying only on internal dependencies. Strong correlations are also not observed between expression complexity and other metrics of internal complexity. This means that the metrics introduced in this thesis are actually useful since other metrics from the same categories cannot point to ontology modules that have either low or high values of GCE metrics and expression complexity.

In this thesis we demonstrated that the network-based analysis of a modularized ontology enriched with a set of ontology metrics can help us to understand how knowledge from a certain domain is actually modularized and to assess the quality of that modularization. Therefore, in our future work we will apply the same methodology to other modularized ontologies in order to examine possibilities for its widespread adoption in evaluation of modularized ontologies.

Chapter 5

Co-authorship networks

This chapter of the dissertation discusses methods for extraction and analysis of co-authorship networks. The formal definition of co-authorship networks is given in Section 5.1. In Section 5.2 we provide an overview of existing methods for co-authorship network extraction. An overview of results related to analysis of co-authorship networks is presented in Section 5.3. Section 5.4 presents an original contribution of the dissertation. Namely, we studied the co-authorship network extracted from the bibliographical records contained in the electronic library of the Mathematical Institute of the Serbian Academy of Sciences and Arts (eLib) [Savić et al., 2015; Savić et al., 2014]. The nature of the bibliographic data enabled us to investigate the structure of scientific collaborations characteristic to authors that publish papers in Serbian mathematical journals. The bibliographic records also cover a wide time range starting from 1932. Thus, we investigated the evolution of the eLib co-authorship network in an 80 year period, from 1932 to 2011, with yearly resolution in order to observe general trends in the evolution of collaborations among authors from the eLib community.

5.1 Formal definition of co-authorship networks

Formally speaking, co-authorship network is an undirected, weighted and attributed graph $G = (V, E)$ where V represent researchers who published at least one paper. Each link $A \leftrightarrow B$ in the set of links E denotes that A and B authored at least one paper together, with or without other co-authors. Link weights express the strength of collaboration between connected researchers. Three weighting schemes are commonly used:

1. *Normal weighting scheme* [Batagelj and Cerinšek, 2013] where two researchers are connected by a link of weight w if they coauthored exactly w different research papers.

2. *Newman's weighting scheme* [Newman, 2004c] considers the total number of authors per joint papers. Let J be the set of papers jointly authored by A and B. Then A and B are connected by a link of weight w that is determined by the following formula

$$w = \sum_{k \in J} \frac{1}{n_k - 1},$$

where n_k is the number of authors of paper k.

3. *Salton's weighting scheme* [Lu and Feng, 2009] assigns weight w to the link that connects researchers A and B using the following formula

$$w = \frac{h_{A,B}}{\sqrt{h_A \cdot h_B}},$$

where $h_{A,B}$ is the number of papers authored by both A and B, h_A the number of papers authored by A, and h_b the number of papers authored by B. In other words, the scheme is a normalized variant of the normal weighting scheme ($0 < w \leq 1$).

Nodes can have attributes that express different characteristics of authors such as productivity (in terms of the number of published papers), impact (in terms of the number of citations to papers they published), career longevity (time span, the time passed from the publication of first to the last paper), etc.

5.2 Extraction of co-authorship networks

Co-authorship networks can be constructed in several ways. However, the most convenient way to construct a co-authorship network is to extract it from a set of bibliographic records provided by digital libraries or bibliographic databases. Other methods, such as interviews or circulating questionnaires, require much human effort and time, and usually result in a network that contains no more than a few tens or hundred of nodes [Newman, 2004c] making the analysis of scientific collaboration less statistically accurate. The development of the World Wide Web enabled creation of massive bibliographic databases typically through aggregation of publication metadata. Bibliographic databases are extremely important in scientific communities since they give scholars ability to search and discovery publications relevant for their work in a centralized and more systematic fashion. If they additionally provide the full-text accessibility of indexed content then we call them digital libraries.

One bibliographic record represents and provides crucial information about one bibliographic unit. Three are three types of bibliographic databases:

- *Article-centered.* In article-centered databases each article has an unique identifier. Authors of an article are identified by their names. Articles are usually grouped by publication venues which are also uniquely identified. The eLib digital library studied in this thesis is a typical example of an article-centered bibliography database.

- *People-centered.* In people-centered databases each author has an unique identifier to which a list of his/her articles is associated.

- *People-article-centered.* In those databases each publication and each author (individual) have unique identifiers.

Extraction of co-authorship networks from bibliographic databases that are both people- and article-centered is a straightforward task. However, people-article-centered bibliography databases are hard to maintain. To the contrary, article-centered and people-centered databases are much easier to maintain, but the extraction of co-authorship networks from those types of bibliographic databases poses several difficulties. In the case of article-centered databases, where authors are identified by names, the author name disambiguation problem appears and can be manifested in two different forms:

- *Name homonymy*: many different individuals can have the same name.
- *Name synonymy*: a single individual may appear under different names in bibliographic records due to orthographic and spelling variants, spelling errors, transliteration, pen names, a name can change in time (e.g. due to marriage), etc.

For people-centered databases there is a problem of the boundary of co-authorship networks. A researcher registered in a people-centered bibliographic database can have publications that are joint works with researchers who are not registered in the database. Therefore, the question is whether to include unregistered researchers in the network. If such researchers are included then the author name disambiguation problem appears. A co-authorship link between two registered researchers is established if there is at least one publication associated to both of them. However, the same publication may appear in different forms due to different citation conventions, spelling errors, different or missing information, etc. As showed by Radovanović et al. [2007] and Mena-Chalco et al. [2014] the citation synonymy problem can be efficiently handled using string similarity measures.

We made a literature review in order to observe how the name disambiguation problem is approached in studies that deal with analysis of co-authorship networks. We employed a manual, citation guided, snowball sampling procedure using the Google Scholar service to collect relevant papers starting from the paper by Newman [2001b] which is the most cited paper in the field. In this way we obtained exactly 76 studies which is large enough body of research works to observe general trends in the extraction of co-authorship networks in practice. To our surprise in 31 papers the name disambiguation problem is not discussed at all, although in the majority of cases the networks were extracted from bibliography databases that are not people-article centered. In 11 studies the name disambiguation problem is mentioned as important, in some cases examples of ambiguous names are given, but those studies do not clearly show how the problem was systematically tackled. In other studies the problem was approached in the following ways:

- through a manual inspection of data,
- using simple initial-based methods, and
- using author similarity heuristics.

Only in one study the problem was approached using more advanced, machine learning techniques. In the following subsections of this chapter, we will give a brief overview of aforementioned approaches to the name disambiguation problem since it is highly related to our particular case study.

5.2.1 Initial-based approaches to name disambiguation

In initial-based approaches to the name disambiguation problem each author is identified by his/her surname and initial(s) of the first name. There are two basic initial-based methods [Newman, 2001c]:

- *The first initial method.* In this method each author is identified by his/her surname and the first initial only. As emphasized by Newman, this approach does not take the name homonymy problem into account at all and it is "clearly prone to confusing two people for one, but will rarely fail to identify two names which genuinely refer to the same person". The method efficiently handle name synonyms when spelling errors and other inconsistencies occur in the first names, but it is sensitive to spelling errors in the last names.

- *All initials method.* In the all initials approach each author is identified by his/her surname and all initials. This method treats authors with the same surname and the first initial but different middle initials as different individuals. Although it seems that this method is more accurate that the first initial method, it will identify a researcher who is inconsistently reporting his middle initial as more different individuals.

Above mentioned initial-based methods for name disambiguation are used in several studies of co-authorship networks [Acedo et al., 2006; Barabasi et al., 2002; Bettencourt et al., 2009; Ding, 2011; Goyal et al., 2006; Hou et al., 2008; Newman, 2004a, 2001b,c,d, 2004c; Pan and Saramäki, 2012; Çavuşoğlu and İlker Türker, 2013, 2014]. In some of them, additional manual effort is invested in order to detect "problematic" author names. For example, Acedo et al. [2006] performed a systematic inspection of papers that share the same author name in order to see whether affiliations of an author across papers are identical. For those authors that have multiple affiliations it is manually decided whether they are the same person or not.

In his studies of collaboration in physics, biomedicine, computer science and mathematics [Newman, 2001b,c,d, 2004c], Newman used initial-based methods to construct corresponding co-authorship networks in order to obtain upper and lower bounds of the number of authors and other quantities related to the structure of the networks (such as the mean degree, small-world and clustering coefficient, the size of the largest component, etc.). He observed that there are no drastic differences in structural quantities when they are computed from co-authorship networks obtained using the first initial and all initials method. However, Newman noticed that initial based methods are in particularly sensitive to authors of Japanese and Chinese descent.

Milojević [Milojević, 2013] investigated accuracy of initial-based methods using simulated bibliographic datasets in which true identities of authors are known. She used Thomson Reuters' Web of Science service to form five real-world bibliographic datasets corresponding to five disciplines. For each dataset she estimated the number of authors, the frequency distributions of last names, first and middle initials, the intrinsic rate and the reporting rate of middle initials, and the distribution of author productivity where the credit for a paper is given only to the first author. Milojević then generated artificial datasets to mimic previously mentioned statistical properties of real datasets, and performed author name disambiguation using initial-based methods. The results showed that the first initial method correctly identifies 97% of authors on average and that the all initial method is typically less accurate than the first initial method. She also proposed a hybrid initial-based method that takes the last name frequency and the size of the dataset implicitly into account. The conducted experimental investigation of the hybrid method showed that it outperforms Newmans's initial based methods.

Contrary to the findings reported by Milojević [Milojević, 2013], the studies by Fegley and Torvik [2013] and Kim et al. [2014] indicate that the initial-based methods significantly distort statistical properties of co-authorship networks. Fegley et al. [Fegley and Torvik, 2013] investigated the effects of splitting (one person represented by two or more nodes in a co-authorship network due to name variants) and lumping (more than one person represented by one node in the co-authorship network due to common names) on statistical properties of two large-scale collaboration networks extracted from the MEDLINE bibliography database and the USPTO patent database. The authors extracted and compared three different variants of mentioned co-authorship networks: one where author names were disambiguated using the Authority disambiguation approach [Torvik and Smalheiser, 2009; Torvik

et al., 2005], one where the first initial method was used to identify authors, and one where the all initials method was used to identify authors. The main conclusions of the study are that:

- Initial-based disambiguation methods drastically underestimate the number of authors compared to a more advanced name disambiguation technique.
- Name homonyms induced by initial-based methods drastically change the structure of examined co-authorship networks. Lumping effects caused by initial-based identification of authors is reflected by a significantly higher average degree and significantly smaller clustering, small-world and assortativity coefficient.

The similar findings were also reported by Kim et al. [2014] who used the DBLP dataset in their experiments. The differences in the conclusions reported in [Milojević, 2013] and the conclusions reported in [Fegley and Torvik, 2013; Kim et al., 2014] clearly indicate that the accuracy of initial-based methods is still an open research question and demands a more comprehensive analysis.

5.2.2 Heuristic approaches to name disambiguation

There are also several simple heuristic approaches to the author name disambiguation problem. The main characteristic of those techniques is that they are based on a simple and easily implementable similarity measures or matching functions/rules considering a minimal set of features. Compared to initial-based approaches those methods are far more sophisticated but simpler than machine learning approaches.

Moody [Moody, 2004] proposed a name disambiguation method in which first and last names of authors are classified as either common or uncommon based on their frequencies. A predefined threshold of 15 appearances is used to mark a first/last name as common. Two different name labels are considered as the same person if the following two conditions are satisfied:

1. Name labels differ only in the middle initial in the sense that in one name label the middle initial is present while in the other is missing.
2. Either first name or last name is uncommon.

When a co-authorship network is extracted then it is checked whether there are pairs of authors who have the same sets of co-authors and whose name labels differ only in the middle initial (in the same sense as in the first condition above). Such pairs of authors are also considered as the same persons.

An approach similar to Moody's were used by Chen et al. [Chen et al., 2013] to construct the co-authorship network of authors publishing in the *Scientometrics* journal. Namely, two authors whose name labels differ only in the middle initial are considered as the same persons if they are affiliated with the same institution.

Bird et al. [2009a] investigated the structure and dynamic of research collaboration in several computer science disciplines using co-authorship networks extracted from the DBLP bibliography database. The authors emphasized that the DBLP data is fairly accurate (due to the massive human effort invested in the maintenance of the database) but that still suffers to some degree from the name disambiguation problem. The authors used several heuristics such as string similarity of author names, the number of co-authors in common, the number of publication venues in common and dates of publications to identify pairs of names that are likely to be the same authors. The candidates obtained by previously mentioned heuristics are manually examined in order to identify name synonyms. String

similarity measures were also used to identify author name synonyms in the extraction of the co-authorship network of the ED-MEDIA conference [Ochoa et al., 2009].

Martin et al. [2013] investigated co-authorship and citation patterns in Physical Review journals. They used a name disambiguation approach that relies on the similarity of author names, collaboration patterns and institutional affiliations. In contrast to the previously described name disambiguation approaches, the starting assumption of the proposed method is that each author of each paper is a different individual. This means that two authors with identical names are not initially treated as the same person. All authors that have the same name and at least one shared affiliation are treated as one person. Then author pairs with similar names are identified. Two authors are considered as similar if they have the same last name and compatible first/middle names (identical if fully given, or compatible initials). For similar authors a further similarity measure based on the number of shared affiliations, the number of shared co-authors and the number of joint publication venues is computed. Again two authors with a high similarity are treated as the same person.

5.2.3 Machine learning approaches to name disambiguation

A survey of the state of the art machine learning approaches to author name disambiguation is given in the article by Ferreira et al. [2012]. The authors observed that the main characteristic of the majority of existing techniques is that they try to group references authored by a same author. Reference grouping methods rely on some reference similarity function that is used by some clustering technique to form clusters of papers where one cluster correspond to one author. This means that a similarity function has to quantify to what extent two references are authored by the same person. Each reference can be represented by a feature vector whose elements are reference attributes such as author names and affiliations, publication title, year of the publication, publication venue, keywords, classifications, etc. The similarity between two references is then quantified by a similarity vector where each element of the vector indicate similarity between corresponding reference attributes. Reference similarity functions are either predefined (e.g. string similarity measures and name comparison functions), learned (using classification learning techniques) or graph-based (co-authorship network itself is used to determine similarity between two author name labels). In contrast to reference grouping methods, there are also author assignment methods that construct probabilistic models representing authors (i.e. models that answer what is the probability that author A is author of reference R which is characterized by a set of attributes) using either supervised classification or model-based clustering techniques.

Machine learning approaches to the name disambiguation problem are rarely used in extraction of co-authorship networks that are later analyzed with the aim of revealing patterns and trends in research collaboration. In our literature review that covers more than 70 research articles on the subject we identified just one article where machine learning techniques were employed to disambiguate author names. Huang et al. [2008] studied the evolution of the co-authorship network of computer scientists that was extracted from the bibliography records contained in the CiteSeer digital library. To disambiguate author names the authors used a name disambiguation method they previously proposed in [Huang et al., 2006]. The main characteristics of their method are:

- The DBSCAN clustering algorithm [Ester et al., 1996] is used to form clusters of references.

- A similarity function used by the clustering algorithm is learned by LASVM [Bordes et al., 2005] – an online learning SVM (support vector machines) algorithm. The method is supervised which means that a training set used to build the similarity function has to be manually formed.

- Different string similarity measures are used for different attributes: the edit distance for emails and URLs, the token-based Jaccard similarity for affiliations and addresses and the Soft-TFIDF string similarity measure for names.

5.2.4 Author identification in massive bibliography databases

The identification of authors based on the strict matching of name labels can be used safely when co-authorship networks are extracted from bibliography databases that are people centered. Our survey of co-authorship networks extraction showed that this type of name identification was used when networks were extracted from two massive bibliography databases – Mathematical Reviews and DBLP. Therefore, in this Section we will review the processes of author identification and disambiguation that are employed in the maintenance of these two databases.

Mathematical Reviews. Mathematical Reviews (MR, MathSciNet) is a subscription-based bibliography database maintained by the American Mathematical Society that indexes mathematical books and articles published in peer-reviewed journals. At the moment (October 2014) this database covers over 2000 journals and contains bibliography records for more than 3 million publications authored by nearly 730000 mathematicians. MR is both people-centered and publication-centered bibliography database. For each author, MR maintains an author profile – record that contains all known name variants of the author, institutional affiliation(s), classifications contained in his/her papers, links to profiles of co-authors, and references to cited articles indexed in the database[1]. In the beginning the identification of authors was done entirely by hand [TePaske-King and Richert, 2001]. After 1985, the process of author identification was semi-automated using author matching algorithms. When a new article has to be indexed then for each author of the article a multi-criterion matching against existing author profiles is performed. The matching procedure considers three elements: author name, institutional affiliation and classifications of the article assigned by the MR editors. Unfortunately, the details of the procedure cannot be found in the MR documentation, nor they are documented in relevant scientific articles. As reported in [TePaske-King and Richert, 2001], in roughly eighty percent of cases an unique matching author profile can be found. In the rest of cases the matching algorithm ranks obtained candidates that are manually inspected in order to determine whether a new author profile has to be created.

DBLP. DBLP is a computer-science bibliography database developed by a research group from University of Trier led by professor Michael Lay. Currently, this database contains information about more than 2.7 million publications written by nearly 1.5 million researchers. DBLP is both people-centered and publication-centered database, but DBLP does not grantee that an author profile correspond to exactly one individual nor that one individual is represented by exactly one author profile [Ley, 2009]. In order to increase the quality of data (decrease the number of appearances of author homonyms/synonyms) DBLP employs two strategies [Ley, 2009; Reuther et al., 2006]:

- Different similarity measures are applied on blocks of authors in order to detect possible synonyms. For a huge amount of authors it is computationally intractable to determine similarity considering the whole set of author pairs. Therefore, similarity measures are computed on a

[1] http://www.ams.org/publications/math-reviews/mr-authors

subset of all author pairs that is obtained by a blocking function. DBLP uses the following blocking functions: (1) distance in the DBLP co-authorship network, (2) presence in the same publication venue, and (3) the same keywords in publication titles. This means that similarity measures are computed for author pairs that have distance smaller than two in the DBLP co-authorship network, author pairs who have published in the same journal/conference series and author pairs who have articles that contain the same rare word in their publication titles. DBLP maintainers implemented more than 20 similarity functions which are either classical string similarity functions/string edit distances, classical graph-based similarity functions (such as the number of co-authors in common) and the combinations of the previous two types of similarity measures.

- The disconnected co-authors heuristic is applied on the ego-network of an author in order to detect possible homonyms. If there are two distinct persons that are represented by one node in the co-authorship network due to the same name then it is highly probable that the node is an articulation point in its ego-network (network induced by the node and the nearest neighbors). In other words, the removal of the node from the ego-network will cause the fragmentation of the ego-network into several disjoint connected components that represent unrelated groups of researchers. If the node is not an articulation point in its ego-network then we can be quite confident that the node represents a single person. When an author name homonymy is detected and confirmed then the author profile is split and "mystical" numbers are appended to the name in order to make the homonym persons distinguishable.

5.3 Analysis of co-authorship networks

Existing empirical studies of co-authorship networks cover a wide range of scientific disciplines. Namely, collaboration between researcher using co-authorship networks was investigated for physics [Barrat et al., 2004; Newman, 2001b,c,d, 2004c; Pan and Saramäki, 2012; Ramasco et al., 2004], mathematics [Barabasi et al., 2002; Batagelj and Mrvar, 2000; Brunson et al., 2014; Cerinšek and Batagelj, 2014; Grossman, 2002a,b], computer science [Bird et al., 2009a; Biryukov and Dong, 2010; Divakarmurthy and Menezes, 2013; Elmacioglu and Lee, 2005; Franceschet, 2011; Huang et al., 2008; Newman, 2001b,c,d, 2004c; Shi et al., 2011; Staudt et al., 2012], biomedicine [Newman, 2001b,c,d, 2004c], economy [Goyal et al., 2006], management [Acedo et al., 2006], library and information science [Abbasi et al., 2012a; Yan and Ding, 2009], and sociology [Moody, 2004]. The structure and evolution of scientific collaboration was also investigated for a more narrower disciplines encompassing researchers working on specific research topics such as genetic programming [Luthi et al., 2007; Tomasini and Luthi, 2007; Tomassini et al., 2007], evolutionary computation [Cotta and Guervós, 2007; Cotta and Merelo, 2005], computational geometry [Hui et al., 2011], information retrieval [Ding, 2011], information systems [Zhai et al., 2014], information fusion [Johansson et al., 2011], intelligence in computer games [Lara-Cabrera et al., 2014], information visualization [Börner et al., 2005], social network analysis [Otte and Rousseau, 2002], econophysics [Fan et al., 2004] and steel structures [Abbasi et al., 2012b; Uddin et al., 2012, 2013]. There are also studies which examined properties of co-authorship networks of scientific, mostly computer science, conferences [Cheong and Corbitt, 2009a,b; Hassan and Holt, 2004; Liu et al., 2005; Nascimento et al., 2003; Ochoa et al., 2009; Pham et al., 2012; Reinhardt et al., 2011; Smeaton et al., 2003; Vidgen et al., 2007; Xu and Chau, 2006]. There is also a body of scientific works that revealed co-authorship patterns in individual publication venues by investigating

properties of corresponding journal co-authorship networks [Borner et al., 2004; Chen et al., 2013; Fatt et al., 2010; Fischbach et al., 2011; Hou et al., 2008; Li et al., 2010; Martin et al., 2013].

Perhaps the most influential studies related to analysis of field co-authorship networks are those authored by Mark E. Newman [Newman, 2001b,c,d, 2004c][2]. Using publication metadata from four web accessible databases of scientific papers (Los Alamos e-Print Archive, MEDLINE, SPIRES and NCSTRL) Newman extracted and studied co-authorship networks representing research collaboration in physics (Los Alamos Archive, SPIRES), biomedical research (MEDLINE), and computer science (NCSTRL). Moreover, Newman subdivided data from Los Alamos Archive into three sub-disciplines within physics (astro-physics, physics of condensed matter and theory of high energy physics) thus investigating seven co-authorship networks in total.

Firstly, Newman observed that the distribution of the number of papers per author in each examined network closely follows either pure power-law or truncated power-law (power-law with exponential cut-off). In other words, there is a big variation in scientific productivity among authors in a research field: the majority of authors publish very small number of papers (close to the average number of papers per author), but there is also a small portion of authors whose scientific productivity is significantly larger than the average. However, it is important to emphasize that power-laws in the number of papers per author were observed (by hand) way back in the early 20th century by Lotka [Lotka, 1926] (Lotka's law of scientific productivity) and confirmed by the subsequent (computerized) studies [Pao, 1986; Voos, 1974]. Newman also noticed that the distributions of the number of authors per paper can be also very well approximated by power-laws. This result was explained by a large experimental collaboration in scientific fields covered by the study. Similarly to the distributions of the number of papers per author (distribution of productivity) and the number of authors per paper, empirically observed distributions of the number of collaborators per author (degree distributions of co-authorship networks) do not have characteristic scale, i.e. they are heavy-tailed in the sense that the tails of those distributions are not exponentially bounded. However, only for the SPIRES co-authorship network the degree distribution closely follows a power-law. For other networks investigated by Newman degree distributions obey either power-laws with exponential cut-off or two regime power-laws where degree distribution initially follows a power-law with one scaling exponent and then in the tail changes to a power-law with a different scaling exponent. Heavy tailed degree distributions of co-authorship network were reported in many later studies [Barabasi et al., 2002; Bird et al., 2009a; Borner et al., 2004; Cotta and Merelo, 2005; Divakarmurthy and Menezes, 2013; Elmacioglu and Lee, 2005; Fatt et al., 2010; Franceschet, 2011; Grossman, 2002b; Huang et al., 2008; Hui et al., 2011; Li et al., 2010; Liu et al., 2005; Martin et al., 2013; Mena-Chalco et al., 2014; Nascimento et al., 2003; Ochoa et al., 2009; Ramasco et al., 2004; Tomassini et al., 2007; Xu and Chau, 2006; Yan et al., 2010; Çavuşoğlu and İlker Türker, 2013] suggesting that big variations in collaborative potential of researchers is a typical feature of research communities.

Secondly, Newman observed that all examined networks have a giant connected component, the component that encompasses the majority of authors. In all cases, except for the network of high-energy theory and computer science, the size of the giant connected component is larger than 80% of the total number of authors. For high-energy theory and computer science the largest connected components encompasses 71.4% and 57.2% of the total number of authors, respectively. Newman explained that the size of the largest connected component in those two networks can be considered

[2]Studies [Newman, 2001c] and [Newman, 2001d] present different aspects of analysis of the same set of co-authorship networks. Those studies are an extended version of [Newman, 2001b]. On the other hand, [Newman, 2004c] is the aggregated version of [Newman, 2001c] and [Newman, 2001d].

as an "anomaly" due to a poorer coverage of the corresponding bibliography databases. Also, Newman stressed out the importance of the existence of giant connected components in co-authorship networks. Giant connected components indicate that the corresponding research discipline as a whole is a product of joint rather than many isolated efforts. On the other hand, the absence of a giant connected component implies a poorly cohesive community of researchers and suggests relative immaturity of the field or limited cross institutional collaboration [Liu et al., 2005].

Similarly to complex networks from other domains, Newman observed that examined co-authorship networks possess the small-world property [Watts and Strogatz, 1998a] which is reflected by short distances between randomly selected authors and a high degree of local clustering (much higher than in comparable random graph). Several subsequent studies confirmed that the small-world property is a typical characteristic of co-authorship networks [Borner et al., 2004; Cheong and Corbitt, 2009a,b; Cotta and Merelo, 2005; Divakarmurthy and Menezes, 2013; Elmacioglu and Lee, 2005; Fatt et al., 2010; Franceschet, 2011; Grossman, 2002b; Hassan and Holt, 2004; Huang et al., 2008; Hui et al., 2011; Johansson et al., 2011; Kronegger et al., 2012; Liu et al., 2005; Mena-Chalco et al., 2014; Nascimento et al., 2003; Perc, 2010; Tomassini et al., 2007; Xu and Chau, 2006; Yan et al., 2010; Çavuşoğlu and İlker Türker, 2013]. High degree of local clustering in co-authorship networks Newman explained by three factors:

- Two colleagues that have a co-author in common are introduced to each other by the joint co-author.
- Three colleagues that have no co-author in common are positioned in the same scientific circle (read the same journals, attend the same conferences) and as a result of similar research interests start to collaborate either together or in pairs making a transitive collaboration closure.
- Three colleagues that have no co-author in common work at the same institution/department and due to the geographical proximity establish a collaboration.

Newman also investigated betweenness centrality of authors in co-authorship networks showing that they exhibit so called *funnelling effect*. Namely, for most of authors there are only a few collaborators that lie on the most of the shortest paths connecting the author with the rest of the network. Such collaborators are also called "sociometric superstars". This means that all co-authors of an author are not equally important to the connectivity of the author to other researchers. In other words, collaboration with just one or two famous or influential members of a scientific community would result in shorter distances to other people from the community. In his subsequent study [Newman, 2004a], Newman also investigated properties of the co-authorship network extracted from the "Mathematical Reviews" database which shows collaboration in mathematics and confirmed conclusions from his previous studies. However, in [Newman, 2004a] Newman made two additional important contributions. Namely, he showed that:

- Co-authorship networks tend to exhibit slight assortative mixing, i.e. there is a slight tendency that highly connected authors (authors having high number of collaborators) collaborate among themselves. Subsequent studies also reported the presence of assortative mixing in other co-authorship networks [Bird et al., 2009a; Franceschet, 2011; Huang et al., 2008; Mena-Chalco et al., 2014; Ramasco et al., 2004; Tomassini et al., 2007].
- On a relatively small co-authorship network representing research collaborations at the Santa Fe Institute Newman applied the Girvan-Newman community detection algorithm showing that the

network possesses clear community structure where each of detected communities corresponds to one research division at the institute and encompasses researchers dealing with the same research topic. Several subsequent studies showed that community organization is a typical feature of co-authorship networks [Batagelj and Mrvar, 2000; Bird et al., 2009a; Cotta and Guervós, 2007; Donetti and Muñoz, 2004; Duch and Arenas, 2005; Farkas et al., 2007; Gregory, 2007; Hui et al., 2011; Johansson et al., 2011; Lara-Cabrera et al., 2014; Leskovec et al., 2009; Liu et al., 2005; Luthi et al., 2007; Lužar et al., 2014; Newman, 2004b; Pons and Latapy, 2006; Raghavan et al., 2007; Staudt et al., 2012; Yang and Leskovec, 2014].

Following the work of Newman, Barabási and co-authors [Barabasi et al., 2002] investigated the evolution of co-authorship networks describing scientific collaboration in mathematics and neuroscience in the period from 1991 to 1998. The main empirical observations of the study are:

- Degree distributions of examined networks follow power-laws.
- Average separation (distance, length of the shortest path) between authors decreases in time. This evolutionary trend, which indicates that a co-authorship network evolves into a more compact state ("small-world" gets smaller over time), was also observed for many other co-authorship networks [Franceschet, 2011; Goyal et al., 2006; Huang et al., 2008; Mena-Chalco et al., 2014; Perc, 2010; Çavuşoğlu and İlker Türker, 2013].
- Clustering coefficient of the network also decreases in time.
- Relative size of the largest cluster increases in time.
- Average degree (average number of co-authors) increases in time. Other studies of co-authorship networks also reported a definite trend toward increasing collaboration for various research communities [Bettencourt et al., 2009; Chen et al., 2013; Elmacioglu and Lee, 2005; Franceschet, 2011; Goyal et al., 2006; Grossman, 2002a; Martin et al., 2013; Mena-Chalco et al., 2014; Pham et al., 2012; Tomasini and Luthi, 2007; Zhai et al., 2014; Çavuşoğlu and İlker Türker, 2013].

However, the most important contribution of [Barabasi et al., 2002] is that the authors showed that the integration of new actors (new researchers) in co-authorship networks is governed by the preferential attachment principle which is the one of the ingredients of the scale-free model of complex networks [Barabasi and Albert, 1999]. Naturally, co-authorship networks evolves by addition of new authors and new collaboration links in the network. Barabási et al. showed that a probability that a new researcher establishes a collaboration link to an "old" author, author that is already present in the network, is proportional to the degree centrality of the old author. Moreover, they showed that a probability of a new collaboration link between two old authors is proportional to the product of their degree centralities. Relying on the evidence of preferential attachment for both external (links between a new and old authors) and internal links (links between old authors), Barabási et al. proposed a simple model of complex networks that can explain basic structural characteristics such as heavy-tailed degree distributions as well as observed dynamics of co-authorship networks. The evidence that the preferential attachment principle governs the evolution of large-scale co-authorship graphs was also given by Newman [2001a], Tomasini and Luthi [2007], Lara-Cabrera et al. [2014] and Perc [2010]. Using the co-authorship networks he previously investigated in [Newman, 2001b] Newman also showed that:

- the probability that two researcher establish collaboration increases with the number of their common co-authors, and
- the probability that a researcher acquire a new co-author increases with the number of his/her co-authors.

The co-authorship network of Los Alamos e-Print Archive examined by Newman was also empirically studied by Barrat et al. [2004] together with complex weighted networks from other domains. The authors showed that the distribution of collaboration strength (i.e. the distribution of link weights) is, similarly to the degree distribution of the network, heavy-tailed implying that there is a small, but statistically significant, portion of extremely frequent collaborators. The authors also proposed modifications of pure topological measures that take the weight of links into account (weighted clustering coefficient and weighted average nearest-neighbors degree) concluding that "the study of correlations between weights and topology provide a complementary perspective on the structural organization of the network that might be undetected by quantities based only on topological information".

Bettencourt et al. [2009] mapped the evolution of collaboration networks of eight scientific fields (superstring theory, cosmic strings, cosmological inflation, carbon nanotubes, quantum computing, prions and scarpie, H5N1 influenza and cold fusion which is marked as an example of "pathological" scientific field) over time, from their inception to maturity. The main thesis of [Bettencourt et al., 2009] is that "the creation and spread of new discoveries through a scientific community creates qualitative, measurable changes in its social structure". Therefore, the authors investigated changes in structure of co-authorship networks by observing evolution of several measures such as node-edge ratio, diameter and the size of the largest connected component. The general conclusions of the study are:

- Co-authorship networks "densify" in time which means that there is an increase in the average number of links per node as networks evolve. The authors showed that the densification of examined co-authorship networks can be described by a simple scaling law previously empirically observed by Leskovec et al. [2005, 2007] for large-scale networks from other domains:

$$E(t) = c \cdot N(t)^{\alpha},$$

  where $E(t)$ and $N(t)$ are the number of edges and nodes in the network at time t, c is a normalization constant, while α is the constant scaling exponent of the law. For each examined field, except for cold fusion ("pathological" field), $\alpha > 1$ which means that the number of links (collaborations) in the network grows at a faster rate than the number of nodes (researchers).

- There is a topological transition from a fragmented structure of the network characterized by several disjoint and small connected components in the early stage of evolution to a giant connected component in all fields except for the "pathological" field.
- Diameters of co-authorship networks initially grow very fast and when giant connected components emerge diameters tend to have a stable value.

The authors make a parallel between the Kuhn's theory of the structure of scientific revolutions [Kuhn, 1970] and observed topological transition in successful, non-pathological scientific fields. According to Kuhn, successful scientific fields arise from discovery (novelty of fact, "anomaly" that cannot be explained by the currently dominant theory) and invention (novelty of theory which successfully

explains previously observed "anomalies"). Discoveries usually involve small and independent groups of researchers which means that the corresponding co-authorship network possesses a fragmented structure. This phase is characterized by several incompatible and incomplete theories regarding the discovery. However, if the fragmented scientific community reaches the consensus regarding the explanatory potential of one of competitive theories, then the process of large-scale adoption of the theory starts which inevitably leads to a widespread collaboration in the field. This widespread collaboration naturally ensures the emergence of a giant connected component in the corresponding co-authorship network.

Finally, several studies reported that there are strong correlations between centrality measures and impact of researchers [Abbasi et al., 2012a; Fischbach et al., 2011; Yan and Ding, 2009; Yan et al., 2010]. This means that authors with a high number of co-authors or authors located in the core of the co-authorship network tend to be cited more compared to others.

5.3.1 Co-authorship networks of mathematicians

The body of work most relevant to our case study involves collaboration networks in the field of mathematics. Research of these networks has been deeply influenced by the existence of one prominent mathematician, Paul Erdős (1913 – 1996), whose unique work ethic and lifestyle led to the publication of over 1500 papers with a great number of different co-authors [Grossman, 2013]. The notion of Erdős number was formally defined by Goffman [1969]. In response to Goffman's paper, several mathematicians indicated some interesting facts regarding Erdős number, even Erdős himself in a sarcastic tone speculated on the properties of the Erdős collaboration network [Erdős, 1972]. More serious studies of the Erdős co-authorship network were conducted by Grossman and Ion [1995] and Batagelj and Mrvar [2000]. More general analysis of mathematics collaboration networks were performed by Grossman [2002a,b] who examined statistical properties of the network derived from Mathematical Reviews (MR), and Brunson et al. [2014] in 2014, who studied the evolution of the MR network, identifying two points of drastic reorganization of the network, as well as increased collaboration between mathematics researchers in more recent times. Cerinšek and Batagelj [2014] in 2014, as a part of more general study of various two-mode bibliography networks, investigated the co-authorship network derived from the Zentralblatt database.

Grossman and Ion [1995] examined the basic properties of the network that shows collaboration among Erdős and his co-authors. They observed that a small portion of Erdős co-authors (9%) never collaborated with at least one other mathematician having Erdős number 1. Secondly, they investigated the evolutionary trends in mathematical production using the MR database. The results showed that the number of solo-authored papers constantly decays in time (from over 90% in 1940 to less than 60% in 1995). Consequently, there is a steady increase in the number of publications having two or more authors in the same time frame.

Batagelj and Mrvar [2000] used the truncated Erdős collaboration network to present different techniques for the analysis of large-networks that include different approaches for the identification of important individuals and cohesive subgroups in the network. Truncated Erdős collaboration network is obtained by removing Paul Erdős from the co-authorship network that encompasses Erdős, his co-authors, and co-authors of Erdős's co-authors (mathematicians having Erdős number smaller or equal to 2). The authors observed that the truncated Erdős collaboration network contains a giant connected component that encompasses 99% of the total number of mathematicians in the network. Various clustering techniques (prespecified block-modelling and Ward's hierarchical clustering) were

applied to the core of the giant connected component to uncover cliques and cohesive subgroups in the network. The authors also introduced a measure of collaborativeness that quantifies the openness of an author towards authors located on the periphery of the network and used it to identify the most collaborative Erdős co-authors.

Grossman also investigated the structure [Grossman, 2002b] and evolution [Grossman, 2002a] of the co-authorship network that was constructed from the MR bibliography records which cover the period from 1940 to 2002. He found out that the network has a giant connected component encompassing more than 60% of the total number of authors, the network is a small-world with eight degrees of separation, and has clustering coefficient extremely larger than that of a comparable Erdős-Renyi random graph (more than 10000 time higher). Secondly, Grosmann observed that the network has the scale-free property (the degree distribution of the network follows a power-law with exponential cut-off). Additionally, the distribution of component sizes also can be characterized by a power-law. Finally, Grossman observed that the average degree (the average number of collaborators per author) increases as the network evolves (from 0.49 in 1940 to 2.94 in 2002) suggesting that investigations in mathematics transformed from being mostly individual efforts to being mostly collaborative efforts.

Brunson et al. [2014] also investigated the evolution of the co-authorship network associated to the MR database observing long-term trends and shifting behaviour in mathematical collaboration. In contrast to Grossman [2002a] who investigated the evolution of the network accumulatively, the authors examined the evolution of the network using a fixed-duration sliding window. This means that for each year in the examined time frame (1990-2009) only publications from previous 5 years were taken into account when constructing an evolutionary snapshot of the network. Additionally, the authors studied two sub-networks of the network: the first one constructed from publications that can be considered as research works in "pure" mathematics, and the second one built from publications that can be regarded as more "applied" (the classification of the papers into pure and applied was done using the MSC classification scheme). The authors observed that pure and applied mathematicians are positioned differently in the aggregated network: pure researchers tend to located in the core of the network, while applied researchers are mostly located on the periphery. Secondly, the co-authorship network of applied mathematicians is less cohesive than the network of pure mathematicians. Regarding the evolution of the network, the authors observed two points of the drastic reorganization:

- The *mid-90s* event that is characterized by noticeable increase of the average degree and clustering coefficient, especially among researchers working on applied mathematics. At the same time there is an decrease in cross-disciplinarity (collaboration between researchers from pure and applied mathematics).

- The *early-00s* event that is characterized by an abruptly decrease of both the average number of publications per author and the average strength of collaborative ties. The authors explained this event by a large influx of mathematicians from the former Soviet Union into the MR database.

Secondly, the authors noticed that the small-world coefficients of all three networks (aggregated, pure, applied) decrease in time, much faster than predicted by the model of random graphs. Simultaneously, the average clustering coefficients increase as networks evolve. However, clustering trends in pure and applied mathematics rely on quite different phenomena: the increase of clustering coefficient in pure mathematics is due to the rising average connectivity among respective mathematicians, while increase of clustering coefficient in applied mathematics is caused by an increase of highly cooperative publications (publications that have a relatively large number of authors).

Cerinšek and Batagelj [2014] performed network-based analysis of the Zentralblatt database. Among other things, such as distribution of the number of papers per number of keywords and MSC classifications, the authors also investigated properties of the Zentralblatt co-authorship network. The main finding of the study regarding this aspect is that the subset of the co-authorship network, in which each author's contribution to joint works with other authors is larger than previously fixed high threshold (30 papers), is dominated by "tandems" – components encompassing exactly two authors. Similarly to the previous studies of co-authorship networks, the authors used the co-authorship network to identify most productive and collaborative authors for the whole database, as well for the subset which cover only researchers working in graph theory.

5.4 Case study: ELib co-authorship network

Digitization of mathematical journals in Serbia started in 1995 as a response to the increasing requirement for easier access to old issues of the journal *Publications de l'Institut Mathématique*. Later on, to enable access to the digitized material and to support preservation of old publications, in the year 2002 the Mathematical Institute decided to create an Internet database of freely accessible full-text mathematical journals – eLib [Mijajlović et al., 2010]. The Electronic Library of the Mathematical Institute of the Serbian Academy of Sciences and Arts is defined as a web-orientated application which contains a collection of mathematical journals and can be searched (both in English and Serbian by: authors' names, titles, titles of special sections within the journals, key words and words contained in abstracts, classification numbers), downloaded and printed.

In time when research was performed ELib digitized articles published in the following Serbian journals:

1. "Publications de l'Institut Mathématique" (PIM), published by the Mathematical Institute of the SASA[3] since 1932,
2. "Matematički Vesnik" (MV), published by the Mathematical Society of Serbia since 1949,
3. "Zbornik Radova" (ZR), published by the Mathematical Institute of the SASA since 1951,
4. "Publications of Department of Astronomy" (PDA), published by the Faculty of Mathematics, University of Belgrade, since 1969,
5. "Nastava Matematike" (NM), published by the Mathematical Society of Serbia since 1992,
6. "The Teaching of Mathematics" (TTM), published by the Mathematical Society of Serbia since 1998,
7. "Visual Mathematics" (VM), published by the Mathematical Institute of the SASA since 1999,
8. "Kragujevac Journal of Mathematics" (KJM), published by the Faculty of Sciences, University of Kragujevac, since 2000,
9. "Bulletin, Classe des Sciences Mathématiques et Naturelles, Sciences mathématiques" (Bulletin), published by the SASA since 2001,

[3]Serbian Academy of Sciences and Arts

10. "Review of the National Center for Digitization" (RNCD), published by Faculty of Mathematics, University of Belgrade, since 2002,

11. "Computer Science and Information Systems" (ComSIS), published by the ComSIS Consortium[4] since 2004.

The bibliographic records contained in ELib were generously provided by the Mathematical Institute of the Serbian Academy of Sciences and Arts. The eLib database was designed primarily with an e-library web application in mind, i.e. it was designed as a backend part of an e-library application. That means that its design was optimized for the needs of the e-library application and as such it is not particularly suited for extensive data analysis. Because of the above, the maintainers of the eLib digital library decided to export subset of database data into a textual file. They developed a procedure that denormalizes data from the e-library database and exports it in a CSV file. In the CSV file each paper is described by the following information: ID (identifier of paper in the e-library database), language the paper is written in, paper title, list of keywords, MSC classification, journal in which the paper was published, year of publication, number of pages, first name, last name and gender of each author. We used the data in the CSV file to extract the eLib co-authorship network.

5.4.1 Extraction of the eLib co-authorship network

The name labels contained in the eLib bibliographic records can be divided into two categories: full names (provided both full first name and last name) and short names (first name is reduced to first letter(s)). The eLib bibliographic records contain 8842 name appearances in total, where 5192 name appearances (58.72%) are full names, while 3650 name appearances (41.28%) are short names. As emphasized by Smalheiser and Torvik [2009]: "There is no single paradigmatic author name disambiguation task – each bibliographic database, each digital library, and each collection of publications has its own unique set of problems and issues". Therefore, we made a preliminary analysis of the eLib bibliographic records in order to observe issues related to the name disambiguation problem characteristic to this particular digital library.

5.4.1.1 Preliminary analysis of data

We constructed a preliminary version of the eLib co-authorship network using strict name matching where each distinct name label uniquely determines one node in the network. This means that we started with the assumption that the name disambiguation problem does not occur in the eLib bibliographic records.

In order to detect potential name homonyms we applied the disconnected co-authors heuristic (the same strategy is used in the maintenance of the DBLP bibliography database, for details please refer to Section 5.2.4). The obtained results showed that only 67 authors are articulation points in their ego networks which is 1.58% of the total number of authors. 55 out of 67 articulation points are identified by full name labels. Based on web searches we have not found any evidence that each of those 55 names corresponds to more than one mathematician. In other words, the number of potential name homonyms is extremely small and it can be freely stated that the name homonymy problem cannot seriously affect the results of network analysis. Therefore, we decided to use a name disambiguation technique based on string similarity measures which considers only the name synonymy problem.

[4]The ComSIS Consortium is a group of leading scientific institutions from universities in Serbia including the Serbian Academy of Sciences and Arts, who jointly publish the ComSIS journal.

Secondly, we decided to attack the problem in a semi-automatic, suggestion-based way where a name disambiguation system suggests ambiguous author name pairs that are latter manually examined. A fully automatic unsupervised approach to the problem would also require a manual effort to check obtained results in order to observe and fix errors since existing unsupervised approaches to the name disambiguation are far from being perfect. Similarly, a fully automatic supervised solution would require a manual effort to create a good training set. On the other hand, realization of a fully automatic approach to the name disambiguation is far more complex task than the realization of a semi-automatic, string similarity based approach.

Finally, we noticed that the preliminary version of the eLib co-authorship networks is extremely fragmented. The network consists of a large number of disjoint connected components where the largest one encompasses only 3% of the total number of nodes. Therefore, we propose a name disambiguation approach that is suitable for fragmented co-authorship networks of moderate size. The main characteristic of our approach is that it consists of two analysis of author names. The first analysis of author names consider all name pairs as potentially ambiguous. This means that we intentionally avoided to use a graph-based blocking function because highly ambiguous names according to string similarity measures that are placed in different connected components would be undetected. However, a graph-based blocking function itself gives a more confidence that two names are ambiguous when they do not have an extremely high value of some string similarity measure. Therefore, a graph-based blocking function is employed in the second analysis of author names in order to observe ambiguous names missed in the first analysis.

5.4.1.2 Extraction procedure

The extraction of the eLib co-authorship network is done in five phases (see Figure 5.1):

1. First analysis of author names – this phase resulted with the initial identification of authors and the formation of a lookup table which is used to correct author names.

2. Phase 2 – automatic construction of the inverted index which maps authors to papers they published. In this phase the set of papers is also cleaned in a way that the authorship information associated to papers is updated according to performed named corrections.

3. Phase 3 – automatic construction of the co-authorship network from the inverted index and the cleaned set of papers.

4. Second analysis of author names – the name analysis procedure from step one is performed on each connected component independently in order to detect potential name lookup entries that were not detected in the first analysis of author names.

5. Steps 2 and 3 are repeated in order to obtain the final co-authorship network.

The analysis of author names is conducted to identify authors in the exported data. This step is especially important, because the nodes in co-authorship networks are identified by researchers' names, and there might be spelling errors and other inconsistencies in bibliographic records. The analysis of author names is based on the usage of different string similarity measures. The obtained metric values are inspected manually in order to form the name lookup table that is used to correct author names. The detailed explanation of this phase of the network extraction is given in Section 5.4.1.3.

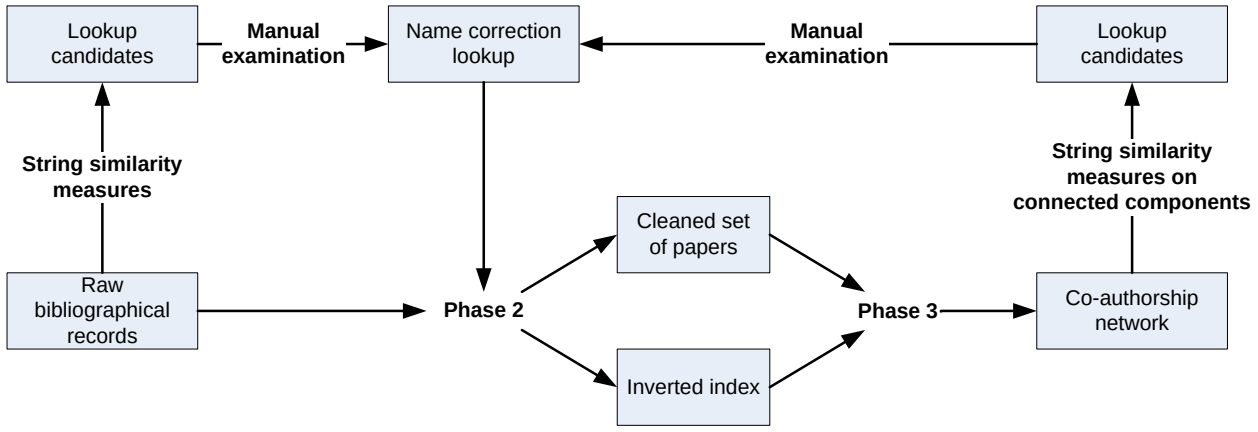


FIGURE 5.1: Data flow in the extraction of the eLib co-authorship network.

In the second stage of the co-authorship network extraction the inverted index is formed. Inverted index, denoted by I, maps an author to the set of papers he or she (co-)authored. In this step all papers are examined (traversed) and for name label A of the currently processed paper P it is checked whether there is a correction in the lookup. If there is a lookup entry $A \to A'$ (name label A has to be corrected to name label A') then updates according to the following rules are performed:

R1 A is removed from the list of authors of P,

R2 A' is added to the list of authors of P,

R3 I is updated so that the following preposition holds: $I(A') = \pi \wedge P \in \pi$.

In the case that the lookup does not contain the correction for name label A then only inverted index is updated to include P in the set of papers associated to A (only rule **R3** above is applied).

The set of keys of the inverted index corresponds to the set of nodes in the co-authorship network. Two authors A and B, where A and B are two keys from the inverted index, are connected by the undirected link $A \leftrightarrow B$ if and only if $I(A) \cap I(B) \neq \varnothing$, where $I(A)$ denotes set of papers published by A. The cardinality of $I(A) \cap I(B)$ is the weight of link $A \leftrightarrow B$. However, from the description of the Phase 2 it can be observed that at the same time papers are also updated according to the performed name corrections (rules **R1** and **R2**). This means that the network can be constructed more efficiently by iterating through papers instead of iterating through all pairs of authors. The complexity of the network construction that is performed using only inverted index is $O(n^2)$ where n is the number of authors. On the other hand, the complexity of the algorithm that iterates through "corrected" papers is $O(p \cdot k^2)$ where p is the number of papers and k is the average number of authors per paper. Since $p < n$ (the number of papers is smaller than the number of authors since there are multi-authored papers) and $k \ll n$ it can be concluded that $p \cdot k^2 \ll n^2$. However, the inverted index is still an important structure for the extraction of the network since it contains information needed to initialize attributes of nodes (productivity and longevity of an author).

The analysis of author names is repeated after the first construction of the co-authorship network, but this time the name pairs are formed from connected components of the network. This means that string distances are computed for researchers contained in the same connected component. This step was motivated by the following observation: if two similar names represent the same author then there is a high probability that they have at least one co-author in common, especially in the case when we are looking for matches between short and full names.

5.4.1.3 Analysis of author names

The analysis of author names in the construction of the eLib co-authorship graph is based on various string similarity measures. String similarity measures quantify either similarity (proximity) or dissimilarity (distance) between two input strings. In this thesis we use three normalized string proximity functions whose implementation is provided by the LingPipe[5] Java library: Jaccard, Jaro-Winkler and TF-IDF. Jaccard and TF-IDF distances are computed both at the token and n-gram ($n = 2$) level. Used string metrics are functions that map a pair of strings to a real number r in the interval [0..1] where

- $r = 1$ indicates two identical strings, and
- a higher value of r indicates a higher similarity between compared strings.

The Jaccard string metric belongs to a class of set based string similarity functions. The main idea of this metric is that it quantifies the degree of the overlap between two token/n-gram sets representing two strings that are being compared. Let p and q be two arbitrary strings. Let T_s denote the set of tokens contained in string s, i.e. the set of substrings of s that are separated by delimiters (one or more white space characters). Let also $N_{s,n}$ denote the set of n-grams contained in s. A N-gram of s is a contiguous sequence of characters in s whose length is equal to n. Then the Jaccard string proximity metric at the token level is defined as

$$\text{Jaccard}(p, q) = \frac{|T_p \cap T_q|}{|T_p \cup T_q|}.$$

Similarly, at the n-gram level we have

$$\text{Jaccard}(p, q, n) = \frac{|N_{p,n} \cap N_{q,n}|}{|N_{p,n} \cup N_{q,n}|},$$

where n is some fixed value (2 for bigrams, 3 for trigrams, etc.).

The Jaro-Winkler string metric is an edit based string similarity function. It was originally defined by Matthew Jaro in 1989 for the purpose of matching the individuals in the 1985 Census of Tampa (Florida) to the individuals in the later independent postenumeration survey [Jaro, 1989]. The Jaro metric was refined by William Winkler [Winkler, 2006] to include the length of the common prefix at the start of compared strings according to his observation that typographical errors are much more likely to occur toward the end of a string. The main components of the Jaro metric are the notion of *common* or *matching* character and the notion of the transposition of matching characters.

Let $p = p_1 p_2 ... p_P$ be a string of length P where p_i denotes the i-th character in string p. For p_i we say that is common with another string q of length Q if and only if

$$(\exists j)\ p_i = q_j \wedge i - D \leq j \leq i + D,$$

where

$$D = \frac{\min(P, Q)}{2}.$$

In other words, the same characters from two strings are considered as matching characters if the distance of their positions is less than half of the length of the shorter string. Let M be the number

[5] http://alias-i.com/lingpipe/

of matching characters according to the previous definition. From input strings p and q we delete all non-matching characters so both of them contains the same characters and both of them are of the same length. Let $p' = p'_1 p'_2 ... p'_M$ and $q' = q'_1 q'_2 ... q'_M$ denote the input strings after the transformation. Two identical characters p'_i and q'_j are considered as *transposed* if $i \neq j$. Let D denote the half of the number of transpositions. Then, the Jaro metric of two given strings p and q is defined as

$$\text{Jaro}(p, q) = \frac{1}{3}\left(\frac{M}{P} + \frac{M}{Q} + \frac{M - T}{M}\right),$$

if $M \neq 0$. When $M = 0$ then $\text{Jaro}(p, q) = 0$. From the definition of the Jaro metric it can be seen that this metric represents the average value of

- the portion of the first string that is matched to the second,
- the portion of the second string that is matched to the first,
- the portion of untransposed common characters.

The Winkler's modification of the Jaro metric incorporates the length of the common prefix for two given strings if the value of the Jaro metric is above some specified threshold T. Winkler originally used $T = 0.7$ which is also used in the implementation provided by LingPipe. Let J be the value of the Jaro metric for two arbitrary strings p and q. Then the Jaro-Winkler similarity is defined as

$$\text{Jaro-Winkler}(p, q) = \begin{cases} J, J < T \\ J + pL(1 - J), J \geq T \end{cases}$$

where L is the length of the common prefix up to a maximum of 4 characters and p represents a constant scaling factor less than 0.25 (usually p is set to 0.1).

The TF-IDF string metric is a set based string similarity measure. This metric is also supervised string similarity function since it has to be trained on a text corpus. To train TF-IDF we used the set of unique author name labels appearing in the eLib records. Similarly as the Jaccard metric, TF-IDF can be computed at both token and n-gram level. Let S be the set of terms, where terms are either tokens or n-grams, representing string s. The main idea of the TF-IDF string metric is that two strings can be considered as close to each other if they contain common distinguishing terms. To define the TF-IDF metric we first need to introduce *term frequency* and *inverse document frequency*. Term frequency, denoted by $\text{TF}(t, S)$, measures how frequently term t appears in S. Raw frequencies can be used, but typically they are either normalized (probability that t appears in S) or logarithmically scaled. On the other hand, inverse document frequency, denoted by $\text{IDF}(t, C)$, measures how much important or informative t is in the training corpus C. The importance of t is determined by its inverse frequency in C. Mathematically speaking,

$$\text{IDF}(t, C) = log\frac{N}{|\{d \in C : t \in d\}|},$$

where N is the number of documents in the training set (in our case one document is one name label), while $|\{d \in C : t \in d\}|$ stands for the number of names in the training set that contains t. Then, for term t its TF-IDF score is simply the product of $\text{TF}(t, S)$ and $\text{IDF}(t, C)$. Therefore, the TF-IDF score of a term contained in a string increases with

1. the number of appearances of the term in the string,
2. the rarity of the term across the training set.

Now, each string can be viewed as a bag of words – vector in D dimensional space, where D is the number of terms in the training set and one component of the vector corresponds to one term from the training set. If term t appears in string S then the value of the corresponding component is equal to the TF-IDF score of t, otherwise it is set to zero. Now, the TF-IDF similarity between two strings s and t is the cosine similarity between their TF-IDF vectors S and T. More formally,

$$\begin{aligned}\text{TF-IDF}(s,t) &= \frac{S \cdot T}{\|S\|\|T\|} = \frac{\sum_{i=1}^{D} S_i \times T_i}{\sqrt{\sum_{i=1}^{D} S_i^2} \times \sqrt{\sum_{i=1}^{D} T_i^2}} \\ &= \frac{\sum_{t \in S \cap T} \text{TF-IDF}(t,S) \times \text{TF-IDF}(t,T)}{\sqrt{\sum_{t \in S} \text{TF-IDF}(t,S)^2} \times \sqrt{\sum_{t \in T} \text{TF-IDF}(t,T)^2}}\end{aligned}$$

Previously described string metrics are calculated independently for:

1. name pairs from the set of unique full name labels appearing in the eLib records (full-full name pairs),
2. name pairs from the set of unique short name labels (short-short name pairs),
3. the Cartesian product of unique full and short name labels (full-short name pairs), where a full name label can be uniquely reduced to an appropriate short name label, i.e. there are no two full name labels in the data that can be reduced (shortened) to the same short name label.

If at least one of the used string metric indicated similarity above 0.6 then the pair of name labels is exported for the manual evaluation. We have not used any automatic strategy to select entries for the lookup, simply because there are different Serbian or Yugoslavian names appearing in our data that have very high degree of similarity. Some representative examples are given in Table 5.1. It can be observed that for all name pairs presented in Table 5.1 we have an extremely high values of the Jaro-Winkler and n-gram measures. On the other hand the values of token distances are below the threshold of 0.6. However, token based metrics can not be used to automatically discard potential candidates for the name correction lookup because the avoidance of Serbian diacritics in name labels can cause extremely low values of token based metrics. For example, in Table 5.2 are shown two name pairs where the first pair of names represent different persons and the other pair the same persons. In the second name pair we have the situation that for one name diacritic marks are not present. For that pair of names the values of token based measures are equal to zero. Moreover, the values of all string similarity measures for the pair of non-identical persons are higher than the values of string similarity measures for the pair of identical persons.

The formed name correction lookup contains 690 entries (name pairs) where 74 entries are detected and added to the lookup in the second name analysis. 206 (29.85%) lookup entries represent full-full name pairs, 369 (53.47%) full-short name pairs and 115 (16.66%) short-short name pairs. The part of the lookup containing some typical examples is shown in Table 5.3. As it can be observed the common errors and inconsistencies in author names are:

TABLE 5.1: Examples of name pairs representing different persons that have high degree of similarity. JT, JN, JW, TIT and TIN denote the value of Jaccard token, Jaccard n-gram, Jaro-Winkler, TF-IDF token, TF-IDF n-gram proximity, respectively.

Name pair	JT	JN	JW	TIT	TIN
Dušan Jovanović - Dušan Jokanović	0.33	0.79	0.94	0.44	0.86
Aleksandar Perović - Aleksandar Pejović	0.33	0.79	0.98	0.44	0.86
Dragana Ranković - Dragica Ranković	0.33	0.71	0.92	0.51	0.77
Dragoljub Jović - Dragoljub Jovanović	0.33	0.82	0.96	0.5	0.94
Aleksandar Ivić - Aleksandar Ilić	0.33	0.75	0.97	0.43	0.85

TABLE 5.2: Proximities of two name pairs where the first pair represent different persons and the second one the same persons.

Name pair	JT	JN	JW	TIT	TIN
Dušan Jovanović - Dušan Jokanović	0.33	0.79	0.94	0.44	0.86
Dušan Jovanović - Dusan Jovanovic	0	0.6	0.93	0	0.74

- inversion of an author's first and last name (the first example in Table 5.3).
- spelling errors (the second example),
- anglicisation of personal names (the third example),
- addition of middle names either in full or compact form (the fourth example),
- addition of marital surname for female authors (the fifth example),
- shortening of a name where the first name is shortened to the first letter (the sixth example),
- addition of separators (the seventh example),
- addition of titles, usually the PhD title ("dr") is added to the name of an author (the eighth example).

Also it can be observed that one person can have multiple, different names (the last three examples in Table 5.3). In such cases all name variants are corrected to the name label that is the longest and thus the most discriminative. Table 5.4 lists examples of name pairs that are detected in the second name analysis indicating also the author in common or the path between the same researchers represented by two different name labels in the network. All name pairs identified in the second name analysis are full-short name pairs which means that 20.05% of the total number of short to full name corrections are identified in this phase.

The basic statistical properties of the name correction lookup are provided in Table 5.5. Namely, Table 5.5 shows the mean, standard deviation, coefficient of variation, minimal and maximal value of proximities for all lookup entries per similarity measure. As it can be observed the majority of lookup entries possess a high Jaro-Winkler score (mean above 0.9, small standard deviation and coefficient of variation). The mean score of token based distances is below the threshold of 0.6 indicating that they are able to identify only a small portion of name synonyms. However, the main insight which Table 5.5 provides is the need to consider more than one string proximity measure when collecting candidates for the name correction lookup. As it can be seen for each of used string similarity metrics there is a lookup entry that has an extremely small proximity score (Min column in Table 5.5), much below the threshold of 0.6. Lookup entries having the smallest value of the proximity per similarity measures are shown in Table 5.6.

TABLE 5.3: Excerpt from the name correction lookup.

Name	Corrected to	JT	JN	JW	TIT	TIN
Nikola Hajdin	Hajdin Nikola	1	0.71	0.46	1	0.8
Todorqević Stevo	Todorčević Stevo	0.33	0.75	0.98	0.44	0.77
Petronievics Branislav	Petronijević Branislav	0.33	0.67	0.93	0.38	0.74
Nisheva-Pavlova Maria	Nisheva-Pavlova Maria M.	0.67	0.95	0.98	0.95	0.98
Rajter-Ćirić Danijela	Rajter Danijela	0.5	0.62	0.92	0.8	0.71
Milogradov-Turin J.	Milogradov-Turin Jelena	0.5	0.74	0.95	0.78	0.87
Lin C.-S.	Lin C.S.	0.8	0.67	0.98	0.91	0.69
Kočinac dr Ljubiša	Kočinac Ljubiša	0.67	0.82	0.97	1	0.89
Van Gulck S.	Van Gulck Stefan	0.4	0.67	0.92	0.73	0.83
Gulck Stefan Van	Van Gulck Stefan	1	0.87	0.72	1	0.93
Gulck S. Van	Van Gulck Stefan	0.4	0.47	0.69	0.73	0.69

TABLE 5.4: Examples of corrections identified in the second name analysis. AIC/Path denotes the author in common or path connecting nodes represented by names, while MS is the maximal similarity which is obtained by string similarity metric M.

Name	Corrected to	AIC/Path	MS	M
Petković T.	Petković Tatjana	Bogdanović Stojan	0.91	JW
Mijatović M.	Mijatović Milorad	Pilipović Stevan	0.91	JW
Kocić V.	Kocić Vlajko	Kečkić Jovan	0.89	JW
Lepović M.	Lepović Mirko	Gutman Ivan ↔ Cvetković Dragoš	0.91	JW
Perišić D.	Perišić Dušanka	Pilipović Stevan ↔ Lozanov-Crvenković Z.	0.9	JW

TABLE 5.5: Statistical properties of the name correction lookup. SD - standard deviation, CV - coefficient of variation, Min - minimal value, Max - Maximal value.

Measure	Mean	SD	CV	Min	Max
JT	0.4	0.26	0.66	0	1
JN	0.62	0.17	0.27	0.08	0.94
JW	0.91	0.08	0.09	0.26	0.99
TIT	0.58	0.26	0.45	0	1
TIN	0.76	0.13	0.17	0.12	0.98

TABLE 5.6: Lookup entries with the smallest proximity score per string similarity measure (indicated by the bold typeface).

Name	Corrected to	JT	JN	JW	TIT	TIN
Bilimovitch A.	Bilimović Anton	**0**	0.42	0.88	**0**	0.52
Basilewitsch W.	Baziljević V.	0.2	**0.08**	0.77	0.05	0.12
Karabin M.	Vukićević-Karabin M.	0.6	0.5	**0.26**	0.69	0.67

5.4.2 Analysis of the eLib co-authorship network

The publication dynamics of the eLib journals is investigated by measuring the number of papers at a yearly level. The construction of the eLib co-authorship network enables us also to examine other context-relevant static and dynamic aspects of the eLib community: the number of authors (where we distinguish between male and female authors), the fraction of "returning" authors, the average number of authors per paper, the fraction of single-authored papers, the distribution of the number of papers per author, and the distribution of the number of authors per paper.

Definition 5.1 (Returning, old author; new author). An author is called returning or old if he or she already published paper in one of the eLib journals. An author that publishes a paper for the first time in an eLib journal is called a new author.

Definition 5.2 (**Author timespan).** The timespan for author A is defined as the number of years that passed from the publication of A's first article to the publication of A's last article in eLib journals. If A published exactly one paper in the eLib journals then A has timespan equal to one.

The analysis of structure of scientific collaborations in the eLib journals is based on standard methods and metrics used in analysis of social networks. Connected component analysis is conducted to determine properties of connected components contained in the network.

We distinguish between two types of components in the eLib co-authorship network: non-trivial and trivial components.

Definition 5.3 (**Trivial component).** A component of a co-authorship network is considered trivial if it is a complete sub-graph of the network (each two nodes in the component are directly connected), and the weight of each link is equal to one.

Trivial components represent collaborations established by publication of exactly one paper. If there is a group of authors that published exactly one paper together, and if this paper is the only published paper for each member of the group, then the members of the group form a trivial component.

We used the following domain-independent metrics to quantify nodes (authors) in the eLib network: degree centrality, betweenness centrality, small-world coefficient, and clustering coefficient. To quantify collaboration strength we use the normal scheme where two researchers are connected by a link of weight w if they authored w joint papers (with or without other co-authors).

Definition 5.4 (**Isolated author).** An author is called isolated if his degree centrality is equal to zero. Isolated nodes in the eLib co-authorship network represent authors who have not collaborated with other authors from the eLib community by publishing joint papers in the eLib journals.

To evaluate the productivity of researchers Lindsey [1980] suggests the following methods: normal count, fractional (adjusted) count, and straight count. Normal count gives every author one credit, straight count assigns all the credit to the first author only, while fractional count assigns credit equal to $1/n$ to each of the n co-authors. In this study, we use the normal counting method for measuring author's productivity. Additionally, timespan (see Definition 5.2) is recorded for each author, since it is a metric of long-term presence in the eLib journals. Correlations between author metrics are investigated by the computation of Spearman's rank correlation coefficient.

Small-world and clustering coefficients for components are calculated by averaging the values of mentioned metrics for all authors from the components. The number of publications for a component is the total number of publications written by authors from the component.

Definition 5.5 (**Component timespan).** The timespan of a component is the number of years that passed from the creation of the component to the last event which changed the structure of the component.

In order to identify cohesive subgroups in the eLib co-authorship networks we used a method based on the modularity measure introduced by Girvan and Newman [2002]. In the case of weighted networks, the modularity measure, denoted by Q, is defined as

$$Q = \sum_{c=1}^{n_c} \left[\frac{W_c}{W} - \left(\frac{S_c}{2W} \right)^2 \right],$$

where n_c is the number of communities in the partition, W_c is the sum of weights of intra-community links of community c, S_c is the total weight of links incident to nodes in c, and W is the total weight of links in the network. Simply stated, modularity accumulates the difference between the total weight of links within a cluster and the expected total weight in an equivalent network with links placed at random. Although widely used, the modularity measure has a weakness known as the resolution limit problem – community detection techniques based on modularity maximization may fail to to identify modules smaller than a scale which depends on the total size of the network. Therefore, the application of modularity maximization methods requires investigation of the quality of obtained community partitions. In order to assess the reliability of the community detection method we use the definition of community proposed by Radicchi et al. [2004] adopted for weighted networks. A community is called *Radicchi strong* if for each node in the community the sum of weights of links within the community (strength of intra-community links) is higher than the sum of weights of links connecting the node with the rest of graph (strength of inter-community links). Initially, we investigated the performance of five different community detection techniques on the largest connected component by observing the value of Q and the number or Radicchi strong comunities. The results showed that the Louvain method [Blondel et al., 2008] is the most suitable for our case study. The method uses a greedy multi-resolution approach to maximize Q starting from the partition where all nodes are put in different communities. When Q is optimized locally the algorithm builds the coarse-grained description of the network (network of communities), and then repeats the same procedure until a maximum of modularity is attained.

Each component is created by publishing a paper that is written by one or more new authors. Therefore, the year when the paper was published determines the time of creation of the component. At the time of creation each component is either an isolated node or trivial component. There are two types of evolutionary events that change the structure of the component: inclusion of a new author into the component and renewal of collaborations among authors from the component. Inclusion of new authors is achieved by the establishment of a collaboration between an old author from the component and an author that published a paper in the eLib journals for the first time. This event causes the creation of one new node and one new link in the component. Renewal of collaborations is manifested by an increase of link weights.

To investigate the evolution of the eLib co-authorship network we construct time-ordered snapshots of the network, i.e. the sequence of networks

$$S = \langle N_y \rangle, 1932 \leq y \leq 2011,$$

where N_y denotes the snapshot of the network in the year y. The set of nodes of N_y contains all authors that published at least one article before or during y. A link between authors A and B is present in N_y if A and B established collaboration before or during y. Accordingly, the weight of link $A \leftrightarrow B$ is the number of papers A and B authored together before or during y. Since co-authorship networks evolve by adding nodes and edges, the sequence S satisfies the following property: network N_y is a sub-graph of network N_w if $y < w$. The evolution of a quantifiable property P of the eLib co-authorship network is investigated by the examination of the numerical sequence $P(S) = \langle P(N_y) \rangle$.

The main property of trivial components is that they do not evolve. Evolutionary properties of non-trivial components are investigated by constructing sequence S from the full eLib co-authorship network (network N_{2011}) after removing isolated nodes and trivial connected components. This means that isolated nodes and trivial connected components exist in a network within S only if they lose

mentioned properties in one of subsequent network snapshots.

In the evolutionary analysis we also distinguish three types of collaborations: collaborations between old (returning) authors, old and new authors, and new authors. In the computation of the number of collaborations link weights have to be taken into account. Let A and B be two authors connected by link $A \leftrightarrow B$ of weight w in network snapshot N_y. The following cases are possible:

1. Authors A and B do not exist in network N_{y-1}. Then link $A \leftrightarrow B$ in year y denotes one collaboration between A and B as new authors and $w-1$ collaborations between A and B as old authors.

2. A exists and B does not exist in N_{y-1}. Then link $A \leftrightarrow B$ in year y denotes one collaboration between old author A and new author B, and $w-1$ collaborations between A and B as old authors.

3. Both A and B exist in N_{y-1} and they are connected by a link of weight z. In this case link $A \leftrightarrow B$ in year y denotes $w-z$ collaborations between A and B as old authors.

5.4.2.1 Publication dynamics

In total 6480 research papers were published in the eLib journals from 1932 to 2011. Figure 5.2 shows the publication dynamics of eLib, i.e. the number of papers published per year. It can be noticed that there are several periods in the evolution of eLib where the number of papers per year exhibits an increasing trend, as well as several periods when it shows a decreasing trend. Also, it can be observed that there was no scientific production in the eLib journals during and immediately after the Second World War (1939–1946). In the first five years of eLib, the number of published articles had a relatively stable evolution which means that there was no drastic increase or decrease in the number of published papers. The first long-term growth trend in the number of publications appeared during the Informbiro period (1947–1953) which is characterized by the conflict between Yugoslavia and the Soviet Union. In that time two new Yugoslav mathematical journals were founded, and the number of publications increased from 15 in 1947 to 81 in 1953. After the Informbiro period the scientific production exhibited a general decreasing trend which ended by 1963. After 1963 the number of published papers started to increase and this trend ended in 1979. In the mentioned year the highest number of published papers before Yugoslav breakup is recorded (164 published papers).

After the death of Yugoslav president Josip Broz Tito in 1980, the economic crisis and national tensions in Yugoslavia started to emerge, leading to the Yugoslav breakup in 1991 and ethnic wars in the period from 1991 to 1995. These events evidently affected scientific production of the eLib community: Figure 5.2 shows that the largest continual decrease in the number of papers per year occured in the period from 1980 to 1996. Especially in the war period (1991–1995) an extremely low degree of scientific production can be observed. In the study of mathematics scientific production from our neighboring country, Dravec Braun [2012] correlates the impact of the Serbo-Croatian war and establishment of universities and institutes with the Croatian mathematics scene, highlighting the two periods of stability and development before and after 1993. Although the war activities were not present on the territory of Serbia, the country was faced with the international sanctions which caused hyperinflation and brain drain. In that period the government's funds for education and science were drastically reduced, and researchers, mostly worried how to survive in such hard-living conditions, were demotivated for scientific work and publishing.

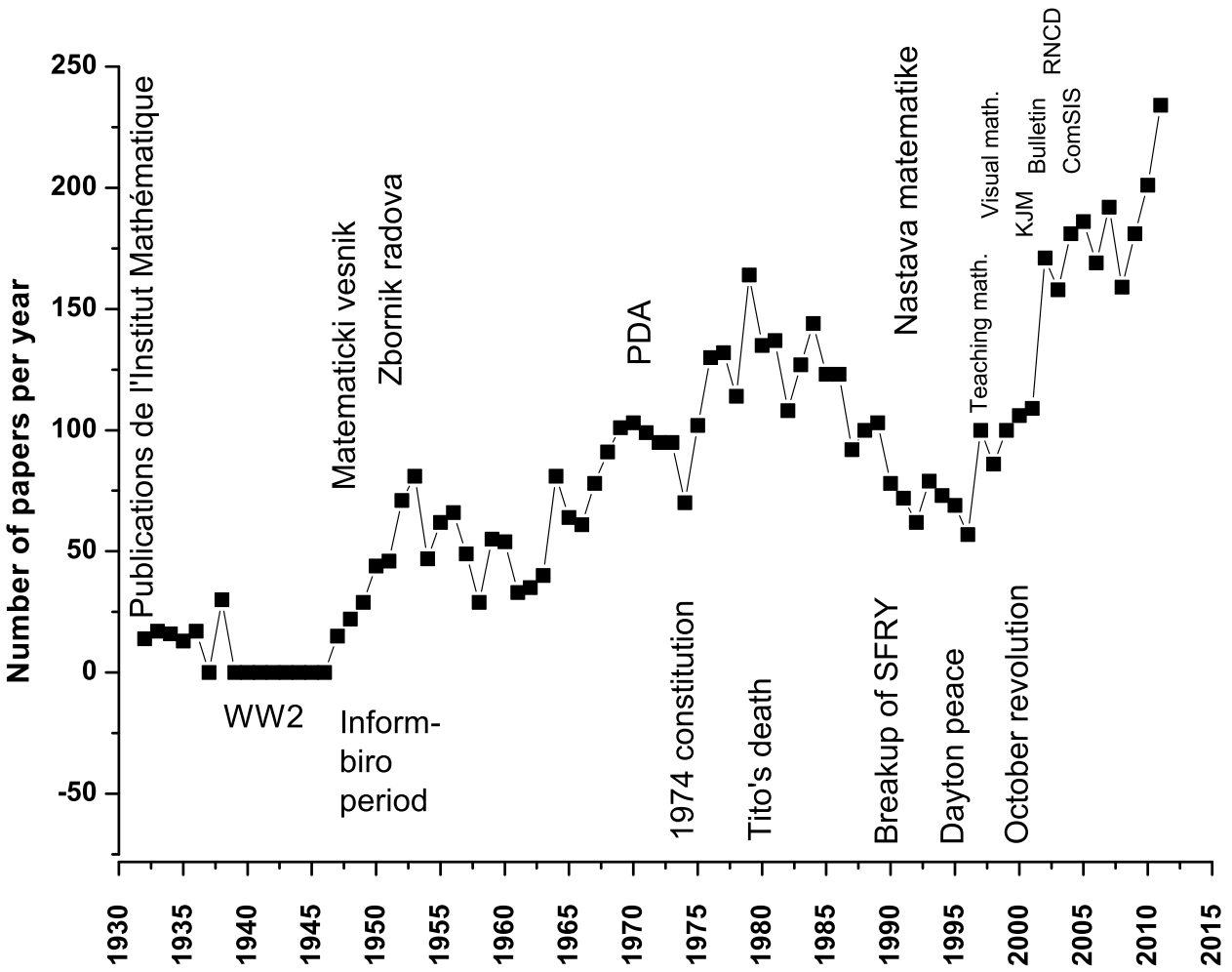


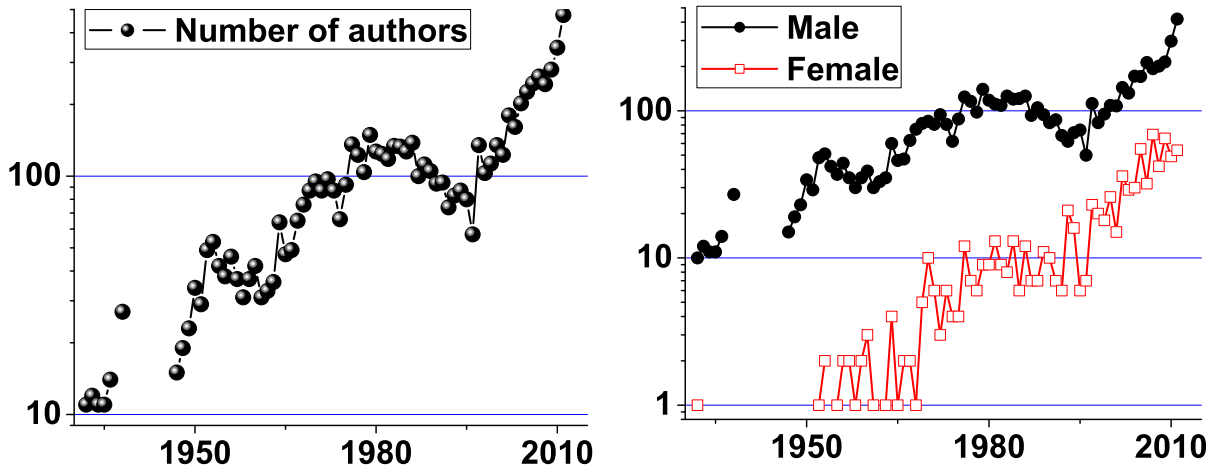
FIGURE 5.2: The number of papers published in the eLib journals per year. Above the line are shown the names (or abbreviations) of journals in the time they were founded, while important events in Yugoslav/Serbian history are positioned below the line.

After the Dayton peace agreement (end of 1995), relative stabilization of the political situation in former Yugoslav republics caused a growth in the number of publications. However, the largest discontinuity in the number of published papers per year occured after the so-called "October revolution" (downfall of Slobodan Milošević government by the end of 2000), when the country entered a transitional period towards free-market economy and started to open to the rest of the world. Due to the new rules set by the Serbian ministry of science which emphasized the number of publications in journals as the main criterion for evaluation of scientific work, after 2001 the number of articles in the eLib journals per year is significantly higher than in previous years. Additionally, in the last years four new journals indexed by eLib are founded.

5.4.2.2 Author dynamics

The total number of authors that published papers in the eLib journals during the examined period is 3597, where 3147 (87.49%) authors are male and 450 (12.51%) authors are female. Figure 5.3(a) shows the number of authors per year. It can be observed that the evolution of the number of authors per year has similar shape to the evolution of the number of papers per year. Pearson's correlation coefficient between these two variables is 0.929, while Spearman's correlation coefficient is

0.981. Additionally, in each year the number of male authors is significantly higher than the number of female authors (Figure 5.3(b)). The smallest ratio between male and female authors was in 2007 (193 male and 69 female authors), while the largest ratio (excluding years when there were no female authors – period from 1933 to 1951, 1954, and 1962) was in 1968 (75 male and 1 female author).



(A) (B)

FIGURE 5.3: The number of eLib authors (a) and the number of male and female authors (b) per year.

The fraction of returning authors per year is shown in Figure 5.4. It can be seen that two periods considering returning authors can be distinguished. After 1998 the fraction of returning authors is always smaller than 0.5, which means that the majority of authors are new authors. In contrast, before 1998, in the majority of years, the majority of authors were returning authors. The notable exception is the year 1949 when the fraction of returning authors is the lowest during the whole eLib evolution. In that year 74% of authors were new authors.

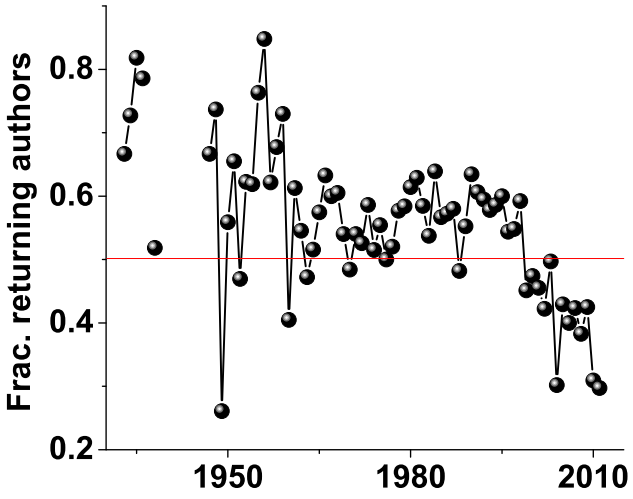
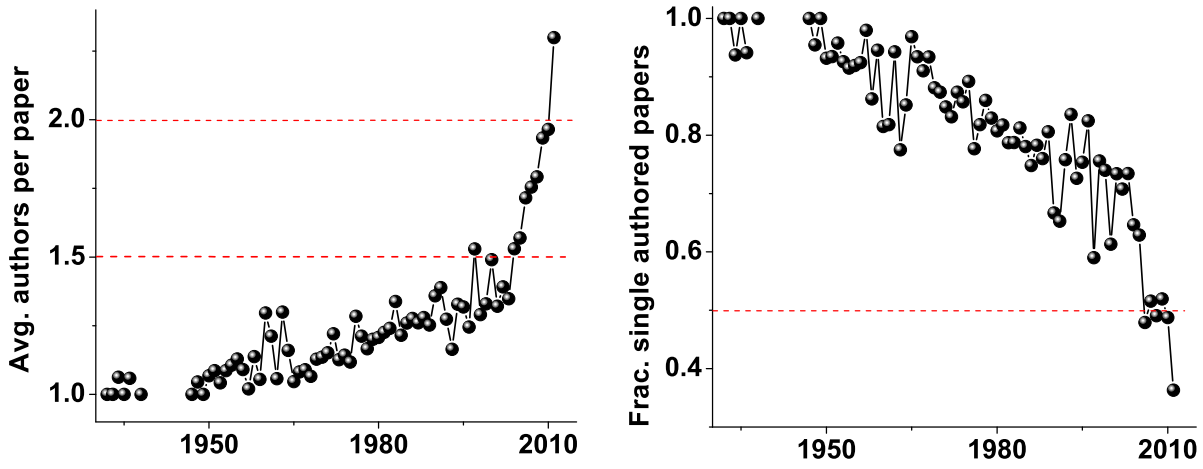


FIGURE 5.4: The fraction of returning authors per year.

5.4.2.3 Basic characteristics of collaboration and productivity of eLib authors

The majority of articles in the eLib journals in the investigated timeframe are single-authored papers: 4836 papers (74.63% of the total number of papers) are written by exactly one author. This situation

is not surprising for mathematical journals, since researchers in mathematics and humanities usually engage in solitary work, while laboratory scientists tend to write articles with many co-authors. Figure 5.5 shows the evolution of the average number of authors per paper and the fraction of single-authored papers. It can be seen that the average number of authors per paper increases, while the fraction of single-authored papers decreases as eLib evolves. A similar evolutionary trend was also observed for articles indexed in "Mathematical Reviews" in the period from 1940 to 2000 [Grossman, 2002a]. As can be seen in Figure 5.5(a), the average number of authors per paper was slowly increasing from 1 to only 1.56 in the period from 1932 to 2005. However, in the last years (2005–2011) the average number of authors per paper has been growing significantly faster than in previous years, reaching 2.29 authors per paper in 2011. One of the factors which caused such fast growth is the foundation of new journals, RNCD and ComSIS, whose scope is not purely mathematical, but oriented to applications of mathematics and computer science, where the number of authors per paper is generally higher compared to pure mathematical research. Naturally, as the average number of authors per paper increases the fraction of single-authored papers decreases. More than half of the papers per year in the period 1932–2005 are single-authored papers, and only in the last years of eLib evolution the majority of papers were written by two or more authors.



(A) (B)

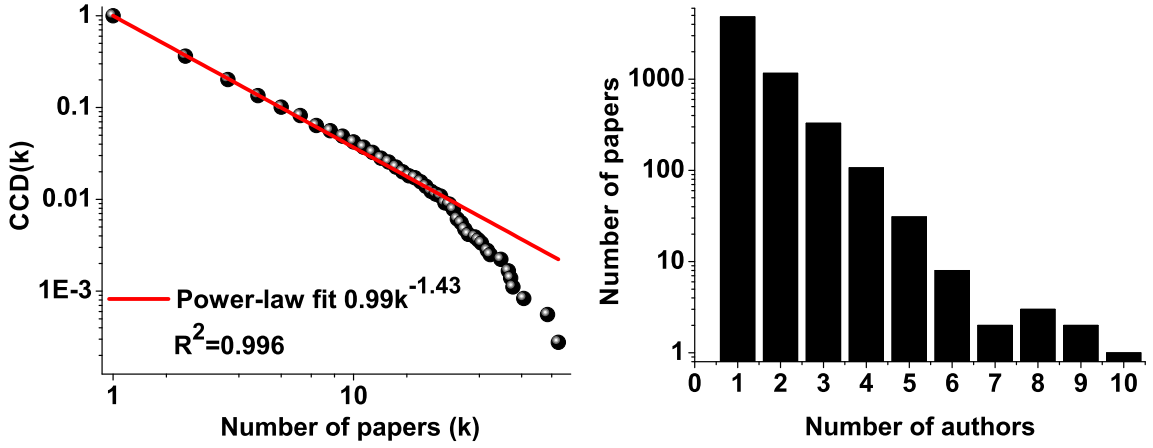
FIGURE 5.5: The evolution of the average number of authors per paper (a), and the fraction of single authored papers (b).

Figure 5.6(a) shows the complementary cumulative distribution of the number of papers (k) per author plotted on log-log scales. It can be seen that the distribution closely follows a power-law with a faster decay (so called "cutoff" or truncated power-law) for $k > 25$. The same phenomena was also observed for the distribution of the number of papers per author in Los Alamos electronic preprint archive [Newman, 2001c]. The power-law nature of the distribution implies that the majority of authors published a small number of papers that is close to the average value. On the other hand, there is a small, but statistically significant, fraction of authors whose production is extremely higher compared to the average eLib author. For example, the most productive author present in the eLib journals is Ivan Gutman who published 71 papers in total, which is drastically higher than the average number of papers per author (the average number of papers per author is 2.458).

The emergence of power-laws in empirical data can be explained by the principle of cumulative advantage (also known as "rich get richer" or preferential attachment principle). When applied to the distribution of the number of papers per author, the principle of cumulative advantage denotes

that the probability that author A will publish a paper in the future is proportional to the number of papers A already published. In other words, the principle states that author A who at some point in time published more papers than author B has higher probability to publish a paper in the future than B. Observed cutoff in the distribution also has a natural, evolutionary explanation. Cutoffs in power-law distributions appear when time or capacity constraints are incorporated into the principle of cumulative advantage [Amaral et al., 2000]. Even the most productive authors after some time stop publishing papers (due to retirement or death) thus introducing time constraints to the principle of cumulative advantage which governs the inequalities in the number of published papers per author.

The inequality in scientific production of a group of authors can be also expressed using Lorenz curve which is a plot of the fraction of papers produced by the most prolific authors against the fraction of authors that produced them. Figure 5.7 shows Lorenz curve for eLib authors. It can be observed that less than 10% of the most productive authors produced more than 50% of the total number of publications published in eLib journals in the examined time frame. Additionally, it can be noticed that the inequality of production among eLib authors is close to the famous Pareto principle (the 80-20 rule): 75% of the total number of publications is authored by 25% of the total number of eLib authors.



(A) (B)

FIGURE 5.6: Complementary cumulative distribution of the number of papers per author (a), and the distribution of the number of authors per paper (b).

In contrast to the heavy-tailed distribution of the number of papers per author, the distribution of the number of authors per paper (Figure 5.6(b)) possesses a characteristic scale. This means that there are no papers with an extremely large number of authors: the maximal number of authors per paper is equal to ten[6], while the average number and standard deviation of authors per paper are equal to 1.36 and 0.756, respectively. The majority of eLib papers are written by exactly one author (74.63%), 17.91% by two authors, 5.07% by three authors, and only 2.37% of the total number of publications have more than three authors.

5.4.2.4 The structure of the eLib co-authorship network

The co-authorship network formed from eLib bibliographic records contains 3597 nodes (authors) and 2766 links (collaborations). Basic quantities describing structural properties of the eLib co-authorship

[6]The article "Serbian Virtual Observatory" published in "Review of the National Center for Digitization" in 2009 is the article with the highest number of authors per paper.

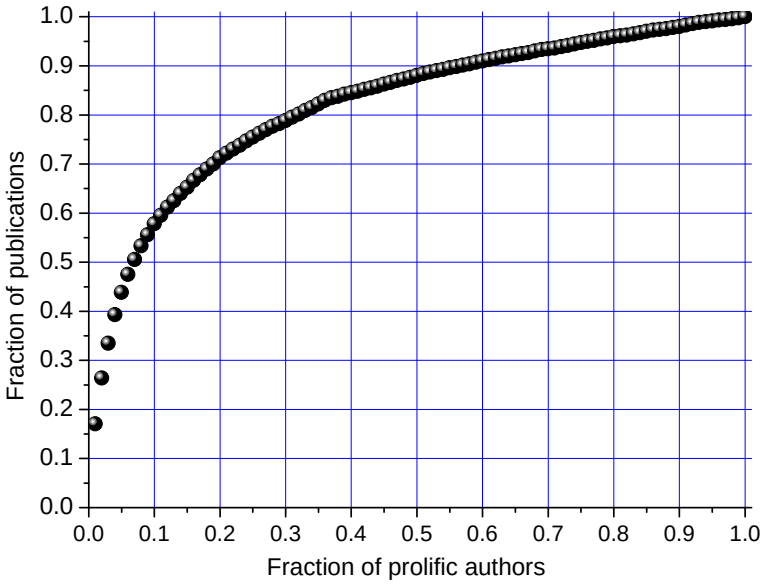


FIGURE 5.7: Fraction of papers written by the most prolific eLib authors.

network are summarized in Table 5.7. In contrast to the co-authorship networks analyzed by Newman [2001c], Grossman [2002b], Barabasi et al. [2002], Nascimento et al. [2003], Liu et al. [2005], Bettencourt et al. [2009], and Perc [2010], where the existence of a giant connected component is observed, the eLib co-authorship network is extremely fragmented: it contains 625 connected components neither of which is a giant connected component. The network exhibits small average shortest path lengths and a drastically larger clustering coefficient than the clustering coefficient of comparable Erdős-Renyi random graph. The clustering coefficient of the random graph with $N = 3579$ nodes and $L = 2766$ links is equal to $2N/L(L-1) = 0.00094$, while the clustering coefficient of the eLib co-authorship network is 0.44. This means that the eLib co-authorship network exhibits the Watts-Strogatz small-world property [Newman, 2001d; Watts and Strogatz, 1998a]. Also, there is a slight tendency that highly connected eLib authors be coauthors among themselves (the assortativity coefficient is equal to 0.115). Almost the same degree of assortativity is previously observed for the co-authorship graph extracted from the bibliographic records of "Mathematical Reviews" where the assortativity coefficient is equal to 0.12 [Newman, 2002].

Table 5.7 also provides basic structural quantities of co-authorship networks that are restricted to individual journals indexed by eLib. It can be seen that journal co-authorship networks mostly have similar structural characteristics as the eLib network: they are sparse, fragmented (a large number of connected components), do not contain a giant connected component, have a significant number of isolated nodes, and exhibit the small-world property. The number of papers published in an eLib journal strongly correlates with the number of authors and the number of components present in the journal: Pearson's correlation coefficient (PCC) between the number of papers and the number of authors is 0.959 (the value of Spearman's correlation coefficient is 0.718), while the PCC between the number of papers and the number of components is 0.909 (Spearman's correlation coefficient is 0.711).

Bibliometric indicators such as impact factor and h-index are commonly used to evaluate and compare scientific impact of journals [Ivanovic et al., 2012]. To the contrary, structural properties of journal co-authorship networks enable us to observe differences between journals that are related to the collaborative behaviour of authors rather than the scientific impact of their work. The smallest fraction of isolated nodes, only 3% of the total number of authors, and the highest clustering coefficient

TABLE 5.7: Basic structural parameters of the eLib co-authorship network and co-authorship networks restricted to individual journals: #P – the number of papers, #N – the number of nodes (authors), #L – the number of links, I – the fraction of isolated authors, MFR – Male-Female Ratio, #C – the number of connected components (isolated nodes are excluded), LC – the relative size of the largest connected component, SW – small-world coefficient, CC – clustering coefficient, AC – assortativity coefficient

Net	#P	#N	#L	I	MFR	#C	LC	SW	CC	AC
ELIB	6480	3597	2766	0.33	6.99	625	0.06	2.117	0.44	0.115
PIM	2150	1147	563	0.42	8.10	202	0.05	1.721	0.25	0.108
PDA	122	48	21	0.47	8.60	7	0.12	1.421	0.30	-0.435
MV	2264	1255	576	0.46	15.51	225	0.01	1.355	0.27	0.542
ComSIS	202	520	766	0.03	9.40	124	0.06	1.235	0.81	0.427
TTM	125	102	66	0.35	5.00	19	0.11	1.426	0.45	0.133
Bulletin	84	84	87	0.15	4.25	13	0.38	1.843	0.59	-0.199
KJM	256	294	226	0.26	4.34	66	0.06	1.266	0.49	0.204
NM	352	195	58	0.63	2.09	24	0.04	1.293	0.26	0.296
ZR	327	172	73	0.54	5.61	24	0.05	1.378	0.46	0.3
RNCD	338	236	249	0.24	1.62	44	0.06	1.304	0.65	0.696
VM	260	212	182	0.36	6.85	43	0.08	1.224	0.39	0.681

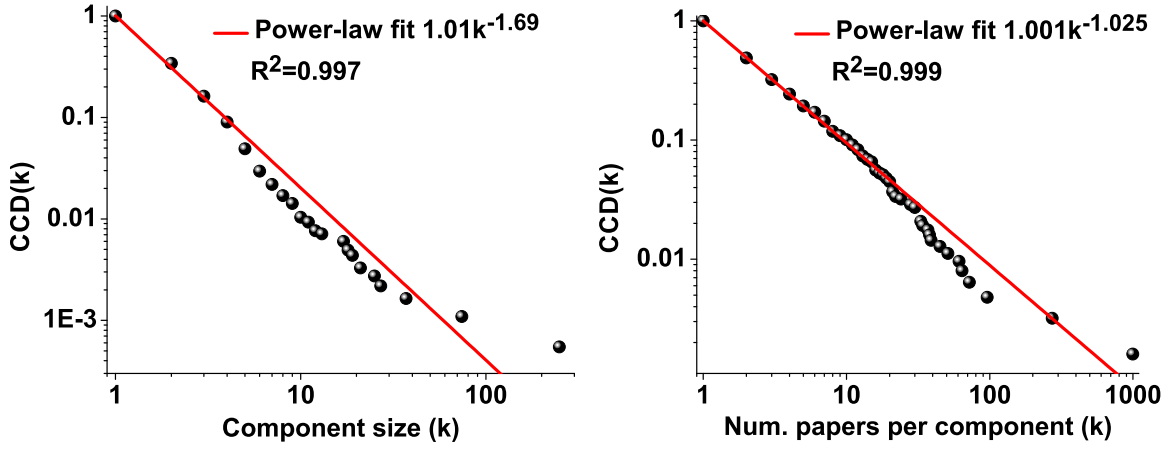
is exhibited by the co-authorship network representing collaborations in the "Computer Science and Information Systems" (ComSIS) journal. This means that this journal mostly publishes papers with two or more authors, and has the most cohesive community of authors compared to other journals. Additionally, this journal has the highest link-node ratio, as well as the ratio between the number of links and the number of non-isolated nodes, which means that ComSIS authors established higher intensity of collaborations compared to other journals. The highest degree of collaborative behaviour exhibited by ComSIS authors can be explained by the fact that ComSIS is the only computer science journal indexed by eLib. Generally speaking, research in computer science, due to its experimental and applicative (industrial) component, mostly requires effort of a group of people, and consequently has higher collaborative potential compared to research in mathematics. The largest fraction of isolated nodes is in the co-authorship network of "Nastava matematike", where more than 60% of the total number of authors are isolated. This means that this journal publishes mostly single-authored papers. The largest male-female ratio is in "Matematički vesnik", every fifteenth author present in this journal is female. In contrast, "Review of the National Center for Digitization" has male-female ratio 1.69 which means that this journal has the smallest gender gap among eLib journals. From the data presented in Table 5.7 it can be also observed that the co-authorship networks of "Review of the National Center for Digitization", "Visual Mathematics", "Matematički Vesnik" and "Computer Science and Information Systems" exhibit strong assortative mixing, i.e. highly connected authors publishing in those journals tend to connected among themselves. To the contrary, the co-authorship network of "Publications of Department of Astronomy" possesses strong disassortativity which means that highly connected authors tend not to collaborate with other highly connected authors.

As already mentioned, the eLib co-authorship network contains a large number of connected components, neither of them being giant. Figure 5.8(a) shows the complementary cumulative distribution of component sizes. It can be seen that the distribution can be very well approximated by the power-law with the scaling exponent $\gamma = 2.69$ (the coefficient of determination is $R^2 = 0.99$). This means that the majority of components are small-size components, but there are also connected components

whose size is much larger than the average component size. The power-law scaling also appears in the distribution of the number of papers written by authors from the same component (Figure 5.8(b), $\gamma = 2.02$, $R^2 = 0.99$). Let A and B denote two connected components, where component A is larger than component B. The principle of cumulative advantage in the case of these two distributions suggests the following:

1. There is a higher probability that the eLib community will be expanded with an author who knows or collaborates with authors from component A than with those that are contained in the smaller component B. The mentioned probability is proportional to component size.

2. It is more probable that a newly published eLib paper, written by at least one returning author, will be written by authors from component A than authors from component B. Again, the probability is proportional to the number of papers written by authors from the connected component.



(A) (B)

FIGURE 5.8: The distribution of the size of components (a), and the number of papers per component (b) in the eLib co-authorship network.

The eLib co-authorship network contains nearly the same number of trivial and non-trivial components: 319 components are trivial (51.04%), while 306 of them are non-trivial. There are three largest trivial components, where each of them contains six authors. On the other hand, there are 126 smallest non-trivial components. Those non-trivial components are groups of two authors which published more than one paper together in the eLib journals, but have not collaborated with any other eLib author. The structural characteristics of ten largest connected components are shown in Table 5.8. The largest connected component encompasses 249 authors, which is only 6% of the total number of authors, or 10.49% of the total number of non-isolated authors. The number of papers published by authors from the largest component is 997, which is 15.38% of the total number of papers. However, authors from the largest connected component published the highest number of papers per component, 3.65 times more than the second highest number of papers per component. Articles written by authors from the largest component are published in 10 (out of 11) journals indexed by eLib, in all journals except "Visual mathematics". Additionally, the largest component is the only component that has a timespan which is the same as the lifetime of the whole network: it contains authors that published papers in 1932, as well as authors that published their first papers in 2011.

TABLE 5.8: The ten largest connected components in the eLib co-authorship network: #N – the number of nodes (authors), #L – the number of links, #P – the number of papers that authors in the component published, #J – the number of journals where authors from the component published their papers, EP – evolution period of the component, $\langle d \rangle$ – average degree of node in component, SW – small world coefficient, D – diameter, and CC – clustering coefficient.

#N	#L	#P	#J	EP	$\langle d \rangle$	SW	D	CC
249	399	997	10	80	3.20	7.48	19	0.52
74	111	273	8	59	3.00	5.17	11	0.44
37	97	20	2	8	5.24	2.76	5	0.84
27	43	51	2	43	3.18	2.59	5	0.60
25	38	39	6	48	3.04	3.49	7	0.55
21	24	61	4	42	2.28	3.09	7	0.21
19	27	34	7	41	2.84	3.11	6	0.57
19	44	10	1	8	4.63	2.37	5	0.84
18	19	72	5	52	2.11	3.23	7	0.13
17	80	6	1	5	9.41	1.41	2	0.86

Each author in the co-authorship network can be characterized by several metrics used in social network analysis. Table 5.9 presents values of author metrics for top ten authors from the largest connected component when ranked by degree centrality. It can be seen that the best connected author is Ivan Gutman, a Serbian Academician from Kragujevac, who is connected to 50 other authors. Ivan Gutman also published the highest number of papers in the eLib journals. The best ranked author by betweenness centrality is Žarko Mijajlović, full professor at the Faculty of Mathematics, University of Belgrade, who is also the second best connected author. Mijajlović is the most central author in the largest connected component and can be viewed as the strongest middleman connecting different groups of authors. Top ten best ranked authors by degree centrality also have high betweenness centrality, i.e. all of them are positioned in the top 20 best ranked authors by betweenness centrality. Paul Erdős is the best ranked non-Serbian mathematician by betweenness centrality (his rank is 11). Petar M. Vasić has the highest value of clustering coefficient, which means that his co-authors established the tightest degree of collaboration between themselves compared to co-authors of other top ten highest degree authors. The highest value of author timespan (see Definition 5.2) has Stanković Bogoljub, a Serbian Academician from Novi Sad. He does not belong to the largest, but the second largest connected component. The first paper of Bogoljub Stanković published in the eLib journals is from 1953, while the last one is from 2011.

For each author in the eLib community we computed the Spearman correlation coefficient between the co-authorship based metrics and metrics of productivity. The number of published papers and author timespan are representatives of metrics of productivity, while degree centrality, betweenness centrality and clustering coefficient are co-authorship network based metrics. Results are summarized in Table 5.10. All computed correlations are significant at 0.05 level. It can be seen that there are strong correlations between the number of published papers and timespan, and between degree centrality and clustering coefficient. The correlation between the number of papers and timespan is expected to be strong: if an author has a large number of publications it is more likely that they are published in wider time range than in a smaller one. Strong correlation between degree centrality and clustering coefficient implies that co-authors of highly connected authors established a higher number of collaborations between themselves than co-authors of slightly connected authors. However, the

TABLE 5.9: The top ten highest degree authors in the largest component of the eLib co-authorship network: Deg. – degreee, #P – the number of published papers, #PR – rank of author according to the number of published papers, S – author timespan, SR – rank according to timespan, B – betweenness, BR – rank according to betweenness centrality, and CC – clustering coefficient.

Name	Deg.	#P	#PR	S	SR	B	BR	CC
Ivan Gutman	50	71	1	40	10	12263	7	0.0383
Žarko Mijajlović	16	44	4	40	10	17983	1	0.1000
Dragoš Cvetković	14	41	5	42	9	12255	8	0.0769
Zoran Ognjanović	14	23	14	16	31	4089	15	0.1098
Boško Jovanović	12	36	6	37	12	6581	10	0.0758
Slobodan Simić	12	21	15	36	13	3042	17	0.1364
Miomir Stanković	11	12	23	34	15	3218	16	0.1091
Petar M. Vasić	10	23	14	21	27	15691	4	0.2000
Slaviša Prešić	8	26	11	47	5	16722	2	0.1071
Jovan D. Kečkić	8	46	3	34	15	5183	13	0.1428

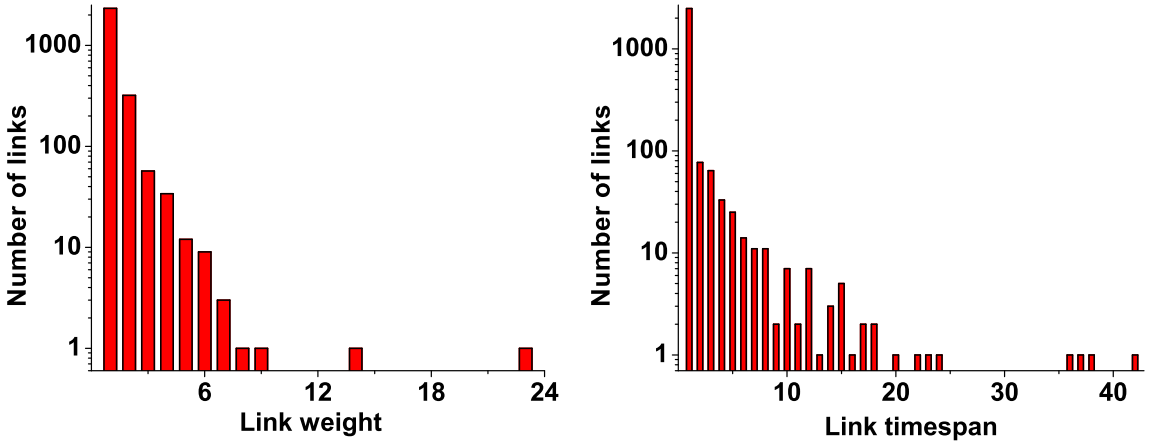
most important are correlations between different types of author metrics, i.e. network-based metrics of importance (centrality metrics) and metrics of productivity (the number of published papers and timespan). It can be seen that betweenness centrality has stronger correlations with the number of published papers and timespan than degree centrality: correlations between betweenness and metrics of importance are moderate, while correlations between degree centrality and metrics of importance are weak. This means that betweenness centrality is a better indicator of author productivity and long term presence in the eLib journals than degree centrality.

TABLE 5.10: Values of Spearman's correlation coefficient for author metrics.

	Num. papers	Timespan	Degree	Betweenness
Timespan	0.91			
Degree	0.12	0.11		
Betweenness	0.51	0.49	0.47	
Clustering coef.	-0.06	-0.07	0.79	0.11

Links in the eLib co-authorship network have weights which denote the number of papers two authors jointly published. Therefore, link weight can be viewed as a measure of strength of collaboration between two authors. Figure 5.9(a) shows the distribution of link weights for the eLib co-authorship network plotted on semi-log scales. It can be seen that the distribution is monotonically decreasing. The majority of all links (64.69%) have weight that is equal to one which implies that eLib authors mostly publish only one joint paper together in the eLib journals. The highest collaboration strength have authors Izidor Hafner and Tomislav Žitko. They published 23 joint papers in "Visual mathematics" in the period from 2002 to 2007. Another aspect related to scientific collaborations is timespan: the time passed from the first to the last publication of two authors. Figure 5.9(a) shows the distribution of link timespan for the eLib co-authorship network plotted on semi-log scales. The largest link timespan is exhibited by authors Ranko Bojanić and Miloš Tomić. Those two authors published seven eLib papers together, the first in 1954, and the last in 1995.[7]

[7]The last joint eLib paper of Ranko Bojanić and Miloš Tomić is dedicated to the memory of Slobodan Aljančić with whom they co-authored their first eLib paper.



(A) (B)

FIGURE 5.9: The distribution of link weights (a) and link timespans (b) in the eLib co-authorship network.

The importance of collaboration can be measured by link betweenness. Links with a high value of betweenness are separating different communities of nodes in a network, and this observation is used in the construction of the Girvan-Newman community detection algorithm [Girvan and Newman, 2002]. The highest value of betweenness in the eLib co-authorship network has the collaboration between Slaviša Prešić and Zoran Ivković. They published exactly one joint paper in 1967,[8] which can be considered as the most important paper for the overall connectedness of the largest connected component of the eLib co-authorship network.

For each link (collaboration) in the eLib community we computed the Spearman correlation coefficient between the following metrics: link weight, link timespan, number of authors in common for two authors that are connected by the link and link betweenness. Results are summarized in Table 5.11. All computed correlations, except the correlation between timespan and authors in common, are significant at 0.05 level. However, the only strong Spearman correlation is between link weight and link timespan. This means that authors who collaborated in a longer time interval tend to have more papers in common compared to co-authors with a shorter collaboration timespan. On the other hand, the absence of strong correlations between link weight and link betweenness indicates that non-frequent collaborations are equally important to the connectedness of components as frequent collaborations.

TABLE 5.11: Values of Spearman's correlation coefficient for link (collaboration) metrics.

	Weight	Timespan	Authors in common
Timespan	0.78		
Authors in common	0.05	0.02	
Betweenness	0.17	0.19	-0.15

[8]The paper has title "Une simple méthode pour obtenir la décomposition effective de Wold dans le cas des chaines de Markoff de corélations stationnaires", and is published in "Matematički Vesnik".

5.4.2.5 Communities in the eLib co-authorship network

In order to select the best community detection method for our case study we initially investigated performance of five different community detection methods on the largest connected component. Results are presented in Table 5.12. It can be observed that the Louvain method shows the best performance for our network: this method reveals a community partition having the highest modularity and the largest percentage of Radicchi strong communities.

TABLE 5.12: Comparative analysis of performance of different community detection methods applied to the largest connected component: C – the number of detected communities, Q – modularity score, Strong – the percentage of Radicchi strong communities.

Method	C	Q	Strong [%]	Reference
Girvan-Newman edge betweenness	11	0.813	72.7	[Girvan and Newman, 2002]
Walktrap	23	0.824	82.6	[Pons and Latapy, 2006]
Infomap	30	0.802	66.7	[Rosvall and Bergstrom, 2007]
Label propagation	29	0.803	79.3	[Raghavan et al., 2007]
Louvain	16	0.834	93.7	[Blondel et al., 2008]

Since the Louvain method shows the best performance on the largest connected component we selected this method to investigate the community structure of ten largest connected components in the network. Results are summarized in Table 5.13. It can be observed that for each component the value of the modularity measure Q is higher than 0.3. Usually a value of Q larger than 0.3 is considered as a clear indication that the network possesses community organization according to the modularity based definition of community [Fortunato and Barthélemy, 2007]. Moreover, the modularity score of the five largest eLib components is even higher than 0.5, and the largest component has the largest value of modularity.

TABLE 5.13: Results of community detection for ten largest connected components in the eLib co-authorship graph: N – the number of nodes in the component, Q – modularity score, C – the number of detected communities.

N	Q	C	N	Q	C
249	0.834	16	21	0.503	4
74	0.716	8	19	0.486	3
37	0.507	4	19	0.500	4
27	0.531	5	18	0.435	5
25	0.583	4	17	0.334	3

To investigate the quality of obtained community partitions we examine in detail the communities detected in the three largest connected components. Figure 5.10 shows the visualization of the largest connected component after community detection, while Table 5.14 provides a description of the obtained communities. The largest cohesive subgroup is organized around Ivan Gutman who is the best connected eLib author and the most productive author. The central figure in the second largest community is Žarko Mijajlović who is the most central author according to the betweenness centrality metric. The third largest community which is organized around Jovan Karamata (1902–1967) encompasses the oldest generation of authors present in eLib journals, also including Paul Erdős. From this community the whole component started to emerge: the first collaboration among eLib authors is the collaboration between Jovan Karamata and Hermann Wendelin which was established in 1934. It can be observed that for each detected community the number of intra-community links (denoted

by "IntraL" in Table 5.14) is significantly higher than the number of inter-community links (denoted by "InterL"). The same holds also for the sum of weights of intra-community ("IntraW") and inter-community ("InterW") links which means that the overall strength of collaboration among members of each community is higher than the strength of collaboration among authors belonging to different communities. Moreover, each of the detected communities, except community C6, is Radicchi strong which means that each author from a community collaborates more often with authors from his/her community than with authors from other communities. In case of community C6 there are only two authors who are not Radicchi strong: (1) Slobodan Simić has 9 joint publications with members of his community and 10 joint publications with members of communities C1 and C5, and (2) Vlajko Kocić has 1 joint publication with Slobodan Simić and 3 joint publication with Jovan Kečkić who belongs to community C5. For the majority of detected communities (all of them except for C3, C5 and C6) the author having the highest degree centrality in the community (shown in Table 5.14) is at the same time the author who is most central according to the betweenness centrality metric.

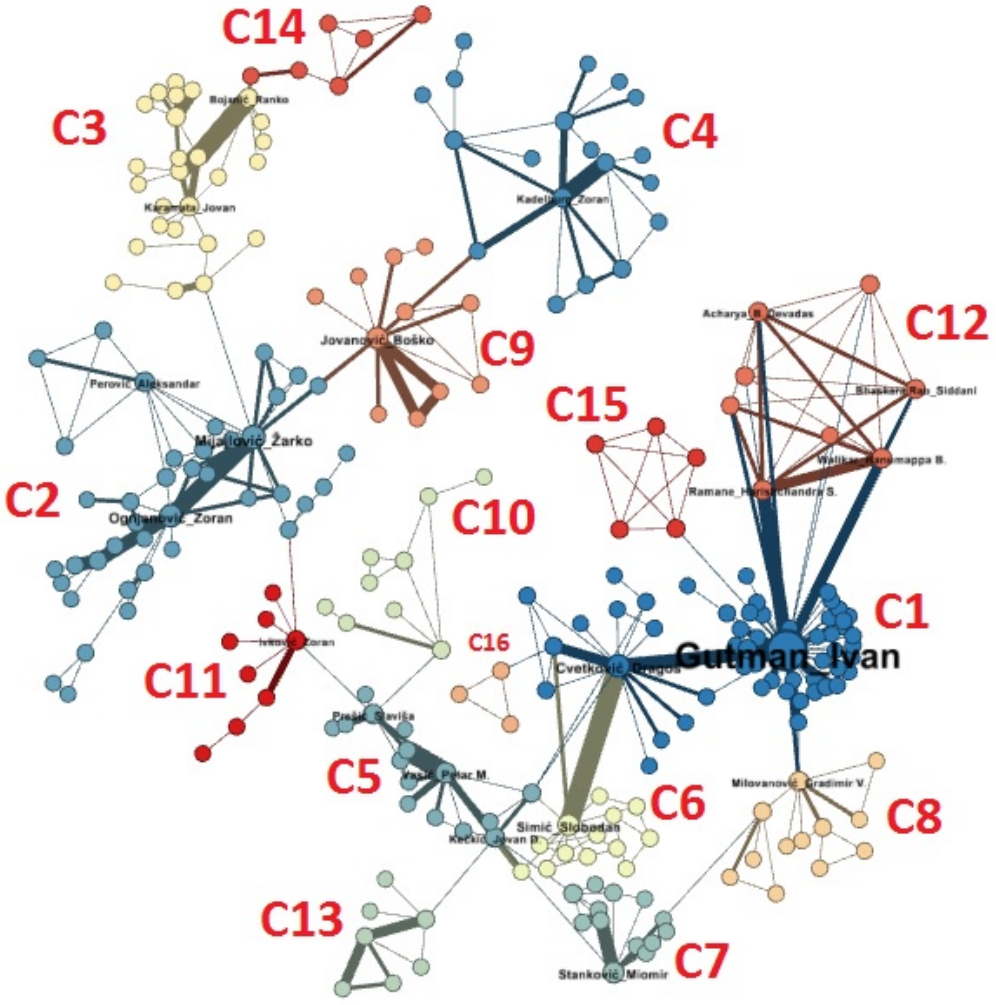


FIGURE 5.10: Visualization of the largest connected component in the eLib co-authorship graph. Nodes from the same community are in the same color. Additionally, each community is marked with an appropriate identifier (C1, C2, etc.) used in Table 5.14.

Figure 5.11 shows the structure of the second largest connected component after community detection. The characteristics of the partition are given in Table 5.15. It can be observed that for each detected community the number of intra-community links is significantly higher than the number of

TABLE 5.14: Description of detected communities for the largest connected eLib component.

Community	Size	Max. degree author	IntraL	InterL	IntraW	InterW	Strong
C1	54	Ivan Gutman (50)	82	15	108	33	yes
C2	40	Žarko Mijajlović (16)	66	4	106	6	yes
C3	26	Jovan Karamata (8)	35	2	64	3	yes
C4	19	Zoran Kadelburg (7)	25	1	42	2	yes
C5	15	Petar M. Vasić (10)	23	8	42	10	yes
C6	13	Slobodan Simić (12)	20	4	20	13	no
C7	13	Miomir Stanković (11)	23	3	34	3	yes
C8	12	Gradimir Milovanović (8)	15	3	18	4	yes
C9	11	Boško Jovanović (12)	14	3	26	6	yes
C10	9	Jovan Petrić (5)	10	1	11	1	yes
C11	8	Zoran Ivković (8)	7	2	9	2	yes
C12	8	Ramane Harishchandra (8)	21	8	31	18	yes
C13	7	Svetozar Milić (5)	8	1	16	1	yes
C14	6	Snežana Pejović (4)	8	1	10	2	yes
C15	5	Song Zhang (5)	10	1	10	1	yes
C16	3	Bolian Liu (3)	3	1	3	1	yes

inter-community links. The same also holds for the sum of weights of this two types of links. Moreover, each detected community is Radicchi strong which clearly suggests that the applied community detection technique produced a good partition into communities. The authors having the highest degree centrality in communities denoted by C1, C4, C5, C6 and C8 are Serbian mathematicians affiliated with the University of Novi Sad. Community C5 is organized around Bogoljub Stanković, a Serbian Academician from Novi Sad, who is the author with the maximal value of timespan for the whole network in the examined time period: the first paper of Bogoljub Stanković published in eLib journals is from 1953, while the last one is from 2011. For 6 out of 8 communities (all except C2 and C7) the author having the highest degree in the component is also the author with the highest betweenness centrality. The authors having the maximal betweenness centrality in C2 and C7 are Miroslava Petrović-Torgašev and Ratko Tošić, respectively.

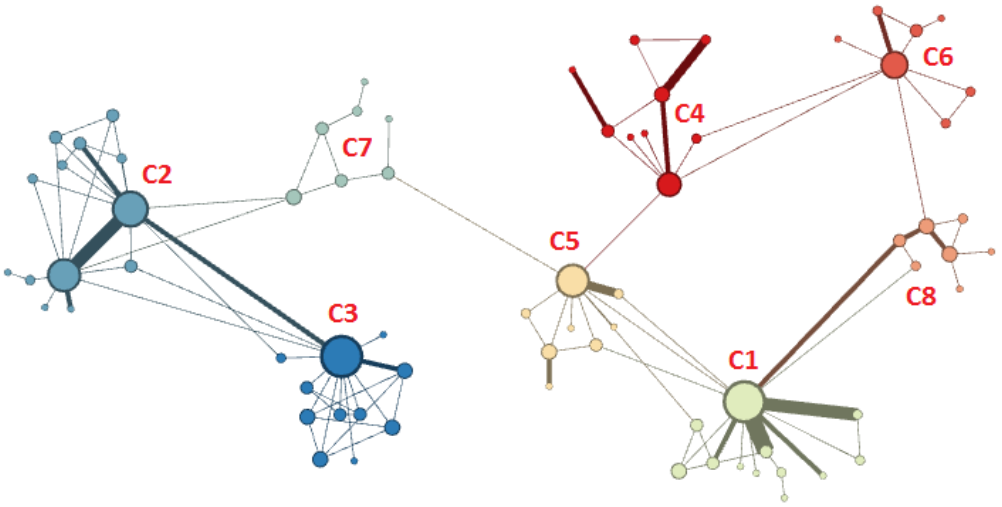


FIGURE 5.11: Visualization of the second largest connected component in the eLib co-authorship graph after community detection.

The third largest connected component in the eLib co-authorship network encompasses eLib authors who published their papers in two eLib journals: "Computer Science and Information Systems" and "Review of the National Center for Digization". The scope of mentioned journals is not purely

TABLE 5.15: Description of detected communities for the second largest connected eLib component.

Community	Size	Max. degree author	IntraL	InterL	IntraW	InterW	Strong
C1	14	Stevan Pilipović (13)	18	5	28	6	yes
C2	13	Leopold Verstraelen (11)	19	6	25	7	yes
C3	11	Ryszard Deszcz (13)	19	4	20	5	yes
C4	9	Dragoslav Herceg (7)	10	3	14	3	yes
C5	8	Bogoljub Stanković (10)	9	6	12	6	yes
C6	7	Djurdjica Takači (8)	8	3	9	3	yes
C7	7	Mirjana Djorić (4)	7	3	7	3	yes
C8	5	Arpad Takači (5)	5	2	6	3	yes

mathematical, but oriented to applications of mathematics and computer science, where the number of authors per paper is generally higher compared to pure mathematical research. Consequently, this component is denser than the previously two described connected components. The details of obtained communities for the third largest component are provided in Table 5.16. It can be observed that all detected communities are Radicchi strong. Additionally, for each component the author having the highest degree centrality has the highest betweenness centrality.

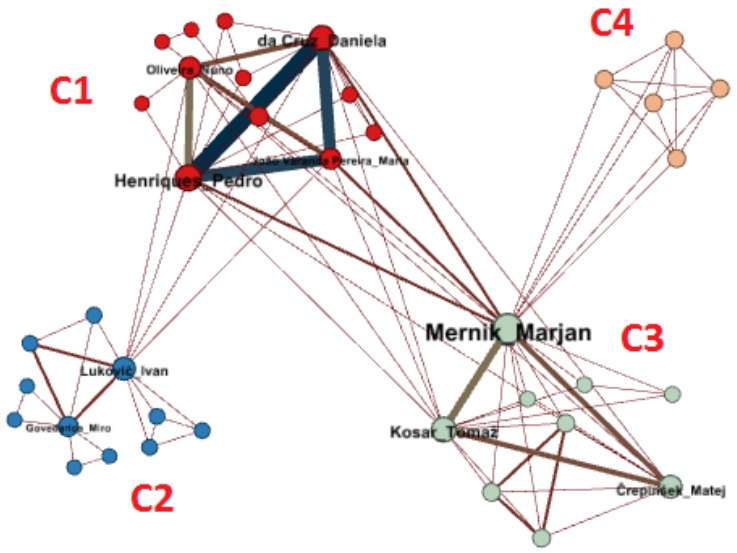


FIGURE 5.12: Visualization of the third largest connected component in the eLib co-authorship graph after community detection.

TABLE 5.16: Description of detected communities for the third largest connected eLib component.

Community	Size	Max. degree author	IntraL	InterL	IntraW	InterW	Strong
C1	12	Pedro Henriques (13)	25	16	49	19	yes
C2	11	Ivan Luković (10)	18	4	21	4	yes
C3	9	Marjan Mernik (17)	23	17	33	20	yes
C4	5	Bryant R. Barrett (5)	10	5	10	5	yes

5.4.2.6 The evolution of the eLib co-authorship network

The eLib co-authorship network evolved from 11 isolated nodes (authors) in 1932 to 3597 nodes and 2766 links in 2011. The first co-authorship link appeared in 1934 connecting authors Jovan Karamata and Hermann Wendelin who co-authored the paper titled "Zu Fragen über nichtvertauschbare Grenzprozesse." Figure 5.13 shows the evolution of the fraction of isolated nodes and the ratio between

the number of links and non-isolated nodes. It can be observed that the fraction of isolated nodes is continuously decreasing after 1949:

- In 1949 92% of the total number of authors have not collaborated with any other author by publishing papers in the eLib journals,
- In 1997 less than half of authors are isolated,
- In 2011 only 33% of the total number of authors are those who exclusively publish single-authored papers in the eLib journals.

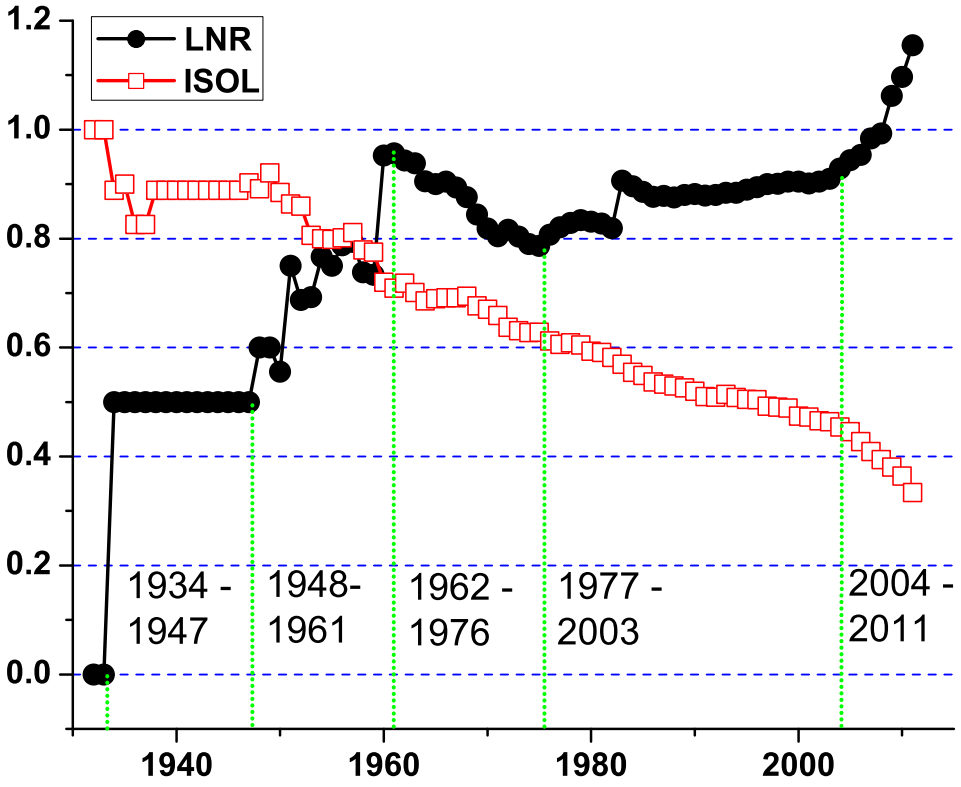


FIGURE 5.13: The evolution of the ratio between the number of links and non-isolated nodes (LNR), and the fraction of isolated nodes (ISOL) in the eLib co-authorship network.

The evolution of the ratio between the number of links and non-isolated nodes (LNR) enables us to observe different periods in the evolution of the eLib co-authorship network that are characterized by different intensity of collaborations among eLib authors. If LNR increases as a network evolves then the number of links (collaborations) grows at a faster rate than the number of non-isolated nodes ("collaborative" authors), i.e. the degree of collaborative behaviour among authors is increasing. On the contrary, the decrease of LNR implies the decrease of collaborative behaviour. As can be seen in Figure 5.13, six periods in the evolution of the eLib co-authorship network can be distinguished:

1. Before 1934, all published papers in the eLib journals are single-authored papers, which means that this period characterizes the absence of collaborative behaviour among eLib authors.
2. In the period from 1934 to 1947, collaborations among eLib authors started to emerge. In this period LNR is equal to 0.5 which means that all connected components have size 2. However, only two non single-authored papers were published in this period, that is four authors collaborated with others, and there were no authors who collaborated with more than one author.

3. The period from 1948 to 1961 was characterized by an intensive growth of collaborations among eLib authors: LNR increased from 0.5 to 0.96. The number of non-isolated authors by the end of 1961 is 66 and those authors are connected by 69 links, where 64 links (97%) represent collaborations established in this period.

4. In the fourth period (1962–1976) LNR decreased from 0.95 to 0.81 implying lower intensity of collaborative behaviour among authors in comparison with the previous period. By the end of 1976 the number of non-isolated authors is 290. Those authors are connected by a significantly smaller number of links (234).

5. In the next period (1977–2003) LNR increased from 0.81 to 0.91 indicating a period when collaborations among authors again started to intensify. By the end of this period the network contained 1151 non-isolated authors that were connected by 1047 links.

6. The last period (2004–2011) has the same characteristics as the previous period (the number of links grows faster than the number of non-isolated nodes). However, in this period LNR grows at a faster rate than in all previous periods implying that the collaborative behaviour among eLib authors is the most intensive in the last years of eLib evolution. Additionally, in this period, for the first time in the evolution of the eLib co-authorship network, LNR became greater than one, denoting that the network contained more links than non-isolated nodes.

In order to determine the dominant type of collaboration for each of the last four characteristic periods in the evolution of the eLib co-authorship network, we computed the number of collaborations between old (returning) authors, old and new authors, and new authors. It is important to notice that in the computation of the number of collaborations link weights have to be considered (see Section 5.4.2). Table 5.17 shows the number of collaborations per type for different periods in eLib evolution. It can be seen that the dominant type of collaboration in periods 1948–1961 and 1977–2003 are collaborations between returning authors, in the period 1962–1976 collaborations between returning and new authors, and in the last years the majority of collaborations are formed by new authors. This means that the periods with different intensity of collaboration among eLib authors are additionally characterized by different types of collaborative behaviour.

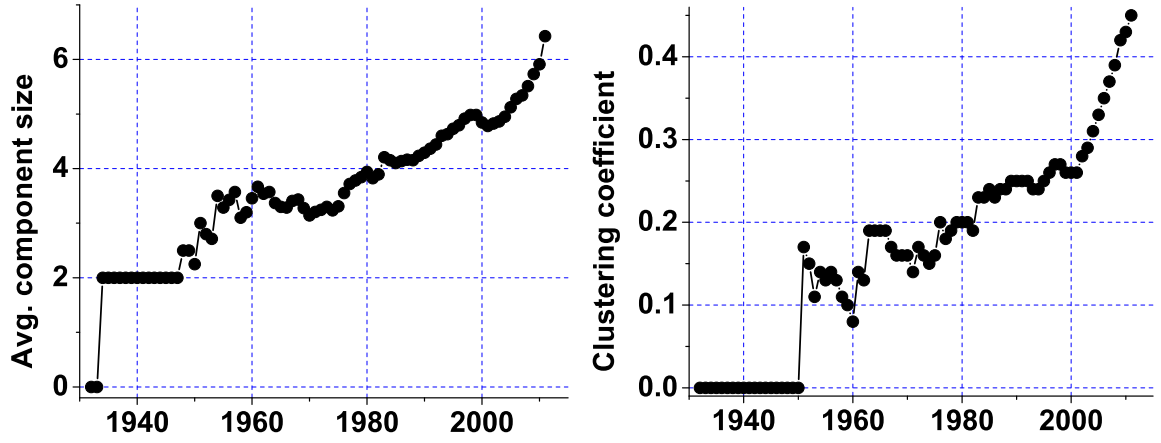
TABLE 5.17: The number of collaborations between old authors (Old-Old), old and new authors (Old-New) and new authors (New-New) for the last four characteristic periods in the evolution of the eLib co-authorship network. The most dominant types of collaborations are bold.

Period	Old-Old		Old-New		New-New	
1948–1961	39	**43.82%**	19	21.35%	31	34.83%
1962–1976	66	32.04%	79	**38.35%**	61	29.61%
1977–2003	372	**35.23%**	331	31.34%	353	33.43%
2004–2011	487	23.08%	543	25.73%	1080	**51.18%**

As already mentioned, all components in the co-authorship network are either trivial or non-trivial. Since trivial components represent collaborations that resulted from publishing exactly one paper, their main characteristic is that they have not evolved in the examined time range (1932–2011). For each year in the network evolution we measured the average size (Figure 5.14(a)) and clustering coefficient (Figure 5.14(b)) of non-trivial components. It can be observed that after 1970 connected components tend to be larger and more cohesive. The increase of average size and cohesiveness

of non-trivial components corresponds to the increase of intensity of collaborations in the last two characteristic periods of the network evolution (see Figure 5.13). The average size of non-trivial components in 1970 is equal to 3.14, while in 2011 it is more than two times larger (6.43). Similarly, the clustering coefficient of non-trivial components in 1970 is 0.16, while in 2011 it is nearly three times larger (0.45). Additionally, in the last decade of the network evolution (after 2001) both average size and clustering coefficient grow at higher rates than in previous years:

- The average size of non-trivial components increased from 3.14 in 1970 to 4.84 in 2000, which is average increase of 0.057 per year. In the last decade average increase per year of the average size of non-trivial components is 0.16 (2.8 times higher than in the period from 1970 to 2000).
- The clustering coefficient of non-trivial components increased from 0.16 in 1970 to 0.26 in 2000, which is average increase of 0.003 per year. The same quantity had average increase of 0.02 in the last decade of the network evolution.



(A) (B)

FIGURE 5.14: The evolution of the average component size (a), and clustering coefficient (b) for non-trivial components in the eLib co-authorship graph.

As already shown in Section 5.4.2.4, betweenness centrality is a better indicator of author productivity and long-term presence in the eLib journals than degree in the co-authorship graph. We continue by presenting the analysis of the evolution of the strength of correlations between authors' centrality metrics and metrics of productivity and long-term presence in the eLib journals. For each year y in the evolution of the eLib co-authorship network, and each author A that was present in year y, we computed vector $V_y(A) = \langle n_y, t_y, d_y, b_y \rangle$, where n_y, t_y, d_y and b_y denote the number of papers, timespan, degree centrality and betweenness centrality of author A in year y, respectively. In that way a sequence of author metrics vectors per year is computed, which enabled us to investigate the evolution of correlations between author metrics. During the whole examined period there is a strong correlation between author productivity and long-term presence: Spearman's correlation coefficient between the number of published papers and timespan is always greater than 0.9 after 1949. Figure 5.15 shows the evolution of Spearman's correlation coefficient between centrality metrics and metrics of productivity. It can be seen that only between 1950 and 1954 degree centrality had stronger correlations with productivity/long-term presence than betweenness centrality. More importantly, the strength of correlations between betweenness centrality and productivity/long-term presence is continuously increasing after 1970. Thus we can expect even stronger correlations between

mentioned quantities in the future. To the contrary, the strength of correlations between degree and productivity/long-term presence started to decrease in 1997.

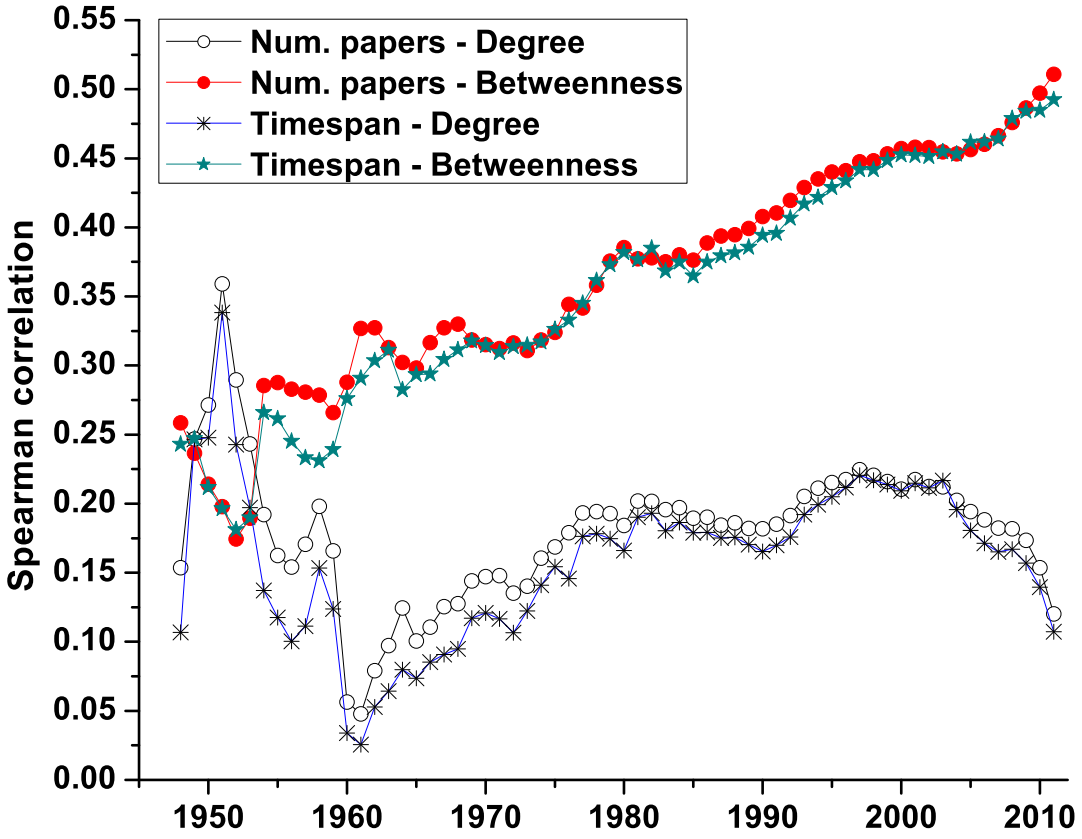


FIGURE 5.15: The evolution of Spearman's correlations between co-authorship network based author metrics (degree and betweenness centrality) and metrics of productivity (the number of publications and author timespan) for eLib authors.

Another interesting aspect of the co-authorship network evolution is the change of the top-ranked author according to certain metric as the network evolves. Therefore, we determined the most productive, the best connected and the most central author for each five-year period in the evolution of the eLib co-authorship network. Results are summarized in Table 5.18. Mihailo Petrović (1868–1943), who is generally considered as one of the most prominent Serbian mathematician, is the most productive author in the first years of eLib evolution (1932–1945). In total he published 15 papers in the eLib journals, where three papers were published posthumously (after 1943). All of his eLib articles are single-authored papers. The next two decades are marked by the dominance of Jovan Karamata (1902–1967) in the eLib journals. From 1950 to 1970 this famous Serbian mathematician is at the same time the most productive, the most connected and the most central eLib author. He published 34 papers in the eLib journals in the period from 1932 to 1960. This means that his dominance lasted for a whole decade after he published his last paper. In the 1980s the most productive author is Djuro Kurepa (1907–1993), another famous Serbian mathematician known for his contributions to set theory and mathematical logic (especially the Kurepa tree). He published 45 papers in the eLib journals in the period from 1935 to 1989, but had only one collaborator among eLib authors (44 of his 45 papers published in the eLib journals are single-authored research works). In the first half of the first decade of the 21st century the most productive author is Bogoljub Stanković, a Serbian Academician born in 1924, who is still an active mathematician. He published 64 papers in the period from 1953 to 2011,

and had 10 collaborators who are mathematicians mostly affiliated with the University of Novi Sad. From 1975 to 2000 the most, or one of the most, connected eLib authors is Petar M. Vasić who published 23 articles in the eLib journals in collaboration with 10 other authors. In 1985 the same author was the most central actor in the eLib co-authorship network. For other years in the period after 1975 the most central eLib authors are Slaviša Prešić (1933–2008) and his student Žarko Mijajlović. The most productive and the best connected author in the last years of eLib evolution is Ivan Gutman. According to the Mathematics Genealogy Project,[9] all eLib authors present in Table 5.18, except Djuro Kurepa who was a student of Maurice René Fréchet, are descendants of Mihailo Petrović.

TABLE 5.18: The top ranked author according to the number of papers, degree, and betweenness centrality in different periods of eLib evolution.

Year	Max. papers		Max. Degree		Max. betweenness	
1932	**3 authors**	2	-	0	-	0
1935	Mihailo Petrović	7	**2 authors**	1	-	0
1940	Mihailo Petrović	12	**4 authors**	1	-	0
1945	Mihailo Petrović	12	**4 authors**	1	-	0
1950	Jovan Karamata	16	Jovan Karamata	2	Jovan Karamata	1
1955	Jovan Karamata	28	Jovan Karamata	5	Jovan Karamata	25
1960	Jovan Karamata	34	Jovan Karamata	8	Jovan Karamata	61
1965	Jovan Karamata	34	Jovan Karamata	8	Jovan Karamata	61
1970	**2 authors**	34	Jovan Karamata	8	Jovan Karamata	68
1975	**2 authors**	34	Petar M. Vasić	9	Slaviša Prešić	113
1980	Djuro Kurepa	38	Petar M. Vasić	10	Slaviša Prešić	261
1985	Djuro Kurepa	42	**6 authors**	10	Petar M. Vasić	485
1990	Djuro Kurepa	45	**6 authors**	10	Slaviša Prešić	615
1995	Djuro Kurepa	45	**7 authors**	10	Slaviša Prešić	1744
2000	Bogoljub Stanković	48	**7 authors**	10	Slaviša Prešić	2482
2005	Bogoljub Stanković	55	Ivan Gutman	23	Slaviša Prešić	6620
2010	Ivan Gutman	64	Ivan Gutman	45	Žarko Mijajlović	15895
2011	Ivan Gutman	71	Ivan Gutman	50	Žarko Mijajlović	17983

The weight of a link in the eLib co-authorship network represents the number of papers two authors connected by the link published together in the eLib journals. Figure 5.16 shows the evolution of the average link weight for links contained in non-trivial connected components. It can be observed that only in two relatively short periods of time (from 1949 to 1956, and from 1976 to 1982) the average link weight exhibits an increasing trend. The increase of the average link weight denotes the intensification of already established collaborations. For example, from 1949 to 1956 the average link weight increased from 1.0 to 2.0 meaning that each two authors who established their first collaboration before 1956 renewed the collaboration once again by the end of 1956 on average.

5.5 Summary

Scientific journals printed in Serbia are in the majority of cases covered insufficiently by the global and widely used digital libraries and bibliography database systems. The electronic library project of the Mathematical Institute of the Serbian Academy of Sciences and Arts (eLib) was started to fill

[9] http://www.genealogy.math.ndsu.nodak.edu/

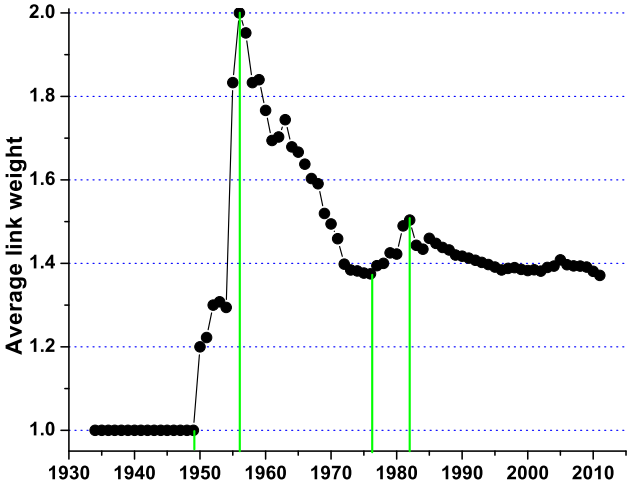


FIGURE 5.16: The evolution of average link weight for non-trivial connected components in the eLib co-authorship graph.

this gap and provide online presence and long-term preservation of mathematical journals printed in Serbia.

We investigated the structure and evolution of the eLib co-authorship network that is determined by papers published in the eLib journals in the period from 1932 to 2011. Techniques and measures used in the analysis of social networks are employed in order to reveal structural properties and evolutionary trends in collaborations among eLib authors. Additionally, we provided the context in which the network formed by investigation of publication dynamics in the eLib journals (number of papers and authors per year), characteristics of published papers (number of authors per paper and fraction of single-authored papers) and characteristics of present authors (number of papers per author and fraction of returning authors).

The analysis of connected components in the network revealed a topological diversity in the network structure that is characterized by the absence of a giant connected component, and power-law scaling behaviour regarding the size of components and the number of papers published by authors from the same component. Additionally, basic structural properties of co-authorship networks that are restricted to individual journals indexed by eLib are investigated in order to identify differences between them.

We showed that the largest connected components of the eLib co-authorship graph possess clear community structure. This means that authors belonging to the largest components are organized into non-overlapping cohesive subgroups. Additionally, we showed that the majority of identified groups tend to be strong in the sense that each author from a group collaborates more often with authors from his/her group than with authors from other groups.

Evolutionary analysis of the eLib co-authorship network revealed that there are six different periods in the evolution of the network that are characterized by different intensity and type of collaborative behaviour among eLib authors. In the last two periods (from 1975) the intensity of collaborations exhibits a growing trend, and non-trivial connected components evolve in a way to become larger and more cohesive. This means that not only are new authors being integrated into non-trivial components, but also authors who have a co-author in common started to collaborate between themselves. Therefore, our findings for mathematical journals printed in Serbia are similar to those reported

by Grossman [2002a] and Brunson et al. [2014] who observed a definite trend toward increasing collaboration in more recent times among mathematicians who publish their research work in journals indexed by "Mathematical Reviews".

We combined metrics used in analysis in social networks (degree and betweenness centrality, clustering and small-world coefficient) and metrics of productivity (the number of published papers and author timespan) to numerically represent characteristics of eLib authors. The analysis of author metrics showed that betweenness centrality is a better indicator of author productivity and long-term presence in the eLib journals than degree centrality. Additionally, evolutionary study of correlations between centrality and productivity metrics revealed that the strength of correlation between productivity metrics and betweenness centrality increases as the network evolves suggesting that even more stronger correlation can be expected in the future. We also investigated the change of the top-ranked eLib authors by co-authorship based metrics and metrics of productivity for each 5-year interval in the examined time period. Not surprisingly, eight widely recognized Serbian mathematicians are identified as the top-ranked eLib authors in different periods of time, five of whom are/were members of the Serbian Academy of Sciences and Arts.

Chapter 6

Conclusions and future work

The research presented in this dissertation investigated techniques for extraction and analysis of complex networks representing software systems, semantic web ontologies and scientific collaboration. Therefore, its contributions are two-folds. Firstly, we proposed new methods for the extraction of networks from mentioned domains. Secondly, on several case studies we demonstrated the benefits of network-based analysis of concrete systems from those domains. In contrast to the previous work on the subject, analyses presented in this dissertation are not purely topological, but combine techniques and metrics developed under the framework of complex network theory with domain-dependent metrics (software, ontology and metrics of researcher productivity and long-term presence).

The first contribution of this dissertation is SNEIPL [Savić et al., 2012, 2014] – extendable, language-independent extractor of software networks based on the language-independent enriched Concrete Syntax Tree (eCST) representation of source code [Budimac et al., 2012; Rakić and Budimac, 2011a] that is realized as one of the back-ends of the SSQSA framework [Budimac et al., 2012; Rakić et al., 2013]. The applicability of SNEIPL was demonstrated on software systems written in different programming languages which belong to different paradigms. Focusing on software systems written in Java, we showed that networks obtained using SNEIPL are highly similar to those formed by a language-dependent tool and much precise than networks obtained using a language-independent fuzzy parsing approach.

The second contribution of this dissertation is ONGRAM, extractor of ontology networks that are attributed with a rich suite of ontology metrics. ONGRAM is also based on the eCST representation. Therefore, it can be expanded to support different knowledge representation languages. Moreover, the eCST representation of ontological descriptions enabled us to define new and adopt existing software metrics of internal complexity for ontology evaluation.

The extraction of co-authorship networks poses quite different problems compared to the extraction of software and ontology networks. Namely, nodes of software and ontology networks can be easily formed since software/ontology entities are uniquely identified by their (fully qualified) names. On the other hand, author names present in bibliographic records cannot be used to uniquely identify nodes in co-authorship network due to synonymy and homonymy of names. In the dissertation we presented a data-driven methodology for the extraction of co-authorship that includes semi-automatic, heuristically based name disambiguation techniques suitable for sparse and fragmented co-authorship networks.

Using tools that were developed as a part of this dissertation we formed an experimental dataset of complex networks that includes five class collaboration networks associated to software systems

written in Java, three networks associated to one modularized semantic web ontology and a network representing scientific collaboration in Serbian mathematical journals. We showed that examined networks possess the small-world property in the Watts-Strogatz sense. Other properties of (weakly) connected components such as their degrees of strong connectivity, assortativity patterns, degree distributions and characteristics of highly connected nodes vary across domains, systems and levels of abstractions reflecting specificity of the individual systems.

The analysis of strongly connected components (SCCs) present in examined software networks revealed that SCCs tend to densify with size. This densification phenomena can be modeled by power-laws whose scaling exponents can be used as indicators of software design quality. For software and ontology networks we also observed that there is a strong disbalance between in-degree and out-degree of highly coupled entities. Moreover, the extent to which in-degree dominates over out-degree increases with total-degree for the majority of examined networks. This result implies that highly coupled entities in real software/ontological systems are caused dominantly by internal reuse, and consequently that the presence of high coupling can indicate only negative aspects of internal reuse, not negative aspect of internal aggregation. We also introduced the metric-based comparison test which enabled us to investigate in detail the characteristics of strongly connected components and highly coupled nodes (hubs) in software/ontology networks. Finally, in this dissertation we showed that graph clustering evaluation metric can be used as software and ontology cohesion metrics.

As the last contribution of the dissertation, we studied properties of the co-authorship network extracted from bibliographic records contained in the Electronic Library of the Mathematical Institute of the Serbian Academy of Sciences and Arts (eLib) [Savić et al., 2015; Savić et al., 2014]. ELib digitizes the most prominent mathematical journals printed in Serbia. The goal of the study was to identify patterns and long-term trends in scientific collaborations that are characteristic for a community which mainly consists of Serbian (Yugoslav) mathematicians. Analysis of connected components of the network revealed a topological diversity in the network structure: the network contains a large number of components whose sizes obey a power-law, the majority of components are isolated authors or small trivial components, but there is also a small number of relatively large, non-trivial components of connected authors. We observed that the largest connected components of the eLib co-authorship graph possess strong community structure. Our evolutionary analysis showed that the evolution of the network can be divided into six periods that are characterized by different intensity and type of collaborative behavior among eLib authors. Analysis of author metrics showed that betweenness centrality is a better indicator of author productivity and long-term presence in the eLib journals than degree centrality. Moreover, the strength of correlation between productivity metrics and betweenness centrality increases as the network evolves suggesting that even more stronger correlation can be expected in the future.

In the dissertation we mainly focused on the structure of software and ontology networks. In our future work we plan to investigate their evolution. Namely, analysis of the evolution of strongly connected components and hubs of software/ontology networks can provide valuable insights related to the maintenance of software/ontology systems. We will also investigate how the evolution of software/ontology systems affects the structure of software/ontology networks at different levels of abstraction in order to reveal evolutionary patterns that may lead to predictive models of software/ontology evolution. Regarding co-authorship networks, we plan to investigate the coverage, position and importance of Serbian researchers in co-authorship networks constructed from massive, online bibliography databases. In this type of study we will additionally analyze citation networks in order to estimate the influence of Serbian researchers to contemporary science.

Appendix A

Degree distributions of software networks

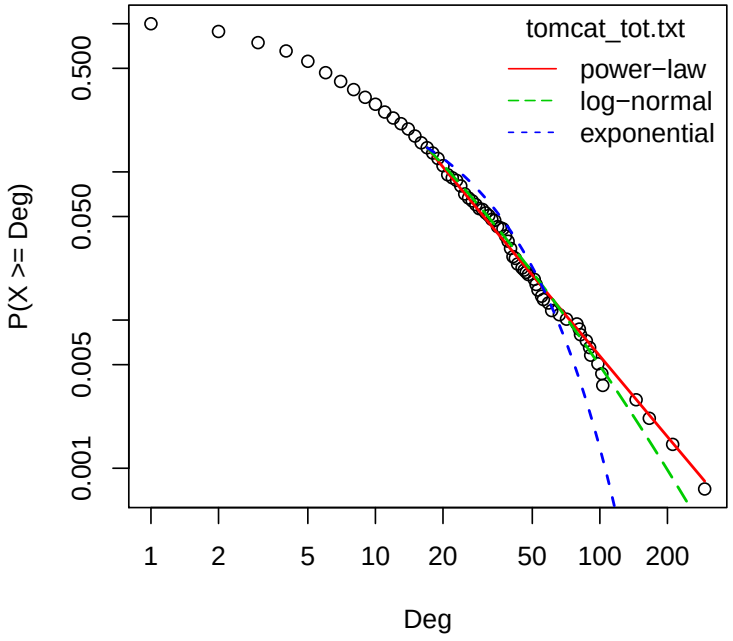


FIGURE A.1: Complementary cumulative degree distribution of the Tomcat class collaboration network.

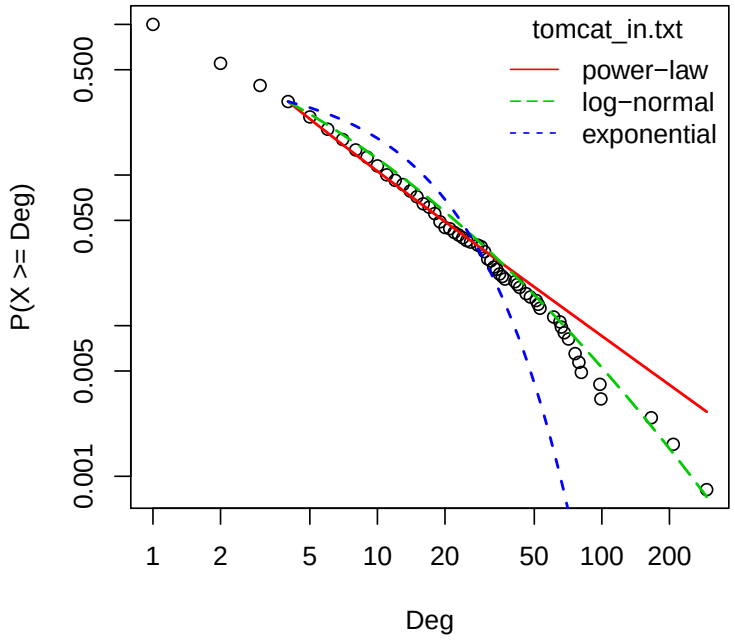


FIGURE A.2: Complementary cumulative in-degree distribution of the Tomcat class collaboration network.

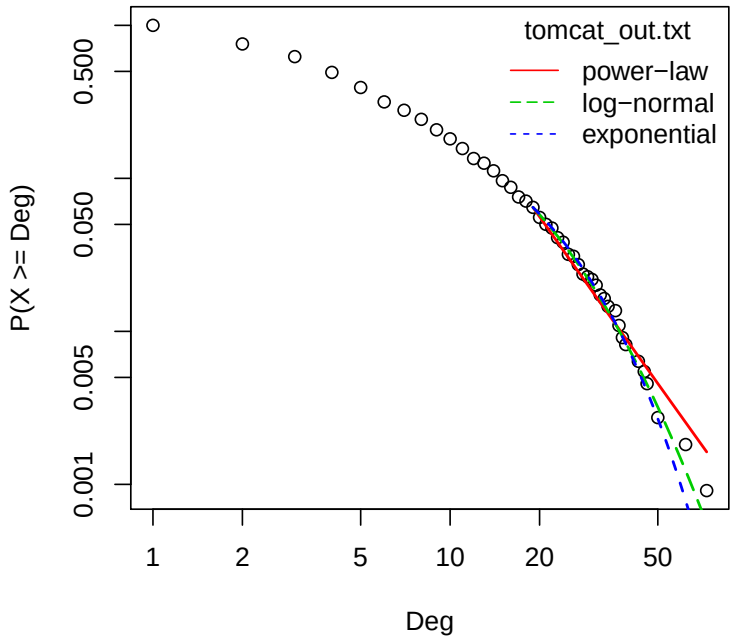


FIGURE A.3: Complementary cumulative out-degree distribution of the Tomcat class collaboration network.

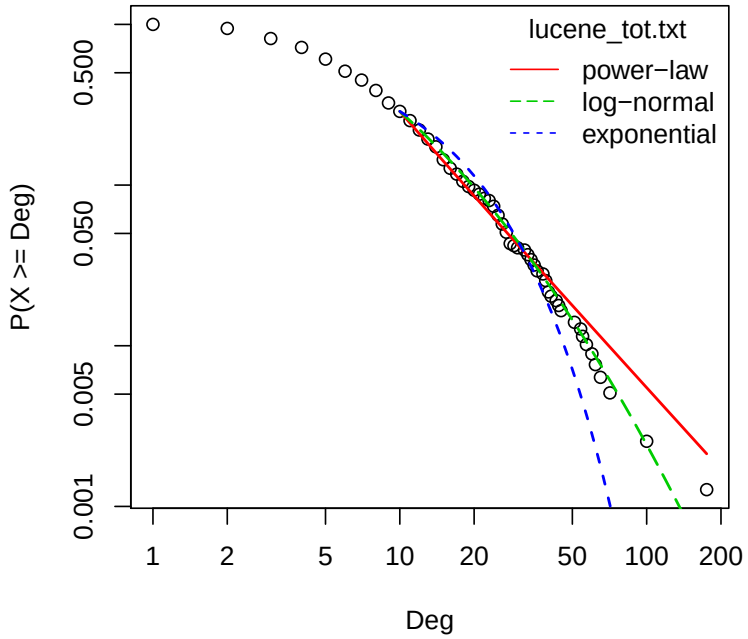


FIGURE A.4: Complementary cumulative degree distribution of the Lucene class collaboration network.

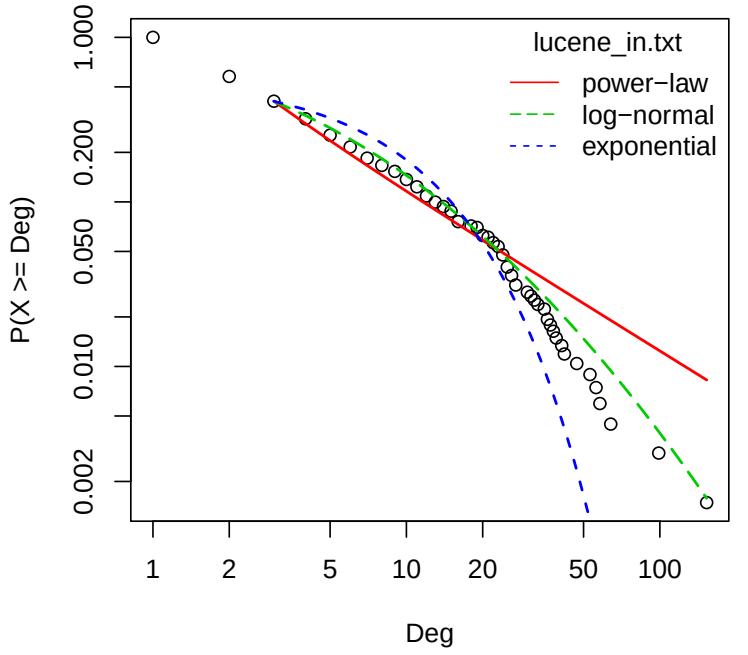


FIGURE A.5: Complementary cumulative in-degree distribution of the Lucene class collaboration network.

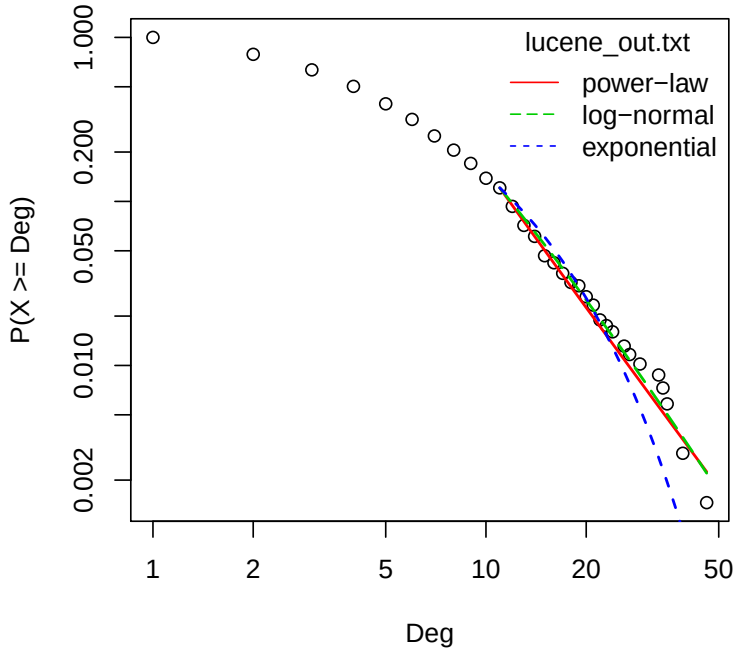


FIGURE A.6: Complementary cumulative out-degree distribution of the Lucene class collaboration network.

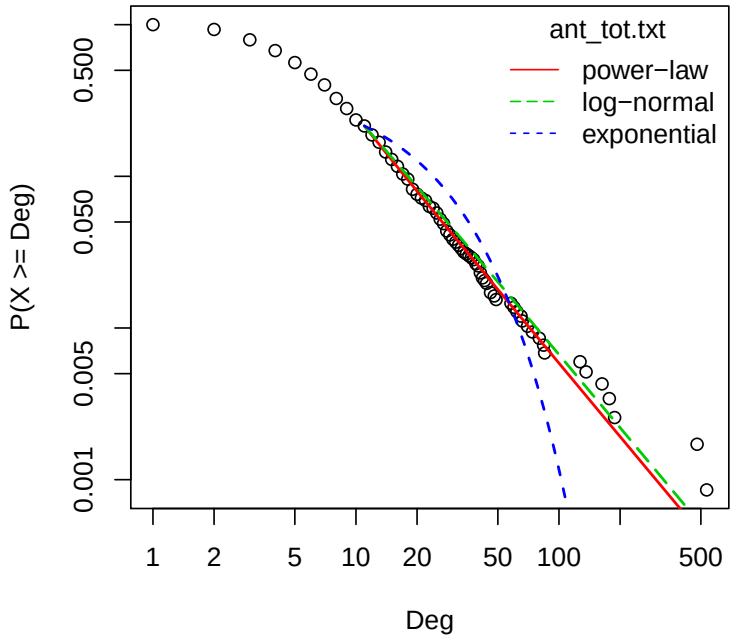


FIGURE A.7: Complementary cumulative degree distribution of the Ant class collaboration network.

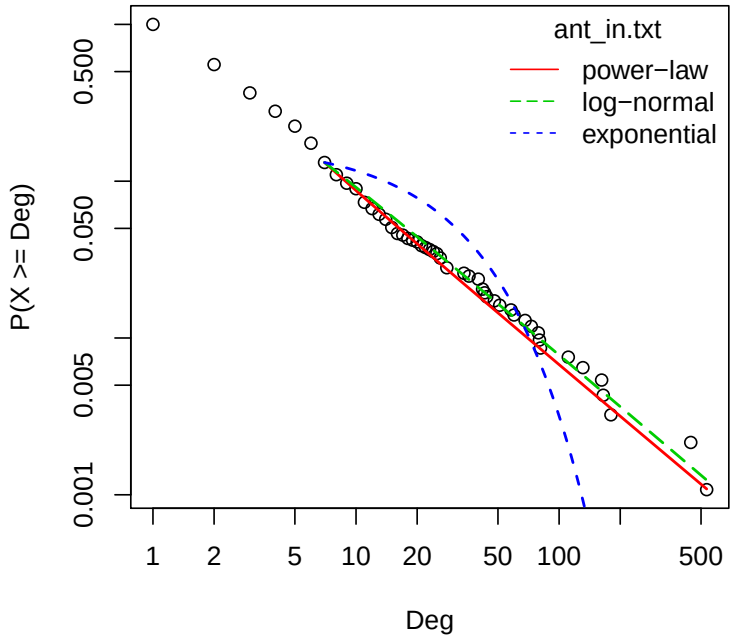


FIGURE A.8: Complementary cumulative in-degree distribution of the Ant class collaboration network.

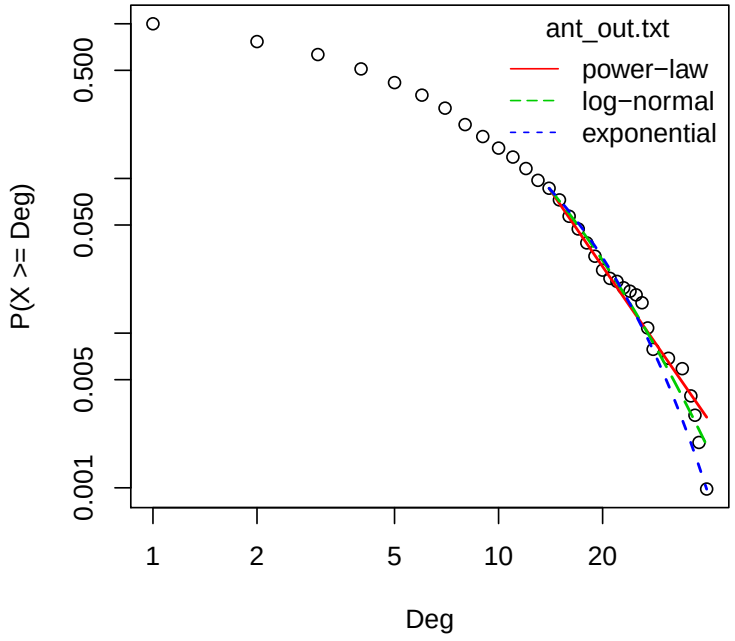


FIGURE A.9: Complementary cumulative out-degree distribution of the Ant class collaboration network.

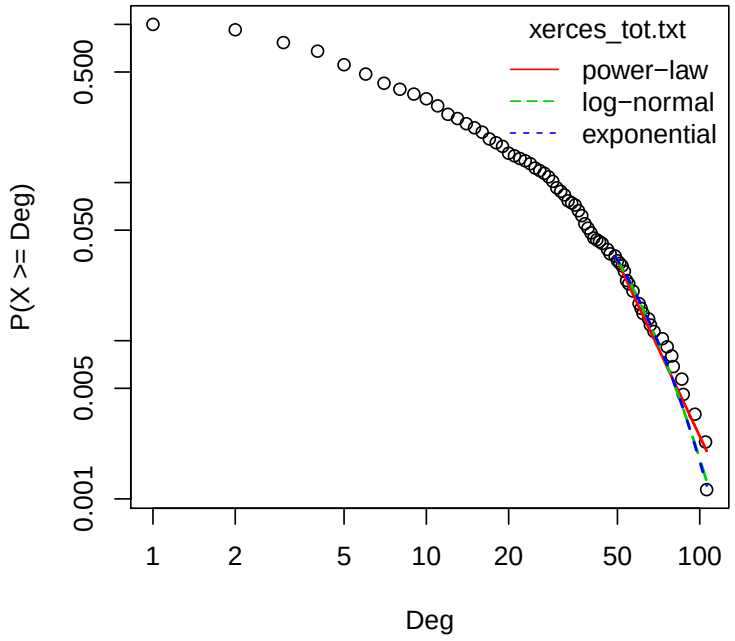


FIGURE A.10: Complementary cumulative degree distribution of the Xerces class collaboration network.

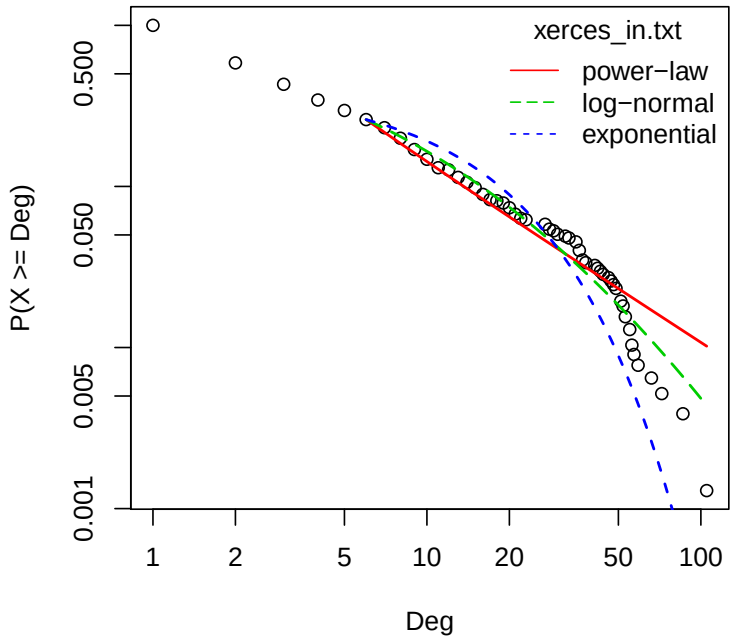


FIGURE A.11: Complementary cumulative in-degree distribution of the Xerces class collaboration network.

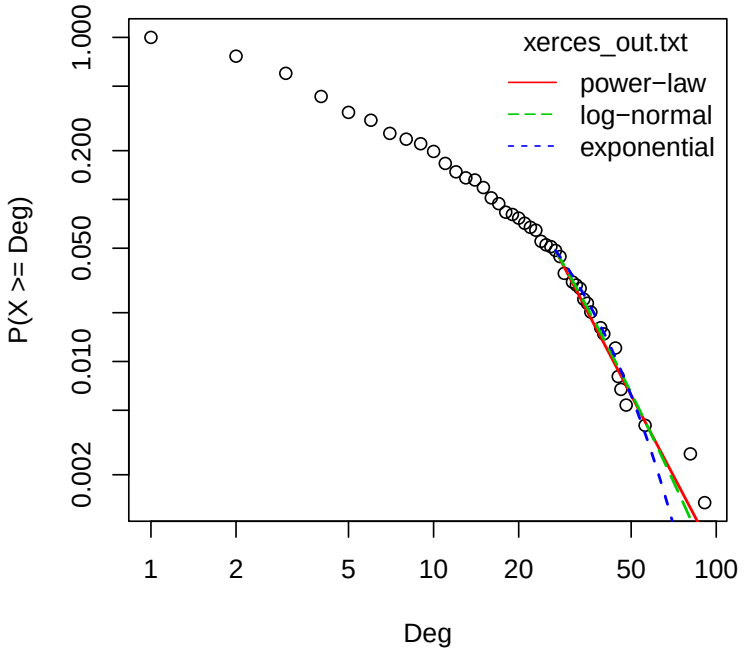


FIGURE A.12: Complementary cumulative out-degree distribution of the Xerces class collaboration network.

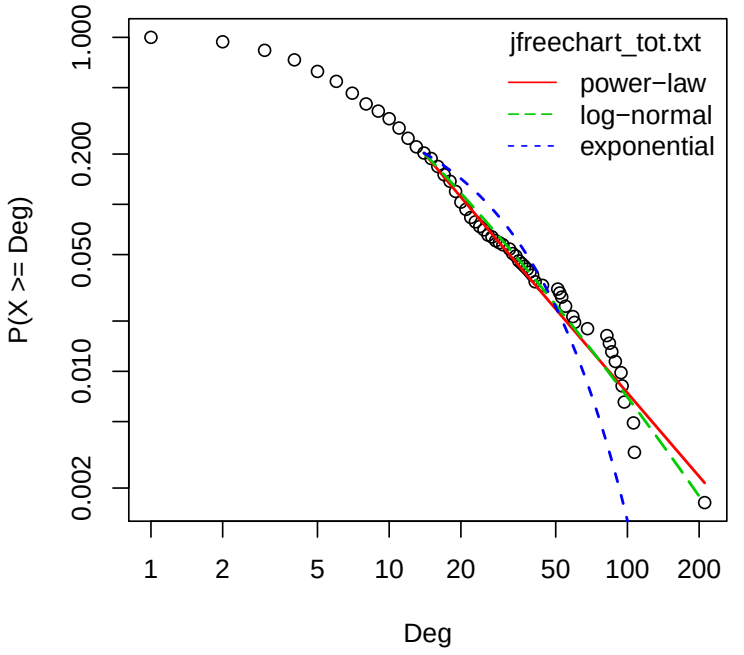


FIGURE A.13: Complementary cumulative degree distribution of the JFreeChart class collaboration network.

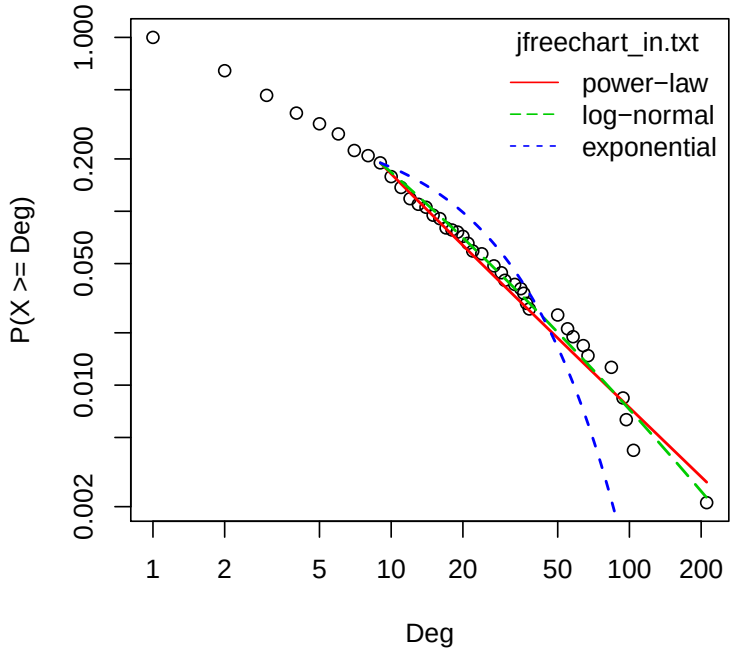


FIGURE A.14: Complementary cumulative in-degree distribution of the JFreeChart class collaboration network.

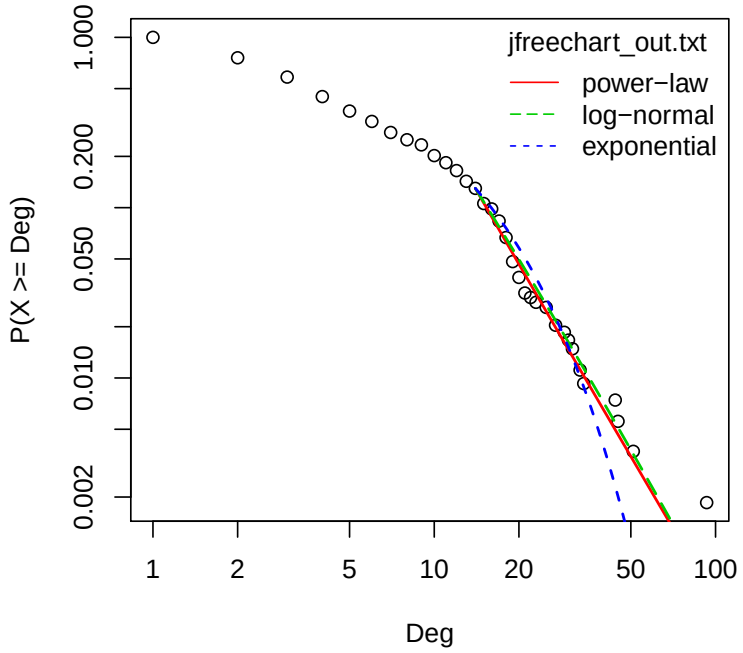


FIGURE A.15: Complementary cumulative out-degree distribution of the JFreeChart class collaboration network.

Bibliography

A. Abbasi, K. S. K. Chung, and L. Hossain. Egocentric analysis of co-authorship network structure, position and performance. *Information Processing & Management*, 48(4):671 – 679, 2012a. ISSN 0306-4573. doi: 10.1016/j.ipm.2011.09.001.

A. Abbasi, L. Hossain, and L. Leydesdorff. Betweenness centrality as a driver of preferential attachment in the evolution of research collaboration networks. *Journal of Informetrics*, 6(3):403 – 412, 2012b. ISSN 1751-1577. doi: 10.1016/j.joi.2012.01.002.

F. J. Acedo, C. Barroso, C. Casanueva, and J. L. Galán. Co-authorship in management and organizational studies: An empirical and network analysis. *Journal of Management Studies*, 43(5):957–983, 2006. ISSN 1467-6486. doi: 10.1111/j.1467-6486.2006.00625.x.

R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1): 47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47.

R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

R. Albert, H. Jeong, and A. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406 (6794):378–382, 2000.

L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, 2000. doi: 10.1073/pnas.200327197.

N. Anquetil, C. Fourrier, and T. C. Lethbridge. Experiments with clustering as a software remodularization method. In *Proceedings of the Sixth Working Conference on Reverse Engineering*, WCRE '99, pages 235–255, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0303-9. doi: 10.1109/WCRE.1999.806964.

D. F. Bacon and P. F. Sweeney. Fast static analysis of C++ virtual function calls. In *Proceedings of the 11th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '96, pages 324–341, New York, NY, USA, 1996. ACM. ISBN 0-89791-788-X. doi: 10.1145/236337.236371.

A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi: 10.1126/science.286.5439.509.

A.-L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311:590–614, 2002.

A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, 2004. doi: 10.1073/pnas.0400087101.

V. Batagelj and M. Cerinšek. On bibliographic networks. *Scientometrics*, 96(3):845–864, 2013. ISSN 0138-9130. doi: 10.1007/s11192-012-0940-1.

V. Batagelj and A. Mrvar. Some analyses of Erdos collaboration graph. *Social Networks*, 22(2): 173–186, May 2000. doi: 10.1016/s0378-8733(00)00023-x.

A. Bavelas. Communication patterns in Task-Oriented Groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 1950. doi: 10.1121/1.1906679.

G. Baxter, M. Frean, J. Noble, M. Rickerby, H. Smith, M. Visser, H. Melton, and E. Tempero. Understanding the shape of java software. In *Proceedings of the 21st Annual ACM SIGPLAN Conference on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '06, pages 397–412, New York, NY, USA, 2006. ACM. ISBN 1-59593-348-4. doi: 10.1145/1167473.1167507.

M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965. ISSN 1099-1743. doi: 10.1002/bs.3830100205.

S. Bechhofer, R. Volz, and P. Lord. Cooking the semantic web with the owl api. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science*, pages 659–675. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20362-9. doi: 10.1007/978-3-540-39718-2_42.

D. Berner, H. Patel, D. Mathaikutty, and S. Shukla. Automated extraction of structural information from SystemC-based IP for validation. In *Sixth International Workshop on Microprocessor Test and Verification (MTV '05)*, pages 99–104, 2005. doi: 10.1109/MTV.2005.8.

T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

Á. Beszédes, R. Ferenc, and T. Gyimóthy. Columbus: a reverse engineering approach. In *Proceedings of the 13th IEEE Workshop on Software Technology and Engineering Practice (STEP 2005)*, pages 93–96. IEEE Computer Society, IEEE Computer Society, 2005.

L. M. A. Bettencourt, D. I. Kaiser, and J. Kaur. Scientific discovery and topological transitions in collaboration networks. *J. Informetrics*, 3(3):210–221, 2009.

P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos. Graph-based analysis and prediction for software evolution. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 419–429, Piscataway, NJ, USA, 2012. IEEE Press. ISBN 978-1-4673-1067-3.

J. M. Bieman and B.-K. Kang. Cohesion and reuse in an object-oriented system. In *Proceedings of the 1995 Symposium on Software Reusability*, SSR '95, pages 259–262, New York, NY, USA, 1995. ACM. ISBN 0-89791-739-1.

C. Bird, E. Barr, A. Nash, P. Devanbu, V. Filkov, and Z. Su. Structure and dynamics of research collaboration in computer science. In *Proceedings of the Ninth SIAM International Conference on Data Mining*, page 826–837. SIAM, April 2009a.

C. Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu. Putting it all together: Using socio-technical networks to predict failures. In *Proceedings of the 2009 20th International Symposium on Software Reliability Engineering*, ISSRE '09, pages 109–119, Washington, DC, USA, 2009b. IEEE Computer Society. ISBN 978-0-7695-3878-5. doi: 10.1109/ISSRE.2009.17.

M. Biryukov and C. Dong. Analysis of computer science communities based on DBLP. In *Proceedings of the 14th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL'10, pages 228–235, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15463-8, 978-3-642-15463-8.

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang. Complex networks: structure and dynamics. *Physics Reports*, 424(4–5):175–308, 2006. ISSN 0370-1573. doi: 10.1016/j.physrep.2005.10.009.

P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014. doi: 10.1080/15427951.2013.865686.

P. Boldi, M. Rosa, and S. Vigna. Robustness of social and web graphs to node removal. *Social Network Analysis and Mining*, 3(4):829–842, 2013. ISSN 1869-5450. doi: 10.1007/s13278-013-0096-x.

B. Bollobás. *Random Graphs*. Cambridge University Press, 2001.

B. Bollobás and O. Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics*, 1(1):1–35, 2003.

B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 132–139, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0-89871-538-5.

B. Bollobás and O. M. Riordan. *Mathematical results on scale-free random graphs*, pages 1–34. Wiley-VCH Verlag GmbH & Co. KGaA, 2005. ISBN 9783527602759. doi: 10.1002/3527602755.ch1.

P. Bonacich. Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology*, 2(1):113–120, 1972. doi: 10.1080/0022250X.1972.9989806.

P. Bonacich. Power and Centrality: A Family of Measures. *American Journal of Sociology*, 92(5):1170–1182, 1987. ISSN 00029602. doi: 10.2307/2780000.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *The Journal of Machine Learning Research*, 6:1579–1619, Dec. 2005. ISSN 1532-4435.

Boris Motik, Peter F. Patel-Schneider and Bijan Parsia (Editors). OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition). http://www.w3.org/TR/owl2-syntax/. Accessed: 2014-12-31.

K. Borner, J. T. Maru, and R. L. Goldstone. The simultaneous evolution of author and paper networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5266–5273, 2004. doi: 10.1073/pnas.0307625100.

U. Brandes. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks*, 30(2):136–145, 2008. doi: 10.1016/j.socnet.2007.11.001.

L. C. Briand, S. Morasca, and V. R. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–86, 1996. ISSN 0098-5589.

L. C. Briand, J. W. Daly, and J. Wüst. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, 3(1):65–117, 1998. ISSN 1382-3256. doi: 10.1023/A:1009783721306.

L. C. Briand, J. W. Daly, and J. K. Wüst. A unified framework for coupling measurement in object-oriented systems. *IEEE Trans. Softw. Eng.*, 25(1):91–121, Jan. 1999. ISSN 0098-5589.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998a. Elsevier Science Publishers B. V.

S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998b.

J. Brooks, F.P. No silver bullet: essence and accidents of software engineering. *Computer*, 20(4): 10–19, 1987. ISSN 0018-9162. doi: 10.1109/MC.1987.1663532.

J. C. Brunson, S. Fassino, A. McInnes, M. Narayan, B. Richardson, C. Franck, P. Ion, and R. Laubenbacher. Evolutionary events in a mathematical sciences research collaboration network. *Scientometrics*, 99(3):973–998, 2014. ISSN 0138-9130. doi: 10.1007/s11192-013-1209-z.

J. Buckner, J. Buchta, M. Petrenko, and V. Rajlich. Jripples: a tool for program comprehension during incremental change. In *Proceedings of the 13th International Workshop on Program Comprehension*, IWPC '05, pages 149–152, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2254-8. doi: 10.1109/WPC.2005.22.

Z. Budimac, G. Rakić, and M. Savić. SSQSA architecture. In *Proceedings of the Fifth Balkan Conference in Informatics*, BCI '12, pages 287–290, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1240-0. doi: 10.1145/2371316.2371380.

K. Börner, L. Dall'Asta, W. Ke, and A. Vespignani. Studying the emerging global brain: Analyzing and visualizing the impact of co-authorship teams. *Complexity*, 10(4):57–67, 2005. ISSN 1099-0526. doi: 10.1002/cplx.20078.

A. Capiluppi and C. Boldyreff. Identifying and improving reusability based on coupling patterns. In H. Mei, editor, *High Confidence Software Reuse in Large Systems*, volume 5030 of *Lecture Notes in Computer Science*, pages 282–293. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-68062-8. doi: 10.1007/978-3-540-68073-4_31.

A. Capiluppi and T. Knowles. Software engineering in practice: design and architectures of FLOSS systems. In C. Boldyreff, K. Crowston, B. Lundell, and A. Wasserman, editors, *Open Source Ecosystems: Diverse Communities Interacting*, volume 299 of *IFIP Advances in Information and Communication Technology*, pages 34–46. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02031-5. doi: 10.1007/978-3-642-02032-2_5.

A. Capiluppi, C. Boldyreff, and K.-J. Stol. Successful reuse of software components: a report from the open source perspective. In S. Hissam, B. Russo, M. Mendonça Neto, and F. Kon, editors, *Open Source Systems: Grounding Research*, volume 365 of *IFIP Advances in Information and Communication Technology*, pages 159–176. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24417-9. doi: 10.1007/978-3-642-24418-6_11.

J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, WWW Alt. '04, pages 74–83, New York, NY, USA, 2004. ACM. ISBN 1-58113-912-8. doi: 10.1145/1013367.1013381.

M. Cerinšek and V. Batagelj. Network analysis of Zentralblatt MATH data. *Scientometrics*, pages 1–25, 2014. ISSN 0138-9130. doi: 10.1007/s11192-014-1419-z.

A. Chatzigeorgiou, N. Tsantalis, and G. Stephanides. Application of graph theory to oo software engineering. In *Proceedings of the 2006 International Workshop on Workshop on Interdisciplinary Software Engineering Research*, WISER '06, pages 29–36, New York, NY, USA, 2006. ACM. ISBN 1-59593-409-X. doi: 10.1145/1137661.1137669.

Y. Chen, K. Börner, and S. Fang. Evolving collaboration networks in Scientometrics in 1978–2010: a micro–macro analysis. *Scientometrics*, 95(3):1051–1070, 2013. ISSN 0138-9130. doi: 10.1007/s11192-012-0895-2.

G. Cheng and Y. Qu. Term Dependence on the Semantic Web. In *Proceedings of the 7th International Conference on The Semantic Web*, ISWC '08, pages 665–680, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88563-4. doi: 10.1007/978-3-540-88564-1_42.

F. Cheong and B. J. Corbitt. A social network analysis of the co-authorship network of the Australasian Conference of Information Systems from 1990 to 2006. In *17th European Conference on Information Systems, ECIS 2009, Verona, Italy, 2009*, pages 292–303, 2009a.

F. Cheong and B. J. Corbitt. A social network analysis of the co-authorship network of the Pacific Asia Conference on Information Systems from 1993 to 2008. In *Pacific Asia Conference on Information Systems, PACIS 2009, Hyderabad, India, July 10-12, 2009*, page 23, 2009b.

S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994. ISSN 0098-5589. doi: 10.1109/32.295895.

E. J. Chikofsky and J. H. Cross II. Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1):13–17, 1990. ISSN 0740-7459. doi: 10.1109/52.43044.

Y. Chiricota, F. Jourdan, and G. Melançon. Software components capture using graph clustering. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension*, IWPC '03, pages 217–226, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1883-4. doi: 10.1109/WPC.2003.1199205.

A. Clauset, C. Shalizi, and M. Newman. Power-law distributions in empirical data. *SIAM Review*, 51 (4):661–703, 2009. doi: 10.1137/070710111.

G. Concas, M. Marchesi, S. Pinna, and N. Serra. Power-Laws in a Large Object-Oriented Software System. *IEEE Transactions on Software Engineering*, 33(10):687–708, 2007. ISSN 0098-5589.

G. Concas, M. Marchesi, A. Murgia, and R. Tonelli. An empirical study of social networks metrics in object-oriented software. *Advances in Software Engineering*, 2010:4:1–4:21, Jan. 2010. ISSN 1687-8655. doi: 10.1155/2010/729826.

G. Coskun, M. Rothe, K. Teymourian, and A. Paschke. Applying community detection algorithms on ontologies for identifying concept groups. In O. Kutz and T. Schneider, editors, *Workshop on Modular Ontologies*, volume 230, pages 12–24. IOS Press, 2011. ISBN 978-1-60750-798-7. doi: 10.3233/978-1-60750-799-4-12.

L. d. F. Costa, O. N. Oliveira, G. Travieso, F. A. Rodrigues, P. R. Villas Boas, L. Antiqueira, M. P. Viana, and L. E. Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011. doi: 10.1080/00018732.2011.572452.

C. Cotta and J. J. M. Guervós. Where is evolutionary computation going? A temporal analysis of the EC community. *Genetic Programming and Evolvable Machines*, 8(3):239–253, 2007. doi: 10.1007/s10710-007-9031-0.

C. Cotta and J. Merelo. The complex network of evolutionary computation authors: An initial study. Preprint available at `http://arxiv.org/abs/physics/0507196`, 2005.

D. M. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs*. Johann Ambrosius Barth, Heidelberg, third edition, 1995. ISBN 3-335-00407-8. Theory and applications.

D. Cvetković and S. Simić. Graph spectra in computer science. *Linear Algebra and its Applications*, 434(6):1545 – 1562, 2011. ISSN 0024-3795. doi: http://dx.doi.org/10.1016/j.laa.2010.11.035.

A. P. S. de Moura, Y.-C. Lai, and A. E. Motter. Signatures of small-world and scale-free properties in large computer programs. *Phys. Rev. E*, 68(1):017102, Jul 2003. doi: 10.1103/PhysRevE.68.017102.

J. Dean, D. Grove, and C. Chambers. Optimization of object-oriented programs using static class hierarchy analysis. In M. Tokoro and R. Pareschi, editors, *Proceedings of the 9th European Conference on Object-Oriented Programming (ECOOP'95)*, volume 952 of *Lecture Notes in Computer Science*, pages 77–101. Springer Berlin Heidelberg, 1995. ISBN 978-3-540-60160-9. doi: 10.1007/3-540-49538-X_5.

L. Ding, J. Shinavier, Z. Shangguan, and D. McGuinness. SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in Linked Data. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web – ISWC 2010*, volume 6496 of *Lecture Notes in Computer Science*, pages 145–160. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-17745-3. doi: 10.1007/978-3-642-17746-0_10.

Y. Ding. Scientific collaboration and endorsement: Network analysis of coauthorship and citation networks. *Journal of Informetrics*, 5(1):187 – 203, 2011. ISSN 1751-1577. doi: 10.1016/j.joi.2010.10.008.

P. Divakarmurthy and R. Menezes. The effect of citations to collaboration networks. In R. Menezes, A. Evsukoff, and M. C. González, editors, *Complex Networks*, volume 424 of *Studies in Computational Intelligence*, pages 177–185. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-30286-2. doi: 10.1007/978-3-642-30287-9_19.

L. Donetti and M. A. Muñoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2004(10):P10012, 2004.

S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Structure of growing networks with preferential linking. *Phys. Rev. Lett.*, 85:4633–4636, Nov 2000. doi: 10.1103/PhysRevLett.85.4633.

J. Dravec Braun. Effects of war on scientific production: mathematics in Croatia from 1968 to 2008. *Scientometrics*, 93(3):931–936, 2012. ISSN 0138-9130. doi: 10.1007/s11192-012-0735-4.

S. Ducasse, M. Lanza, and S. Tichelaar. Moose: an extensible language-independent environment for reengineering object-oriented systems. In *2nd International Symposium On Constructing Software Engineering Tools (COSET 2000)*, 2000.

J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, Aug 2005. doi: 10.1103/PhysRevE.72.027104.

J. Ebert, B. Kullbach, V. Riediger, and A. Winter. GUPRO: generic understanding of programs – an overview. In *Electronic Notes In Theorethical Computer Science*, volume 72, pages 47–56, 2002. doi: 10.1016/S1571-0661(05)80528-6.

E. Elmacioglu and D. Lee. On six degrees of separation in DBLP-DB and more. *SIGMOD Rec.*, 34 (2):33–40, June 2005. ISSN 0163-5808. doi: 10.1145/1083784.1083791.

D. M. Erceg-Hurn and V. M. Mirosevich. Modern robust statistical methods: an easy way to maximize the accuracy and power of your research. *The American Psychologist*, 63(7):591–601, 2008. doi: 10.1037/0003-066X.63.7.591.

P. Erdős. On the fundamental problem of mathematics. *The American Mathematical Monthly*, 79(2): 149, 1972.

P. Erdős and A. Rényi. On random graphs. I. *Publ. Math. Debrecen*, 6:290–297, 1959.

P. Erdős and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231, 1996.

M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM Comput. Commun. Rev.*, 29:251–262, August 1999. ISSN 0146-4833.

Y. Fan, M. Li, J. Chen, L. Gao, Z. Di, and J. Wu. Network of econophysicists: A weighted network to investigate the development of econophysics. *International Journal of Modern Physics B*, 18 (17n19):2505–2511, 2004. doi: 10.1142/S0217979204025579.

I. Farkas, D. Ábel, G. Palla, and T. Vicsek. Weighted network modules. *New Journal of Physics*, 9 (6):180, 2007.

C. K. Fatt, E. Ujum, and K. Ratnavelu. The structure of collaboration in the Journal of Finance. *Scientometrics*, 85(3):849–860, 2010. ISSN 0138-9130. doi: 10.1007/s11192-010-0254-0.

B. D. Fegley and V. I. Torvik. Has large-scale named-entity network analysis been resting on a flawed assumption? *PLoS ONE*, 8(7):e70299, 2013. doi: 10.1371/journal.pone.0070299.

W. Feller. On the Kolmogorov-Smirnov limit theorems for empirical distributions. *The Annals of Mathematical Statistics*, 19(2):177–189, 1948.

N. E. Fenton. *Software Metrics: A Rigorous Approach.* Chapman & Hall, Ltd., London, UK, UK, 1991. ISBN 0442313551.

A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. A brief survey of automatic methods for author name disambiguation. *SIGMOD Record*, 41(2):15–26, Aug. 2012. ISSN 0163-5808. doi: 10.1145/2350036.2350040.

K. Fischbach, J. Putzke, and D. Schoder. Co-authorship networks in electronic markets research. *Electronic Markets*, 21(1):19–40, 2011. ISSN 1019-6781. doi: 10.1007/s12525-011-0051-5.

G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–71, Mar. 2002. ISSN 0018-9162. doi: 10.1109/2.989932.

M. A. Fortuna, J. A. Bonachela, and S. A. Levin. Evolution of a modular software network. *Proceedings of the National Academy of Sciences*, 108(50):19985–19989, 2011. doi: 10.1073/pnas.1115960108.

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3–5):75 – 174, 2010. ISSN 0370-1573. doi: 10.1016/j.physrep.2009.11.002.

S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

M. Fowler. Reducing coupling. *IEEE Software*, 18:102–104, 2001.

M. Franceschet. Collaboration in computer science: A network science approach. *Journal of the American Society for Information Science and Technology*, 62(10):1992–2012, 2011. ISSN 1532-2890. doi: 10.1002/asi.21614.

L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

L. C. Freeman, S. P. Borgatti, and D. R. White. Centrality in valued graphs: A measure of betweenness based on network flow. *Social Networks*, 13(2):141–154, June 1991. doi: 10.1016/0378-8733(91)90017-N.

Y. Gao, Z. Zheng, and F. Qin. Analysis of linux kernel as a complex network. *Chaos, Solitons & Fractals*, 69(0):246 – 252, 2014. ISSN 0960-0779. doi: 10.1016/j.chaos.2014.10.008.

D. Garlaschelli and M. Loffredo. Patterns of link reciprocity in directed networks. *Phys. Rev. Lett.*, 93:268701, Dec 2004. doi: 10.1103/PhysRevLett.93.268701.

W. Ge, J. Chen, W. Hu, and Y. Qu. Object Link Structure in the Semantic Web. In L. Aroyo, G. Antoniou, E. Hyvönen, A. ten Teije, H. Stuckenschmidt, L. Cabral, and T. Tudorache, editors, *The Semantic Web: Research and Applications*, volume 6089 of *Lecture Notes in Computer Science*, pages 257–271. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-13488-3. doi: 10.1007/978-3-642-13489-0_18.

R. Gil and R. García. Measuring the semantic web. In *Advances in Metadata Research, Proceedings of MTSR'05*, 2006.

E. N. Gilbert. Random graphs. *Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002. ISSN 1091-6490. doi: 10.1073/pnas.122653799.

C. Goffman. And what is your Erdős number? *The American Mathematical Monthly*, 76(7):149, 1969.

S. D. Gollapalli, P. Mitra, and C. L. Giles. Ranking authors in digital libraries. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, JCDL '11, pages 251–254, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0744-4. doi: 10.1145/1998076.1998123.

B. H. Good, Y.-A. de Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, Apr 2010. doi: 10.1103/PhysRevE.81.046106.

S. Goyal, M. J. van der Leij, and J. L. Moraga-Gonzales. Economics: An emerging small world. *Journal of Political Economy*, 114(2):pp. 403–412, 2006. ISSN 00223808.

S. Gregory. An algorithm to find overlapping community structure in networks. In J. Kok, J. Koronacki, R. Lopez de Mantaras, S. Matwin, D. Mladenič, and A. Skowron, editors, *Knowledge Discovery in Databases: PKDD 2007*, volume 4702 of *Lecture Notes in Computer Science*, pages 91–102. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74975-2. doi: 10.1007/978-3-540-74976-9_12.

J. Grossman. The evolution of the mathematical research collaboration graph. *Congressus Numerantium*, pages 201–212, 2002a.

J. Grossman. Patterns of collaboration in mathematical research. *SIAM News*, 35(9):8–9, 2002b.

J. W. Grossman. Paul Erdős: The Master of Collaboration. In R. L. Graham, J. Nešetřil, and S. Butler, editors, *The Mathematics of Paul Erdős II*, pages 489–496. Springer New York, 2013. ISBN 978-1-4614-7253-7. doi: 10.1007/978-1-4614-7254-4_27.

J. W. Grossman and P. D. F. Ion. On a portion of the well known collaboration graph. *Congressus Numerantium*, 108:129–131, 1995.

D. Grove and C. Chambers. A framework for call graph construction algorithms. *ACM Trans. Program. Lang. Syst.*, 23(6):685–746, Nov. 2001. ISSN 0164-0925. doi: 10.1145/506315.506316.

D. Grove, G. DeFouw, J. Dean, and C. Chambers. Call graph construction in object-oriented languages. In *Proceedings of the 12th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '97, pages 108–124, New York, NY, USA, 1997. ACM. ISBN 0-89791-908-4. doi: 10.1145/263698.264352.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5 (2):199–220, June 1993. ISSN 1042-8143. doi: 10.1006/knac.1993.1008.

R. Guns and R. Rousseau. Recommending research collaborations using link prediction and random forest classifiers. *Scientometrics*, pages 1–13, 2014. ISSN 0138-9130. doi: 10.1007/s11192-013-1228-9.

Q. Guo, T. Zhou, J.-G. Liu, W.-J. Bai, B.-H. Wang, and M. Zhao. Growing scale-free small-world networks with tunable assortative coefficient. *Physica A: Statistical Mechanics and its Applications*, 371(2):814 – 822, 2006. ISSN 0378-4371. doi: 10.1016/j.physa.2006.03.055.

M. H. Halstead. *Elements of Software Science (Operating and Programming Systems Series)*. Elsevier Science Inc., New York, NY, USA, 1977. ISBN 0444002057.

A. E. Hassan and R. C. Holt. The small world of software reverse engineering. *2013 20th Working Conference on Reverse Engineering (WCRE)*, 0:278–283, 2004. ISSN 1095-1350. doi: 10.1109/WCRE.2004.37.

S. Henry and D. Kafura. Software structure metrics based on information flow. *IEEE Transactions on Software Engineering*, SE-7(5):510–518, Sept 1981. ISSN 0098-5589. doi: 10.1109/TSE.1981.231113.

M. Hitz and B. Montazeri. Measuring coupling and cohesion in object-oriented systems. In *Proc. International Symposium on Applied Corporate Computing*, pages 25–27, 1995.

P. Holme and B. J. Kim. Growing scale-free networks with tunable clustering. *Phys. Rev. E*, 65: 026107, Jan 2002. doi: 10.1103/PhysRevE.65.026107.

M. Horridge and S. Bechhofer. The owl api: A java api for owl ontologies. *Semantic web*, 2(1):11–21, Jan. 2011. ISSN 1570-0844.

B. Hoser, A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Semantic network analysis of ontologies. In *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications*, ESWC'06, pages 514–529, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34544-2, 978-3-540-34544-2. doi: 10.1007/11762256_38.

H. Hou, H. Kretschmer, and Z. Liu. The structure of scientific collaboration networks in Scientometrics. *Scientometrics*, 75(2):189–202, 2008. ISSN 0138-9130. doi: 10.1007/s11192-007-1771-3.

J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, pages 536–544, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45374-1, 978-3-540-45374-1. doi: 10.1007/11871637_53.

J. Huang, Z. Zhuang, J. Li, and C. L. Giles. Collaboration over time: Characterizing and modeling network evolution. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, pages 107–116, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-927-2. doi: 10.1145/1341531.1341548.

Z. Hui, X. Cai, J.-M. Greneche, and Q. Wang. Structure and collaboration relationship analysis in a scientific collaboration network. *Chinese Science Bulletin*, 56(34):3702–3706, 2011. ISSN 1001-6538. doi: 10.1007/s11434-011-4756-9.

D. Hylland-Wood, D. Carrington, and S. Kaplan. Scale-free nature of Java software package, class and method collaboration graphs. Technical Report TR-MS1286, MIND Laboratory, University of Maryland, College Park, USA, 2006.

M. Ichii, M. Matsushita, and K. Inoue. An exploration of power-law in use-relation of java software systems. In *Proceedings of the 19th Australian Conference on Software Engineering*, ASWEC '08, pages 422–431, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3100-7.

K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, and S. Kusumoto. Ranking significance of software components based on use relations. *IEEE Trans. Softw. Eng.*, 31(3):213–225, Mar. 2005. ISSN 0098-5589. doi: 10.1109/TSE.2005.38.

D. Ivanovic, D. Surla, and M. Rackovic. Journal evaluation based on bibliometric indicators and the cerif data model. *Computer Science and Information Systems*, 9(2):791–811, 2012.

M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, June 1989.

S. Jenkins and S. R. Kirk. Software architecture graphs as complex networks: a novel partitioning scheme to measure stability and evolution. *Information Sciences*, 177:2587–2601, June 2007. ISSN 0020-0255. doi: 10.1016/j.ins.2007.01.021.

F. Johansson, C. Martenson, and P. Svenson. A social network analysis of the information fusion community. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1–8, July 2011.

A. Katifori, C. Halatsis, G. Lepouras, C. Vassilakis, and E. Giannopoulou. Ontology visualization methods – a survey. *ACM Computing Surveys*, 39(4), Nov. 2007. ISSN 0360-0300. doi: 10.1145/1287620.1287621.

H. M. Kienle and H. A. Müller. Rigi - an environment for software reverse engineering, exploration, visualization, and redocumentation. *Science of Computer Programming*, 75(4):247–263, 2010. ISSN 0167-6423. doi: 10.1016/j.scico.2009.10.007.

J. Kim, H. Kim, and D. Jana. The impact of name ambiguity on properties of coauthorship networks. *Journal of Information Science Theory and Practice*, 2(2), 2014. doi: 10.1633/JISTaP.2014.2.2.1.

J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models, and methods. In T. Asano, H. Imai, D. Lee, S.-i. Nakano, and T. Tokuyama, editors, *Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-66200-6. doi: 10.1007/3-540-48686-0_1.

G. A. Kohring. Complex dependencies in large software systems. *Advances in Complex Systems*, 12 (06):565–581, 2009. doi: 10.1142/S0219525909002362.

J. Kolek, G. Rakić, and M. Savić. Two-dimensional extensibility of SSQSA framework. In *Proceedings of the 2nd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA)*, pages 35–43, 2013.

P. L. Krapivsky and S. Redner. Network growth by copying. *Phys. Rev. E*, 71:036118, Mar 2005. doi: 10.1103/PhysRevE.71.036118.

P. L. Krapivsky, S. Redner, and F. Leyvraz. Connectivity of growing random networks. *Phys. Rev. Lett.*, 85:4629–4632, Nov 2000. doi: 10.1103/PhysRevLett.85.4629.

L. Kronegger, F. Mali, A. Ferligoj, and P. Doreian. Collaboration structures in Slovenian scientific communities. *Scientometrics*, 90(2):631–647, 2012. ISSN 0138-9130. doi: 10.1007/s11192-011-0493-8.

T. S. Kuhn. *The structure of scientific revolutions*. University of Chicago Press, Chicago, 1970.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *Proceedings of the 25th VLDB Conference*, pages 639–650, 1999.

R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 57–65, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0850-2.

N. Labelle and E. Wallingford. Inter-package dependency networks in open-source software. In *Proceedings of the 6th International Conference on Complex Systems (ICCS), paper no. 226*, 2006.

M. Lanza and S. Ducasse. Polymetric views - a lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, 2003. ISSN 0098-5589. doi: 10.1109/TSE.2003.1232284.

R. Lara-Cabrera, C. Cotta, and A. Fernández-Leiva. An analysis of the structure and evolution of the scientific collaboration network of computer intelligence in games. *Physica A: Statistical Mechanics and its Applications*, 395:523 – 536, 2014. ISSN 0378-4371. doi: 10.1016/j.physa.2013.10.036.

Y. S. Lee, B. S. Liang, S. F. Wu, and F. J. Wang. Measuring the coupling and cohesion of an object-oriented program based on information flow. In *Proceedings of International Conference on Software Quality*, 1995.

E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, Mar 2008. doi: 10.1103/PhysRevLett.100.118703.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081893.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007. ISSN 1556-4681. doi: 10.1145/1217299.1217301.

J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009. doi: 10.1080/15427951.2009.10129177.

J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 631–640, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8.

M. Ley. DBLP: Some lessons learned. *Proc. VLDB Endow.*, 2(2):1493–1500, Aug. 2009. ISSN 2150-8097. doi: 10.14778/1687553.1687577.

C. Li and P. K. Maini. An evolving network model with community structure. *Journal of Physics A: Mathematical and General*, 38(45):9741, 2005.

L. Li, D. Alderson, J. C. Doyle, and W. Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005. doi: 10.1080/15427951.2005.10129111.

L. Li, X. Li, C. Cheng, C. Chen, G. Ke, D. Zeng, and W. Scherer. Research collaboration and ITS topic evolution: 10 years at T-ITS. *Intelligent Transportation Systems, IEEE Transactions on*, 11 (3):517–523, Sept 2010. ISSN 1524-9050. doi: 10.1109/TITS.2010.2059070.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM. doi: 10.1145/956863.956972.

D. Lindsey. Production and citation measures in the sociology of science: The problem of multiple authorship. *Social Studies of Science*, 10(2):145–162, 1980.

X. Liu, J. Bollen, M. L. Nelson, and H. Van de Sompel. Co-authorship networks in the digital library research community. *Inf. Process. Manage.*, 41(6):1462–1480, Dec 2005. ISSN 0306-4573. doi: 10.1016/j.ipm.2005.03.012.

A. J. Lotka. The frequency distribution of scientific production. *Journal of Washington Academy of Science*, 16:317–323, 1926.

P. Louridas, D. Spinellis, and V. Vlachos. Power laws in software. *ACM Trans. Softw. Eng. Methodol.*, 18(1):2:1–2:26, Oct. 2008. ISSN 1049-331X.

H. Lu and Y. Feng. A measure of authors' centrality in co-authorship networks based on the distribution of collaborative relationships. *Scientometrics*, 81(2):499–511, 2009. ISSN 0138-9130. doi: 10.1007/s11192-008-2173-x.

F. Luccio and M. Sami. On the decomposition of networks in minimally interconnected subnetworks. *IEEE Transactions on Circuit Theory*, 16(2):184–188, 1969. ISSN 0018-9324.

A. D. Lucia, V. Deufemia, C. Gravino, and M. Risi. Design pattern recovery through visual language parsing and source code analysis. *Journal of Systems and Software*, 82(7):1177–1193, July 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2009.02.012.

L. Luthi, M. Tomassini, M. Giacobini, and W. B. Langdon. The genetic programming collaboration network and its communities. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1643–1650, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-697-4. doi: 10.1145/1276958.1277284.

B. Lužar, Z. Levnajić, J. Povh, and M. Perc. Community Structure and the Evolution of Interdisciplinarity in Slovenia's Scientific Collaboration Network. *PLoS ONE*, 9(4):e94429, 2014. doi: 10.1371/journal.pone.0094429.

J. Ma and H. Chen. Complex Network Analysis on TCMLS Sub-Ontologies. In *Third International Conference on Semantics, Knowledge and Grid*, pages 551–553, Oct 2007. doi: 10.1109/SKG.2007.25.

Y. Ma, H. Wu, X. Ma, B. Jin, T. Huang, and J. Wei. Stable cohesion metrics for evolving ontologies. *Journal of Software Maintenance and Evolution*, 23(5):343–359, Aug. 2011. ISSN 1532-060X. doi: 10.1002/smr.509.

T. Maillart, D. Sornette, S. Spaeth, and G. von Krogh. Empirical tests of zipf's law mechanism in open source linux distribution. *Phys. Rev. Lett.*, 101:218701, Nov 2008. doi: 10.1103/PhysRevLett.101.218701.

S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, and E. R. Gansner. Using automatic clustering to produce high-level system organizations of source code. In *Proceedings of the 6th International Workshop on Program Comprehension*, IWPC '98, pages 45–52, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 0-8186-8560-3. doi: 10.1109/WPC.1998.693283.

H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947. doi: 10.2307/2236101.

O. Maqbool and H. Babri. Hierarchical clustering for software architecture recovery. *IEEE Transactions on Software Engineering*, 33(11):759–780, 2007. ISSN 0098-5589. doi: 10.1109/TSE.2007.70732.

T. Martin, B. Ball, B. Karrer, and M. E. J. Newman. Coauthorship and citation patterns in the Physical Review. *Physical Review E*, 88:012814, Jul 2013. doi: 10.1103/PhysRevE.88.012814.

B. McBride. Jena: A semantic web toolkit. *IEEE Internet Computing*, 6(6):55–59, Nov. 2002. ISSN 1089-7801. doi: 10.1109/MIC.2002.1067737.

T. J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, 2(4):308–320, July 1976. ISSN 0098-5589. doi: 10.1109/TSE.1976.233837.

H. Melton and E. Tempero. Identifying refactoring opportunities by identifying dependency cycles. In *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ACSC '06, pages 35–41, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-30-9.

H. Melton and E. Tempero. An empirical study of cycles among classes in java. *Empirical Software Engineering*, 12(4):389–415, Aug. 2007. ISSN 1382-3256. doi: 10.1007/s10664-006-9033-1.

J. P. Mena-Chalco, L. A. Digiampietri, F. M. Lopes, and R. M. Cesar. Brazilian bibliometric coauthorship networks. *Journal of the Association for Information Science and Technology*, 65(7):1424–1445, 2014. ISSN 2330-1643. doi: 10.1002/asi.23010.

Z. Mijajlović, Z. Ognjanovic, and A. Pejovic. Digitization of mathematical editions in Serbia. *Mathematics in Computer Science*, 3(3):251–263, 2010. ISSN 1661-8270. doi: 10.1007/s11786-010-0021-x.

S. Milojević. Accuracy of simple, initials-based methods for author name disambiguation. *Journal of Informetrics*, 7(4):767 – 773, 2013. ISSN 1751-1577. doi: 10.1016/j.joi.2013.06.006.

D. Mimno and A. McCallum. Mining a digital library for influential authors. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '07, pages 105–106, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-644-8. doi: 10.1145/1255175.1255196.

B. S. Mitchell and S. Mancoridis. On the automatic modularization of software systems using the Bunch tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, 2006. ISSN 0098-5589. doi: 10.1109/TSE.2006.31.

J. Moody. The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. *American Sociological Review*, 69(2):213–238, 2004.

G. C. Murphy and D. Notkin. Lightweight lexical source model extraction. *ACM Trans. Softw. Eng. Methodol.*, 5(3):262–292, July 1996. ISSN 1049-331X. doi: 10.1145/234426.234441.

G. C. Murphy, D. Notkin, and K. Sullivan. Software reflexion models: bridging the gap between source and high-level models. In *Proceedings of the 3rd ACM SIGSOFT symposium on Foundations of software engineering*, SIGSOFT '95, pages 18–28, New York, NY, USA, 1995. ACM. ISBN 0-89791-716-2. doi: 10.1145/222124.222136.

G. C. Murphy, D. Notkin, W. G. Griswold, and E. S. Lan. An empirical study of static call graph extractors. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(2):158–191, 1998. ISSN 1049-331X. doi: 10.1145/279310.279314.

C. R. Myers. Software systems as complex networks: structure, function, and evolvability of software collaboration graphs. *Phys. Rev. E*, 68(4):046116, Oct 2003. doi: 10.1103/PhysRevE.68.046116.

M. A. Nascimento, J. Sander, and J. Pound. Analysis of SIGMOD's co-authorship graph. *SIGMOD Rec.*, 32(3):8–10, Sept. 2003. ISSN 0163-5808. doi: 10.1145/945721.945722.

M. Newman. Clustering and preferential attachment in growing networks. *Physical Review E*, 64(2): 025102, 2001a.

M. Newman. Coauthorship networks and patterns of scientific collaboration. *Proceedings of the National Academy of Sciences*, 101(1):5200 – 5205, 4 2004a.

M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010. ISBN 0199206651, 9780199206650.

M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001b. doi: 10.1073/pnas.98.2.404.

M. E. J. Newman. Scientific collaboration networks I: network construction and fundamental results. *Phys. Rev. E*, 64:016131, Jun 2001c. doi: 10.1103/PhysRevE.64.016131.

M. E. J. Newman. Scientific collaboration networks II: shortest paths, weighted networks, and centrality. *Phys. Rev. E*, 64:016132, Jun 2001d. doi: 10.1103/PhysRevE.64.016132.

M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, Oct 2002. doi: 10.1103/PhysRevLett.89.208701.

M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67:026126, Feb 2003a. doi: 10.1103/PhysRevE.67.026126.

M. E. J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45:167–256, 2003b. doi: 10.1137/S003614450342480.

M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69: 066133, Jun 2004b. doi: 10.1103/PhysRevE.69.066133.

M. E. J. Newman. Who is the best connected scientist? A study of scientific coauthorship networks. In E. Ben-Naim, H. Frauenfelder, and Z. Toroczkai, editors, *Complex Networks*, volume 650 of *Lecture Notes in Physics*, pages 337–370. Springer Berlin Heidelberg, 2004c. ISBN 978-3-540-22354-2. doi: 10.1007/978-3-540-44485-5_16.

M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004d. doi: 10.1103/PhysRevE.70.056131.

M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemp Phys*, 46:323, 2005.

M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004. doi: 10.1103/PhysRevE.69.026113.

V. H. Nguyen and L. M. S. Tran. Predicting vulnerable software components with dependency graphs. In *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, MetriSec '10, pages 3:1–3:8, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0340-8. doi: 10.1145/1853919.1853923.

X. Ochoa, G. Méndez, and E. Duval. Who we are: Analysis of 10 years of the ED-MEDIA conference. In *In G. Siemens and C. Fulford (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2009*, pages 189–200. AACE, June 2009.

S. Oh, H. Y. Yeom, and J. Ahn. Cohesion and coupling metrics for ontology modules. *Information Technology and Management*, 12(2):81–96, June 2011. ISSN 1385-951X. doi: 10.1007/s10799-011-0094-5.

R. Oliveto, M. Gethers, G. Bavota, D. Poshyvanyk, and A. De Lucia. Identifying method friendships to remove the feature envy bad smell. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 820–823, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0445-0. doi: 10.1145/1985793.1985913.

A. Orme, H. Tao, and L. Etzkorn. Coupling metrics for ontology-based system. *IEEE Software*, 23 (2):102–108, 2006. doi: 10.1109/MS.2006.46.

E. Otte and R. Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of Information Science*, 28(6):441–453, 2002. doi: 10.1177/016555150202800601.

T. D. Oyetoyan, D. S. Cruzes, and R. Conradi. A study of cyclic dependencies on defect profile of software components. *Journal of Systems and Software*, 86(12):3162 – 3182, 2013. ISSN 0164-1212. doi: 10.1016/j.jss.2013.07.039.

R. K. Pan and J. Saramäki. The strength of strong ties in scientific collaboration networks. *EPL (Europhysics Letters)*, 97(1):18007, 2012.

M. L. Pao. An empirical examination of Lotka's law. *Journal of the American Society for Information Science*, 37(1):26–33, 1986. ISSN 1097-4571. doi: 10.1002/asi.4630370105.

T. J. Parr and R. W. Quong. ANTLR: a predicated-LL(k) parser generator. *Software: Practice and Experience*, 25(7):789–810, 1995. ISSN 1097-024X. doi: 10.1002/spe.4380250705.

R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, Apr 2001. doi: 10.1103/PhysRevLett.86.3200.

R. Pastor-Satorras, A. Vázquez, and A. Vespignani. Dynamical and correlation properties of the internet. *Phys. Rev. Lett.*, 87:258701, Nov 2001. doi: 10.1103/PhysRevLett.87.258701.

P. Paymal, R. Patil, S. Bhomwick, and H. Siy. Empirical study of software evolution using community detection. 2011. URL http://cs.unomaha.edu/~bhowmick/STARyNet/papers/techshort.pdf.

M. Perc. Growth and structure of Slovenia's scientific collaboration network. *J. Informetrics*, 4(4): 475–482, 2010. doi: http://dx.doi.org/10.1016/j.joi.2010.04.003.

Peter F. Patel-Schneider and Bors Motik (Editors). OWL 2 Web Ontology Language Mapping to RDF Graphs (Second Edition). http://www.w3.org/TR/owl2-mapping-to-rdf/. Accessed: 2014-12-31.

M. C. Pham, M. Derntl, and R. Klamma. Development patterns of scientific communities in technology enhanced learning. *Educational Technology & Society*, 15(3):323–335, 2012.

P. Pollner, G. Palla, and T. Vicsek. Preferential attachment of communities: The same principle, but a higher level. *EPL (Europhysics Letters)*, 73(3):478, 2006. doi: 10.1209/epl/i2005-10414-6.

P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

A. Potanin, J. Noble, M. Frean, and R. Biddle. Scale-free geometry in OO programs. *Commun. ACM*, 48:99–103, May 2005. ISSN 0001-0782. doi: 10.1145/1060710.1060716.

D. Puppin and F. Silvestri. The social network of Java classes. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1409–1413, New York, NY, USA, 2006. ACM. ISBN 1-59593-108-2. doi: 10.1145/1141277.1141605.

F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658–2663, 2004. doi: 10.1073/pnas.0400054101.

M. Radovanović, J. Ferlež, D. Mladenović, M. Grobelnik, and M. Ivanović. Mining and visualizing scientific publication data from Vojvodina. *Novi Sad Journal of Mathematics*, 37(2):161–180, 2007.

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76:036106, 2007.

G. Rakić and Z. Budimac. Introducing enriched concrete syntax trees. In *Proceedings of the 14th International Multiconference on Information Society (IS), Collaboration, Software And Services In Information Society (CSS)*, pages 211–214, 2011a.

G. Rakić and Z. Budimac. SMIILE prototype. In *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (ICNAAM), Symposium on Computer Languages, Implementations and Tools (SCLIT)*, pages 544–549, 2011b. ISBN 978-0-7354-0956-9. doi: 10.1063/1.3636867.

G. Rakić, Z. Budimac, and M. Savić. Language independent framework for static code analysis. In *Proceedings of the 6th Balkan Conference in Informatics*, BCI '13, pages 236–243, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1851-8. doi: 10.1145/2490257.2490273.

J. J. Ramasco, S. N. Dorogovtsev, and R. Pastor-Satorras. Self-organization of collaboration networks. *Phys. Rev. E*, 70:036106, Sep 2004. doi: 10.1103/PhysRevE.70.036106.

R. G. Raskin and M. J. Pan. Knowledge Representation in the Semantic Web for Earth and Environmental Terminology (SWEET). *Computers & Geosciences*, 31(9):1119–1125, Nov. 2005. ISSN 0098-3004. doi: 10.1016/j.cageo.2004.12.004.

A. Raza, G. Vogel, and E. Plödereder. Bauhaus: a tool suite for program analysis and reverse engineering. In *Proceedings of the 11th Ada-Europe international conference on Reliable Software Technologies*, Ada-Europe'06, pages 71–82, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-34663-5, 978-3-540-34663-0. doi: 10.1007/11767077_6.

S. Redner. Citation Statistics from 110 Years of Physical Review. *Physics Today*, 58(6):49–54, 2005. doi: 10.1063/1.1996475.

W. Reinhardt, C. Meier, H. Drachsler, and P. Sloep. Analyzing 5 years of EC-TEL proceedings. In C. D. Kloos, D. Gillet, R. M. Crespo García, F. Wild, and M. Wolpers, editors, *Towards Ubiquitous Learning*, volume 6964 of *Lecture Notes in Computer Science*, pages 531–536. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23984-7. doi: 10.1007/978-3-642-23985-4_51.

P. Reuther, B. Walter, M. Ley, A. Weber, and S. Klink. Managing the quality of person names in dblp. In J. Gonzalo, C. Thanos, M. Verdejo, and R. Carrasco, editors, *Research and Advanced Technology for Digital Libraries*, volume 4172 of *Lecture Notes in Computer Science*, pages 508–511. Springer Berlin Heidelberg, 2006. ISBN 978-3-540-44636-1. doi: 10.1007/11863878_55.

M. Risi and G. Scanniello. Metricattitude: a visualization tool for the reverse engineering of object oriented software. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 449–456, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254643.

M. A. Rodriguez and J. Bollen. An algorithm to determine peer-reviewers. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 319–328, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-991-3.

M. A. Rodriguez, J. Bollen, and H. Van de Sompel. The convergence of digital libraries and the peer-review process. *Journal of Information Science*, 32(2):149, 2006.

M. Rosvall and C. T. Bergstrom. Maps of information flow reveal community structure in complex networks. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 105, pages 1118–1123, 2007.

M. Savić and M. Ivanović. Graph clustering evaluation metrics as software metrics. In *Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement and Applications (SQAMIA 2014), Lovran, Croatia, September 19-22, 2014*, pages 81–89, 2014.

M. Savić, G. Rakić, Z. Budimac, and M. Ivanović. Extractor of software networks from enriched concrete syntax trees. *AIP Conference Proceedings*, 1479(1):486–489, 2012. doi: 10.1063/1.4756172.

M. Savić, Z. Budimac, G. Rakić, M. Ivanović, and M. Heričko. SSQSA ontology metrics front-end. In *Proceedings of the 2nd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, Novi Sad, Serbia, September 15-17, 2013*, pages 95–101, 2013. URL http://ceur-ws.org/Vol-1053/sqamia2013paper12.pdf.

M. Savić, G. Rakić, Z. Budimac, and M. Ivanović. A language-independent approach to the extraction of dependencies between source code entities. *Inf. Softw. Technol.*, 56(10):1268–1288, Oct. 2014. ISSN 0950-5849. doi: 10.1016/j.infsof.2014.04.011.

M. Savić, M. Ivanović, M. Radovanović, Z. Ognjanović, A. Pejović, and T. Jakšić Krüger. Exploratory analysis of communities in co-authorship networks: A case study. In A. M. Bogdanova and D. Gjorgjevikj, editors, *ICT Innovations 2014*, volume 311 of *Advances in Intelligent Systems and Computing*, pages 55–64. Springer International Publishing, 2015. ISBN 978-3-319-09878-4. doi: 10.1007/978-3-319-09879-1_6.

M. Savić, M. Ivanović, M. Radovanović, Z. Ognjanović, A. Pejović, and T. Jakšić Krüger. The structure and evolution of scientific collaboration in serbian mathematical journals. *Scientometrics*, 101(3):1805–1830, 2014. ISSN 0138-9130. doi: 10.1007/s11192-014-1295-6.

G. Scanniello and A. Marcus. Clustering support for static concept location in source code. In *Proceedings of the 19th International Conference on Program Comprehension (ICPC 2011)*, pages 1–10, 2011. doi: 10.1109/ICPC.2011.13.

G. Scanniello, A. D'Amico, C. D'Amico, and T. D'Amico. Using the Kleinberg algorithm and vector space model for software system clustering. In *18th International Conference on Program Comprehension (ICPC 2010)*, pages 180–189, 2010. doi: 10.1109/ICPC.2010.17.

R. W. Schwanke. An intelligent tool for re-engineering software modularity. In *Proceedings of the 13th international conference on Software engineering*, ICSE '91, pages 83–92, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press. ISBN 0-89791-391-4. doi: 10.1109/ICSE.1991.130626.

N. Shadbolt, T. Berners-Lee, and W. Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006. ISSN 1541-1672. doi: 10.1109/MIS.2006.62.

Q. Shi, B. Xu, X. Xu, Y. Xiao, W. Wang, and H. Wang. Diversity of social ties in scientific collaboration networks. *Physica A: Statistical Mechanics and its Applications*, 390(23–24):4627 – 4635, 2011. ISSN 0378-4371. doi: 10.1016/j.physa.2011.06.072.

M. Shtern and V. Tzerpos. Clustering methodologies for software engineering. *Advances in Software Engineering*, 2012:1:1–1:18, 2012. ISSN 1687-8655. doi: 10.1155/2012/792024.

M. A. Sicilia, D. Rodríguez, E. García-Barriocanal, and S. Sánchez-Alonso. Empirical findings on ontology metrics. *Expert Syst. Appl.*, 39(8):6706–6711, June 2012. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.11.094.

N. R. Smalheiser and V. I. Torvik. Author name disambiguation. *Annual Review of Information Science and Technology*, 43(1):1–43, 2009. ISSN 1550-8382. doi: 10.1002/aris.2009.1440430113.

A. F. Smeaton, G. Keogh, C. Gurrin, K. McDonald, and T. Sødring. Analysis of papers from twenty-five years of SIGIR conferences: What have we been doing for the last quarter of a century? *SIGIR Forum*, 37(1):49–53, Apr. 2003. ISSN 0163-5840. doi: 10.1145/945546.945550.

C. Staudt, A. Schumm, H. Meyerhenke, R. Görke, and D. Wagner. Static and dynamic aspects of scientific collaboration networks. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26-29 August 2012*, pages 522–526, 2012. doi: 10.1109/ASONAM.2012.90.

H. Stuckenschmidt and A. Schlicht. Structure-based partitioning of large ontologies. In H. Stuckenschmidt, C. Parent, and S. Spaccapietra, editors, *Modular Ontologies*, pages 187–210. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-01906-7. doi: 10.1007/978-3-642-01907-4_9.

J. Sudeikat and W. Renz. On complex networks in software: How agent–orientation effects software structures. In H.-D. Burkhard, G. Lindemann, R. Verbrugge, and L. Varga, editors, *Multi-Agent Systems and Applications V*, volume 4696 of *Lecture Notes in Computer Science*, pages 215–224. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-75253-0. doi: 10.1007/978-3-540-75254-7_22.

V. Sundaresan, L. Hendren, C. Razafimahefa, R. Vallée-Rai, P. Lam, E. Gagnon, and C. Godin. Practical virtual method call resolution for Java. In *Proceedings of the 15th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '00, pages 264–280, New York, NY, USA, 2000. ACM. ISBN 1-58113-200-X. doi: 10.1145/353171.353189.

S. Tartir, I. B. Arpinar, M. Moore, A. P. Sheth, and B. Aleman-Meza. OntoQA: Metric-based ontology quality analysis. In *IEEE Workshop on Knowledge Acquisition from Distributed, Autonomous, Semantically Heterogeneous Data and Knowledge Sources*, 2005.

C. Taube-Schock, R. Walker, and I. Witten. Can we avoid high coupling? In M. Mezini, editor, *ECOOP 2011 – Object-Oriented Programming*, volume 6813 of *Lecture Notes in Computer Science*, pages 204–228. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-22654-0. doi: 10.1007/978-3-642-22655-7_10.

A. Telea, H. Hoogendorp, O. Ersoy, and D. Reniers. Extraction and visualization of call dependencies for large c/c++ code bases: A comparative study. In *5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2009)*, pages 81–88, Sept 2009. doi: 10.1109/VISSOF.2009.5336419.

B. TePaske-King and N. Richert. The identification of authors in the Mathematical Reviews database. *Issues in Science and Technology Librarianship*, (31), 2001. doi: 10.5062/F4KH0K9M.

Y. Theoharis, G. Georgakopoulos, and V. Christophides. On the synthetic generation of semantic web schemas. In V. Christophides, M. Collard, and C. Gutierrez, editors, *Semantic Web, Ontologies and Databases*, volume 5005 of *Lecture Notes in Computer Science*, pages 98–116. Springer Berlin Heidelberg, 2008a. ISBN 978-3-540-70959-6. doi: 10.1007/978-3-540-70960-2_6.

Y. Theoharis, Y. Tzitzikas, D. Kotzinos, and V. Christophides. On graph features of semantic web schemas. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):692–702, May 2008b. ISSN 1041-4347. doi: 10.1109/TKDE.2007.190735.

M. Tomasini and L. Luthi. Empirical analysis of the evolution of a scientific collaboration network. *Physica A: Statistical Mechanics and its Applications*, 385(2):750 – 764, 2007. ISSN 0378-4371. doi: 10.1016/j.physa.2007.07.028.

M. Tomassini, L. Luthi, M. Giacobini, and W. Langdon. The structure of the genetic programming collaboration network. *Genetic Programming and Evolvable Machines*, 8(1):97–103, 2007. ISSN 1389-2576. doi: 10.1007/s10710-006-9018-2.

V. I. Torvik and N. R. Smalheiser. Author name disambiguation in MEDLINE. *ACM Trans. Knowl. Discov. Data*, 3(3):11:1–11:29, July 2009. ISSN 1556-4681. doi: 10.1145/1552303.1552304.

V. I. Torvik, M. Weeber, D. R. Swanson, and N. R. Smalheiser. A probabilistic similarity metric for Medline records: A model for author name disambiguation. *Journal of the American Society for Information Science and Technology*, 56(2):140–158, Jan. 2005. ISSN 1532-2882. doi: 10.1002/asi.v56:2.

A. Tosun, B. Turhan, and A. Bener. Validation of network measures as indicators of defective modules in software systems. In *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, PROMISE '09, pages 5:1–5:9, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-634-2. doi: 10.1145/1540438.1540446.

S. Uddin, L. Hossain, A. Abbasi, and K. Rasmussen. Trend and efficiency analysis of co-authorship network. *Scientometrics*, 90(2):687–699, 2012. ISSN 0138-9130. doi: 10.1007/s11192-011-0511-x.

S. Uddin, L. Hossain, and K. Rasmussen. Network effects on scientific collaborations. *PLoS ONE*, 8 (2):e57546, 02 2013. doi: 10.1371/journal.pone.0057546.

S. Valverde and V. Solé. Hierarchical small worlds in software architecure. *Dyn. Contin. Discret. Impuls. Syst. Ser. B: Appl. Algorithms*, 14(S6):305–315, 2007.

S. Valverde, R. F. Cancho, and R. V. Solé. Scale-free networks from optimal design. *EPL (Europhysics Letters)*, 60(4):512–517, 2002. doi: 10.1209/epl/i2002-00248-2.

R. T. Vidgen, S. Henneberg, and P. Naudé. What sort of community is the European Conference on Information Systems? A social network analysis 1993-2005. *European Journal of Information Systems*, 16(1):5–19, 2007. doi: 10.1057/palgrave.ejis.3000661.

H. Voos. Lotka and information science. *Journal of the American Society for Information Science*, 25(4):270–272, 1974. ISSN 1097-4571. doi: 10.1002/asi.4630250410.

D. Vrandečić and Y. Sure. How to design better ontology metrics. In *Proceedings of the 4th European Conference on The Semantic Web: Research and Applications*, ESWC '07, pages 311–325, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-72666-1. doi: 10.1007/978-3-540-72667-8_23.

L. Šubelj and M. Bajec. Software systems through complex networks science: Review, analysis and applications. In *Proceedings of the First International Workshop on Software Mining*, SoftwareMining '12, pages 9–16, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1560-9. doi: 10.1145/2384416.2384418.

R. Žontar and M. Heričko. Adoption of object-oriented software metrics for ontology evaluation. In *Proceedings of the Fifth Balkan Conference in Informatics*, BCI '12, pages 298–301, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1240-0. doi: 10.1145/2371316.2371383.

M. L. Wallace, V. Larivière, and Y. Gingras. A Small World of Citations? The Influence of Collaboration Networks on Citation Practices. *PLoS ONE*, 7(3):e33339, 2012. doi: 10.1371/journal.pone.0033339.

L. Wang, Z. Wang, C. Yang, L. Zhang, and Q. Ye. Linux kernels as complex networks: a novel method to study evolution. *IEEE International Conference on Software Maintenance (ICSM 2009)*, pages 41–50, 2009. ISSN 1063-6773. doi: 10.1109/ICSM.2009.5306348.

L. Wang, P. Yu, Z. Wang, C. Yang, and Q. Ye. On the evolution of linux kernels: a complex network perspective. *Journal of Software: Evolution and Process*, 25(5):439–458, 2013. ISSN 2047-7481. doi: 10.1002/smr.1550.

D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684): 409–10, 1998a.

D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998b.

H. Wen, R. M. D'Souza, Z. M. Saul, and V. Filkov. Evolution of apache open source software. In N. Ganguly, A. Deutsch, and A. Mukherjee, editors, *Dynamics On and Of Complex Networks*, Modeling and Simulation in Science, Engineering and Technology, pages 199–215. Birkhäuser Boston, 2009a. ISBN 978-0-8176-4750-6. doi: 10.1007/978-0-8176-4751-3_12.

L. Wen, R. G. Dromey, and D. Kirk. Software engineering and scale-free networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39:845–854, August 2009b. ISSN 1083-4419. doi: 10.1109/TSMCB.2009.2020206.

R. Wheeldon and S. Counsell. Power law distributions in class relationships. In *Proceedings of the Third IEEE International Workshop on Source Code Analysis and Manipulation*, pages 45–54, 2003. doi: 10.1109/SCAM.2003.1238030.

W. E. Winkler. Overview of record linkage and current research directions. Technical Report RR2006/02, US Bureau of the Census, 2006.

J. Wu, A. E. Hassan, and R. C. Holt. Comparison of clustering algorithms in the context of software evolution. In *Proceedings of the 21st IEEE International Conference on Software Maintenance*, ICSM '05, pages 525–535, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2368-4. doi: 10.1109/ICSM.2005.31.

J. J. Xu and M. Chau. The social identity of IS: analyzing the collaboration network of the ICIS conferences (1980-2005). In *Proceedings of the International Conference on Information Systems, ICIS 2006, Milwaukee, Wisconsin, USA, December 10-13, 2006*, page 39, 2006.

E. Yan and Y. Ding. Applying centrality measures to impact analysis: A coauthorship network analysis. *Journal of the American Society for Information Science and Technology*, 60(10):2107–2118, 2009. ISSN 1532-2890. doi: 10.1002/asi.21128.

E. Yan and R. Guns. Predicting and recommending collaborations: An author-, institution-, and country-level analysis. *Journal of Informetrics*, 8(2):295 – 309, 2014. ISSN 1751-1577. doi: 10.1016/j.joi.2014.01.008.

E. Yan, Y. Ding, and Q. Zhu. Mapping library and information science in China: A coauthorship network analysis. *Scientometrics*, 83(1):115–131, 2010. ISSN 0138-9130. doi: 10.1007/s11192-009-0027-9.

J. Yang and J. Leskovec. Structure and overlaps of ground-truth communities in networks. *ACM Trans. Intell. Syst. Technol.*, 5(2):26:1–26:35, Apr. 2014. ISSN 2157-6904. doi: 10.1145/2594454.

H. Yao, A. M. Orme, and L. Etzkorn. Cohesion Metrics for Ontology Design and Application. *Journal of Computer Science*, 1(1):107–113, 2005.

E. Yourdon and L. L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1979. ISBN 0138544719.

P. Yuan, H. Jin, K. Deng, and Q. Chen. Analyzing software component graphs of grid middleware: Hint to performance improvement. In *Proceedings of the 8th Internationsl Conference on Algorithms and Architectures for Parallel Processing (ICA3PP)*, pages 305–315, 2008.

L. Zhai, X. Li, X. Yan, and W. Fan. Evolutionary analysis of collaboration networks in the field of information systems. *Scientometrics*, pages 1–21, 2014. ISSN 0138-9130. doi: 10.1007/s11192-014-1360-1.

H. Zhang. The Scale-free Nature of Semantic Web Ontology. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 1047–1048, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: 10.1145/1367497.1367649.

H. Zhang, Y.-F. Li, and H. B. K. Tan. Measuring design complexity of semantic web ontologies. *Journal of Systems and Software*, 83(5):803–814, May 2010. ISSN 0164-1212. doi: 10.1016/j.jss.2009.11.735.

X. Zhang, G. Cheng, and Y. Qu. Ontology summarization based on rdf sentence graph. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 707–716, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242668.

T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *Proceedings of the 30th International Conference on Software Engineering*, ICSE '08, pages 531–540, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-079-1. doi: 10.1145/1368088.1368161.

A. Çavuşoğlu and İlker Türker. Scientific collaboration network of Turkey. *Chaos, Solitons and Fractals*, 57(0):9 – 18, 2013. ISSN 0960-0779. doi: 10.1016/j.chaos.2013.07.022.

A. Çavuşoğlu and İlker Türker. Patterns of collaboration in four scientific disciplines of the Turkish collaboration network. *Physica A: Statistical Mechanics and its Applications*, 413(0):220 – 229, 2014. ISSN 0378-4371. doi: 10.1016/j.physa.2014.06.069.

Sažetak

Skoro svaki kompleksan sistem se može predstaviti mrežom koja opisuje interakcije između entiteta od kojih je sistem komponovan. Razumevanje, kontrolisanje i unapređivanje kompleksnih sistema suštinski podrazumeva da smo u stanju da kvantifikujemo, okarakterišemo i pojmimo strukturu i evoluciju njihovih mrežnih reprezentacija. Kompleksne mreže su svuda oko nas: srećemo ih u socijalnim sistemima (npr. onlajn socijalne mreže i mreže saradnje), biološkim sistemima (npr. mreže proteinskih interackcija i neuronske mreže), tehnološkim sistema (npr. mreže električnih vodova, komunikacione mreže i WWW), te konceptualnim sistemima (konceptualne mape, lingvističke mreže). Fokus ove disertacije je na kompleksnim mrežama iz tri domena: (1) mreže ekstrahovane iz izvornog koda računarskih programa koje reprezentuju dizajn softverskih sistema, (2) mreže ekstrahovane iz ontologija semantičkog web-a koje opisuju strukturu deljenog znanja pogodnog za višekratnu upotrebu, i (3) mreže ekstrahovane iz bibliografskih zapisa koje opisuju saradnju istraživača. U okviru disertacije predložene su nove metode za ekstrakciju mreža iz pomenutih domena. Drugo, na nekoliko studija slučaja ilustrovani su benefiti mrežno orjentisane analize konkretnih sistema iz domena obuhvaćenih disertacijom. U poređenju sa prethodnim relevantim istraživanjima, analize prezentovane u disertaciji nisu čisto topološke, nego kombinuju tehnike i metrike razvijene u okviru teorije kompleksnih mreža sa metrikama iz konkretnog domena.

Prvi originalni doprinos disertacije je SNEIPL – proširiv, jezički nezavisan ekstraktor softverskih mreža baziran na jezički nezavisnoj *enriched Concrete Syntax Tree* (eCST) reprezentaciji izvornog koda. Mogućnosti SNEIPL-a su demonstrirane na softverskim sistema pisanim u različitim programskim jezicima koji pripadaju različitim programskim paradigmama. Fokusirajući se na softverske sisteme pisane u programskom jeziku Java pokazali smo da su mreže ekstrahovane koristeći SNEIPL jako slične onima koje se formiraju korišćenjem jezički zavisnog ekstraktora, te znatno preciznije u odnosu na mreže dobijene primenom jezički nezavisnog pristupa baziranog na fazi parsiranju.

Drugi doprinos disertacije je ONGRAM, ekstraktor ontoloških mreža čiji su čvorovi obogaćeni bogatim skupom metrika. ONGRAM je takođe baziran na eCST reprezentaciji. Stoga se on, slično SNEIPL-u, može proširiti da podržava različite jezike za reprezentaciju znanja. Dodatno, eCST reprezentacija ontoloških opisa nam je omogućila da definišemo nove i adaptiramo postojeće softverske metrike koje kvantifikuju unutrašnju kompleksnost ontoloških entiteta. U pogledu ekstrakcije mreža saradnje istraživača prezentovan je poluautomatski pristup pogodan za retke i fragmentisane mreže koji je baziran na heuristikama za detekciju imenskih sinonima i homonima.

Upotrebom alata koji su razvijeni u okviru disertacije formiran je eksperimentalni skup kompleksnih mreža koji obuhvata pet mreža saradnje klasa softverskih sistema pisanih u programskom jeziku Java, tri ontološke mreže koje opisuju strukturu jedne modularizovane ontologije semantičkog web-a i mrežu koja opisuje saradnju istraživača koji su svoje radove publikovali u srpskim matematičkim časopisima. Pokazano je da te mreže poseduju Watts-Strogatz osobinu malog sveta. Druge osobine

(slabo) povezanih komponenti kao što su stepen njihove jake povezanosti, paterni asortativnosti, distribucije stepeni čvorova, te karakteristike jako povezanih čvorova variraju kroz domene, sisteme i stepene apstrakcije reflektujući specifičnosti individualnih sistema.

Analiza jako povezanih komponenti prisutnih u ispitivanim softverskim mrežama je pokazala da one teže da se densifikuju (zgušnjavaju) sa veličinom komponente – prosečan unutrašnji stepen čvora se povećava sa veličinom komponente. Opaženi fenomen densifikacije se može modelovati stepenim zakonima (engl. *power-laws*) čiji se eksponenti mogu iskoristiti kao indikatori kvaliteta dizajna softverskog sistema. Kod softverskih i ontoloških mreža je takođe primećeno da postoji jak disbalans između ulaznog i izlaznog stepena jako povezanih čvorova. Povrh toga, dominacija ulaznog stepena nad izlaznim se povećava sa ukupnim stepenom čvora. Ovaj rezultat implicira da su jako povezani entiteti realnih softverskih/ontoloških sistema uzrokovani dominatno njihovom ponovnom upotrebom (engl. *internal reuse*), te da prisutnost visokog stepena vezivanja (što se generalno smatra lošom pojavom u domenu softverskog i ontološkog inženjerstva) može sugerisati samo negativne aspekte ponovnog iskorišćenja entiteta, a ne i negativne aspekte agregacije entiteta. U okviru teze je takođe uveden metrički baziran test poređenja grupa čvorova koji nam je omogućio da detaljno ispitamo karakteristike jako povezanih komponenti i jako povezanih čvorova softverskih/ontoloških mreža. Na koncu, pokazano je da se metrike za evaluaciju klasterisanja grafova mogu iskoristiti kao softverske i ontološke metrike kohezivnosti.

Kao poslednji originalni doprinos disertacije, istraživane su osobine mreže saradnje istraživača ekstrahovane iz bibliografskih zapisa sadržanih u elektronskoj biblioteci Matematičkog instituta Srpske akademije nauka i umetnosti (eLib). ELib digitalizuje najistaknutije matematičke časopise štampane u Srbiji. Cilj studije je bio da identifikujemo obrasce i dugoročne trendove naučne saradnje karakteristične za zajednicu koju domininantno čine srpski (jugoslovenski) matematičari. Analiza poveznih komponenti je otkrila topološku raznovrsnost u strukturi mreže: eLib mreža se sastoji iz relativno velikog broja komponenti koje se povinuju stepenom zakonu, većinu komponenti čine izolovani autori i male trivijalne komponente, ali takođe postoji i mali broj relativno velikih i kompleksnih komponenti povezanih autora. Primećeno je da najveće komponente eLib mreže poseduju strogu strukturu zajednica (u mreži se mogu uočiti jako kohezivne grupacije autora). Analiza evolucije mreže je pokazala da postoji šest karakterističnih perioda u razvoju eLib mreže koje se razlikuju po intenzitetu i tipu kolaborativnog ponašanja eLib autora. Analizom metrika na nivou autora je pokazano da je ugnježdenost autora u mreži bolji indikator produktivnosti i dugoročne zastupljenosti u eLib časopisima nego broj koautora.

Prošireni izvod

Veliki deo istraživačkih napora prethodnih godina je bio usmeren ka ispitivanju osobina mreža koje reprezentuju razne kompleksne tehnološke, socijalne i biološke sisteme. Watts i Strogatz [Watts and Strogatz, 1998b] su pokazali da tri velike i retke realne kompleksne mreže ispoljavaju efekat malog sveta (dužina najkraćeg puta između dva proizvoljna dva čvora je mala, drastično manja u odnosu na veličinu mreže) i visok stepen lokalnog klasterisanja (najbliži susedi proizvoljnog čvora formiraju relativno gust podgraf). Otkriće je bilo značajno jer klasična teorija slučajnih grafova, do tada korišćena u modelovanju kompleksnih mrežnih struktura, ne ume da objasni simultano prisustvo oba prethodno pomenuta kvaliteta u jednom velikom i retkom grafu. Analiza statističkih osobina grafova koji predstavljaju velike delove *World Wide Web*-a [Albert et al., 1999; Kumar et al., 1999] i Interneta na fizičkom nivou [Faloutsos et al., 1999] vodile su otkriću da se njihove distribucije stepeni (verovatnoća $P(k)$ da proizvoljno odabrani čvor ima tačno k incidentnih linkova, odnosno k najbližih suseda) povinuju stepenom zakonu (engl. *power-law*) oblika $P(k) \sim Ck^{-\gamma}$, što je osobina koju Erdős-Renyi model slučajnog grafa [Bollobás, 2001; Erdős and Rényi, 1959, 1960] ne predviđa. Mreže koje imaju prethodno opisani obrazac povezanosti čvorova se nazivaju *scale-free* mrežama [Barabasi and Albert, 1999]. Većina čvorova u *scale-free* mrežama su slabo povezani (imaju relativno mali broj suseda), ali *scale-free* mreže takođe sadrže malu, ali značajnu, frakciju čvorova (koji se takođe nazivaju habovi ili preferencijalni čvorovi) čiji je stepen povezanosti neočekivano veliki i teži da raste kako mreže evoluiraju. Dve važne konsekvence *scale-free* mrežne organizacije su:

- *"robust, yet fragile"* osobina [Albert et al., 2000; Bollobás and Riordan, 2003] – mreža gubi gigantsku povezanu komponentu uklanjanjem malog broja preferencijalnih čvorova, ali je robusna kada se uklanjaju slučajno odabrani čvorovi, i
- odsustvo kritične tačke u epidemijskim procesima koji se odvijaju na mreži [Pastor-Satorras and Vespignani, 2001].

Newman-ove studije kompleksnih mreža iz različitih domena su otkrile još dve važne karakteristike realnih mreža:

- (dis)assortativno vezivanje preferencijalnih čvorova (u slučaju asortativnog vezivanja habovi teže da budu povezani međusobno, dok se u slučaju disasortativnog vezivanja međusubno "izbegavaju") [Newman, 2002, 2003a], i
- postojanje klaster strukture, odnosno strukture zajednica (mreža se može particionisati u zajednice tako da su čvorovi unutar jedne zajednice gušće povezani međusobno nego sa ostatkom mreže) [Girvan and Newman, 2002; Newman and Girvan, 2004].

Analize realnih kompleksnih mreža iz raznih domena su vodile nastanku teorije kompleksnih mreža (koja se još naziva i naukom o mrežama, engl. *network science*) čiji je fokus na metrikama, statističkim

tehnikama, organizacionim i evolutivnim principima, matematičkim modelima i algoritmima koji mogu da kvantifikuju, objasne, reprodukuju i/ili identifikuju prethodno pomenute osobine realnih mreža [Albert and Barabási, 2002; Boccaletti et al., 2006; Costa et al., 2011; Newman, 2003b].

Istraživanje prezentovano u ovoj disertaciji je fokusirano na tehnike za ekstrakciju i analizu tri tipa realnih mreža:

- softverske mreže – mreže ekstrahovane iz izvornog koda računarskih programa koje reprezentuju dizajn softverskih sistema,
- ontološke mreže – mreže ekstrahovane iz ontologija semantičkog web-a koje opisuju strukturu deljenog znanja pogodnog za višekratnu upotrebu, i
- mreže saradnje istraživača – mreže ekstrahovane iz bibliografskih zapisa koje opisuju socijalne, samo-organizovane sisteme saradnje u nauci.

U okviru disertacije predložene su nove metode za ekstrakciju mreža iz pomenutih domena. Drugo, na nekoliko studija slučaja ilustrovani su benefiti mrežno orjentisane analize konkretnih sistema iz domena obuhvaćenih disertacijom. U poređenju sa prethodnim relevantim istraživanjima, analize prezentovane u disertaciji nisu čisto topološke, nego kombinuju tehnike i metrike razvijene u okviru teorije kompleksnih mreža sa metrikama iz konkretnog domena.

Moderni softverski sistemi se sastoje iz više stotina ili čak hiljada međusobno povezanih entiteta pozicioniranih na različitim nivoima apstrakcije. Na primer, kompleksni softverski sistemi napisani u programskom jeziku Java se sastoje od paketa, paketi grupišu srodne klase i interfejse, dok klase i interfejsi deklarišu ili definišu srodne metode i atribute. Interakcije, zavisnosti, odnosi ili saradnje između softverskih entiteta određuju različite tipove softverskih mreža. U zavisnosti od stepena apstrakcije možemo razlikovati specifične softverske mreže kao što su mreže saradnje paketa, klasa i metoda [Hylland-Wood et al., 2006; Myers, 2003; Valverde et al., 2002]. Dodatno, različiti tipovi povezanosti entieta istog tipa određuju različite softverske mreže [Wheeldon and Counsell, 2003]. Značaj ekstrakcije i analize softverskih mreža se ogleda u nekoliko sfera:

- analiza kompleksnosti dizajna i evolucije softverskih sistema [Baxter et al., 2006; Bhattacharya et al., 2012; Chatzigeorgiou et al., 2006; Concas et al., 2007; de Moura et al., 2003; Hylland-Wood et al., 2006; Ichii et al., 2008; Jenkins and Kirk, 2007; Louridas et al., 2008; Myers, 2003; Puppin and Silvestri, 2006; Taube-Schock et al., 2011; Valverde et al., 2002; Šubelj and Bajec, 2012; Wang et al., 2009, 2013; Wen et al., 2009a; Wheeldon and Counsell, 2003],
- računanje softverskih metrika koje reflektuju kvalitet dizajna softverskog sistema [Bieman and Kang, 1995; Briand et al., 1996, 1998, 1999; Chidamber and Kemerer, 1994; Fenton, 1991; Hitz and Montazeri, 1995],
- formiranje baze fakata u procesu reverznog inženjeringa softverskih sistema [Ducasse et al., 2000; Ebert et al., 2002; Kienle and Müller, 2010; Raza et al., 2006].

Dodatno, softverske mreže se mogu iskoristiti za vizuelizaciju softverskih sistema [Lanza and Ducasse, 2003; Risi and Scanniello, 2012], klasterisanje softverskih entiteta u procesu identifikacije arhitekture sistema na visokom nivou apstrakcije [Chiricota et al., 2003; Mancoridis et al., 1998; Mitchell and Mancoridis, 2006; Scanniello et al., 2010; Wu et al., 2005], identifikaciju i uklanjanje sumnjivih delova izvornog koda (engl. *"bad smell"*, simptomatični deo koda koji može ukazivati na neki dublji problem

ili na kršenje osnovnih principa dizajna softverskih sistema) [Oliveto et al., 2011], lociranje koncepata u izvornom kodu [Scanniello and Marcus, 2011], u procesu razumevanja programa koji prolazi kroz inkrementalne modifikacije [Buckner et al., 2005], za identifikaciju obrazaca dizajna (engl. *design patterns*) u izvornom kodu [Lucia et al., 2009], i predikciji grešaka u softverskim sistemima [Bhattacharya et al., 2012; Bird et al., 2009b; Oyetoyan et al., 2013; Tosun et al., 2009; Zimmermann and Nagappan, 2008].

Termin ontologija ima veoma široko značenje. U domenu informacionih nauka ontologija se definiše kao specifikacija konceptualizacije [Gruber, 1993]. Ontologija formalno opisuje koncepte i relacije prisutne u nekom domenu diskursa i kao takva modeluje određeni deo realnosti. U kontekstu vizije semantičkog Web-a [Berners-Lee et al., 2001; Shadbolt et al., 2006], ontologije predstavljaju formalne specifikacije deljenog znanja pogodnog za višekratnu upotrebu i mogu se iskoristiti u procesima zaključivanja vođenog podacima, integraciji podataka i interoperativnosti računarskih programa (softverskih agenata) koji procesiraju WWW resurse. Ontološke mreže predstavljaju zavisnosti između ontoloških entiteta prisutnih u ontološkom opisu. Ontološki opisi sadrže aksiome koje definišu asocijacije između ontoloških entiteta, kao i aksiome koje definišu specifične osobine entiteta, a koje se mogu iskoristiti za proveru konzistentnosti opisa, te izvođenje novih asocijacija između entiteta. Kako su ontološke mreže osnova ontoloških opisa to se one prirodno koriste za evaluaciju njihove kompleksnosti [Cheng and Qu, 2008; Ding et al., 2010; Ge et al., 2010; Gil and García, 2006; Ma and Chen, 2007; Theoharis et al., 2008b; Zhang, 2008; Zhang et al., 2010] i kvaliteta [Oh et al., 2011; Orme et al., 2006; Tartir et al., 2005; Yao et al., 2005; Zhang et al., 2010]. Slično softverskim mrežama, ontološke mreže se mogu iskoristiti u raznim zadacima reverznog inženjeringa i razumevanja ontologija kao što su identifikacija ključnih ontoloških koncepata [Hoser et al., 2006], automatska modularizacija ontoloških opisa [Coskun et al., 2011; Stuckenschmidt and Schlicht, 2009], njihovo sumiranje [Zhang et al., 2007] i vizualizacija [Katifori et al., 2007].

Saradnja između istraživača je jedan od ključnih faktora naučnog progresa. Mreže saradnje istraživača su socijalne mreže u kojima su dva istraživača povezana neusmerenim linkom ukoliko su koautori na bar jednom radu. Analizom mreže saradnje istraživača možemo steći uvid u strukturu i evoluciju akademske zajednice predstavljene mrežom [Newman, 2001b,c,d, 2004c]. Postojeće empirijske studije mreža saradnje istraživača pokrivaju širok opseg naučnih disciplina: fiziku [Barrat et al., 2004; Newman, 2001b,c,d, 2004c; Pan and Saramäki, 2012; Ramasco et al., 2004], matematiku [Barabasi et al., 2002; Batagelj and Mrvar, 2000; Brunson et al., 2014; Cerinšek and Batagelj, 2014; Grossman, 2002a,b], računarske nauke [Bird et al., 2009a; Biryukov and Dong, 2010; Divakarmurthy and Menezes, 2013; Elmacioglu and Lee, 2005; Franceschet, 2011; Huang et al., 2008; Newman, 2001b,c,d, 2004c; Shi et al., 2011; Staudt et al., 2012], biomedicinu [Newman, 2001b,c,d, 2004c], ekonomiju [Goyal et al., 2006], menadžment [Acedo et al., 2006], bibliotečke i informacione nauke [Abbasi et al., 2012a; Yan and Ding, 2009], i sociologiju [Moody, 2004]. Struktura i evolucija naučnih saradnji je takođe istraživana za uže stručne discipline poput genetskog programiranja [Luthi et al., 2007; Tomasini and Luthi, 2007; Tomassini et al., 2007], evolutivnih algoritama [Cotta and Guervós, 2007; Cotta and Merelo, 2005], računarske geometrije [Hui et al., 2011], pribavljanja informacija [Ding, 2011], informacionih sistema [Zhai et al., 2014], fuzije informacija [Johansson et al., 2011], inteligencije u računarskim igrama [Lara-Cabrera et al., 2014], vizuelizacije informacija [Börner et al., 2005], analize socijalnih mreža [Otte and Rousseau, 2002], i ekonofizike [Fan et al., 2004]. Dodatno, studije koje se bave identifikacijom obrazaca i trendova saradnje specifičnih zajednica autora okupljenih oko srodnih istraživačkih tema su takođe bazirane na mrežama saradnje istraživača – mrežama koautorstva radova prezentovanih na naučnim konferencijama [Cheong and Corbitt, 2009a,b; Hassan and Holt, 2004; Liu

et al., 2005; Nascimento et al., 2003; Ochoa et al., 2009; Pham et al., 2012; Reinhardt et al., 2011; Smeaton et al., 2003; Vidgen et al., 2007; Xu and Chau, 2006], te radova publikovanih u individualnim naučnim časopisima [Borner et al., 2004; Chen et al., 2013; Fatt et al., 2010; Fischbach et al., 2011; Hou et al., 2008; Li et al., 2010; Martin et al., 2013]. Tipične osobine mreža saradnje istraživača su postojanje gigantske povezane komponente, *scale-free* osobina, fenomen malog sveta, struktura zajednica, te fenomen densifikacije saradnje (povećanja prosečnog broja koautora kako mreža evoluira). Mreže saradnje istraživača se takođe koriste za rangiranje i identifikaciju uticajnih autora u bibliografskim bazama podataka [Gollapalli et al., 2011; Mimno and McCallum, 2007], sugestiju recezenata za naučni rad [Rodriguez and Bollen, 2008; Rodriguez et al., 2006], i predikciju budućih saradnji između istraživača [Guns and Rousseau, 2014; Liben-Nowell and Kleinberg, 2003; Yan and Guns, 2014]. Kombinovane sa mrežama afilijacija mreže saradnje istraživača se mogu iskoristiti za istraživanje naučne saradnje na institucionalnom i internacionalnom nivou. Sa druge strane, studija koja obuhvata kombinovanu analizu mreže saradnje istraživača i mreže citata može identifikovati međusobne odnose između autorstva i citata, te uticaj istraživačke saradnje na praksu citiranja [Martin et al., 2013; Wallace et al., 2012].

U pogledu ekstrakcije softverskih mreža, kao prvi orginalni doprinos disertacije, predložen je SNEIPL [Savić et al., 2012, 2014] – proširiv, jezički nezavisan pristup ekstrakciji softverskih mreža baziran na jezički nezavisnoj, *enriched Concrete Syntax Tree* (eCST) reprezentaciji izvornog koda [Rakić and Budimac, 2011a,b]. SNEIPL je realizovan kao jedan od zadnjih delova SSQSA (*Set of Software Quality Static Analyzers*) okruženja [Budimac et al., 2012; Kolek et al., 2013; Rakić et al., 2013]. eCST reprezentacija proširuje stabla parsiranja takozvanim univerzalnim čvorovima koji su predefinisani semantički markeri sintaksnih konstrukcija. Skup eCST univerzalnih čvorova sadrži čvorove koji markiraju definicije softverskih entiteta koji se pojavljuju kao čvorovi u softverskim mrežama, kao i čvorove koji su polazna tačka prilikom rekonstrukcije zavisnosti između softverskih entiteta. Vertikalne zavisnosti, zavisnosti između entiteta koji su pozicionirani na različitim nivoima apstrakcije, se identifikuju na osnovu hijerarhije univerzalnih čvorova u ulaznim eCST stablima. Horizontalne zavisnosti, zavisnosti između entiteta sa istog nivoa apstrakcije, se identifikuju na osnovu algoritma za rezoluciju imena koji je baziran na informacijama sadržanim u *import* naredbama (naredbe markirane IMPORT_DECL univerzalnim čvorom), lokalnim tabelama simbola vezanim za konkretne opsege (funkcije, blokove), vertikalnim zavisnostima između softverskih entiteta, leksičkim pravilima opsega vidljivosti promenljive i rapidnoj analizi tipova (engl. *rapid type analysis*) [Bacon and Sweeney, 1996] koja je adaptirana za eCST reprezentaciju. SNEIPL identifikuje različite tipove zavisnosti između softverskih entiteta (zavisnosti između paketa, klasa/modula, funkcija/metoda, i funkcija i promenljivih) te je tako u mogućnosti da ekstrahuje mreže koje reprezentuju softverske sisteme na različitim nivoima apstrakcije.

U kontrolisanom eksperimentu je pokazano da SNEIPL ekstrahuje izomofrmne softverske mreže iz strukturno i semantički ekvivalentnih programa napisanih u različitim programskim jezicima. Potom je pristup demonstriran ekstrakcijom mreža koje reprezentuju realne softverske sisteme pisane u različitim programskim jezicima (Java, Modula-2 i Delfi) koji pripadaju različitim programskim paradigmama. Korektnost i kompletnost pristupa je ispitivana poređenjem mreža saradnje klasa ekstrahovanih iz 10 realnih softverskih sistema otvorenog koda pisanih u Javi sa odgovarajućim mrežama koje su ekstrahovane upotrebom drugih alata. Naime, u uporednoj analizi su korišćeni:

- *DependencyFinder* – jezički zavisan alat koji mrežne reprezentacije Java softverskih sistema formira na osnovu bajt koda. Alat ima jako dobre preporuke, kako od korisnika koji dolaze iz

akademske zajednice, tako i od korisnika iz industrije softvera[1].

- *Doxygen* – jezički nezavisan ekstraktor baziran na fazi plitkom parsiranju. Ovaj alat je korišćen u relevantim studijama koje se bave analizom softverskih mreža [Hylland-Wood et al., 2006; Myers, 2003], te raznorodnim empirijskim studijama u domenu softverskog inženjerstva [Berner et al., 2005; Capiluppi and Boldyreff, 2008; Capiluppi and Knowles, 2009; Capiluppi et al., 2011; Nguyen and Tran, 2010].

Uporedna analiza je pokazala da su mreže ekstrahovane SNEIPL-om jako slične onima koje se dobijaju korišćenjem *DepedencyFinder*-a (preko 90% sličnosti u svim studijama slučaja), te da SNEIPL daje daleko preciznije rezultate nego *Doxygen*. Kako se na osnovu softverskih mreža računaju softverske metrike koje kvantifikuju kompleknost i kvalitet dizajna softverskih sistema, to SNEIPL omogućava i jezički nezavisno računanje određene klase softverskih metrika – metrika kohezivnosti i metrika uzajamne povezanosti softverskih entiteta (engl. *coupling metrics*). Analizom postojećih, široko korišćenih, jezički nezavisnih alata i okruženja za reverzni inženjering softverskih sistema (*Rigi* [Kienle and Müller, 2010], *Moose* [Ducasse et al., 2000], *Gupro* [Ebert et al., 2002] i *Bauhaus* [Raza et al., 2006]) je ustanovljeno da isti omogućuju jezički nezavisnu reprezentaciju ekstrahovanih fakata (mrežnih modela izvornog koda), ali ne i jezički nezavisnu ekstrakciju fakata. Naime, pomenuti alati su realizovani na način da:

- za svaki podržani programski jezik postoji zaseban ekstraktor fakata [Ducasse et al., 2000; Ebert et al., 2002; Kienle and Müller, 2010], ili
- ekstrakcija fakata je pracijalno jezički nezavisna u smislu da se softverske mreže ekstrahuju iz jezički nezavisne reprezentacije niskog nivoa (nivoa naredbi) za određen podskup podržanih jezika, dok za ostale podržane jezike postoje zasebni ekstraktori mreža [Raza et al., 2006].

Stoga SNEIPL u odnosu na prethodno pomenute alate i okruženja omogućava jezički nezavisan pristup formiranju baze fakata koje se koriste u reverznom inženjeringu softverskih sistema.

"Low coupling–high cohesion" je jedan od bazičnih principa softverskog inženjerstva [Yourdon and Constantine, 1979]. Princip kazuje da uzajamna povezanost između softverskih modula treba da bude minimalna moguća i da istovremeno moduli treba da budu logički i funkcionalno koherentni, što se pak ispoljava jakom uzajamnom povezanošću elemenata od kojih je jedan modul komponovan. U disertaciji je, kao originalan doprinos, uvedena ideja primene metrika za evaluaciju tehnika klasterisanja grafa (GCE metrike) kao softverskih metrika kohezivnosti [Savić and Ivanović, 2014]. Naime, doslovno praćenje *"low coupling–high cohesion"* principa neizbežno mora rezultovati klaster strukturom softverskih mreža, gde klasteri korespondiraju sa modulima, a elementi jednog klastera su gušće povezani međusobno nego sa ostatkom mreže. Standardne metrike kohezivnosti koje se koriste u softverskom inženjerstvu ocenjuju kohezivnosti softverskih modula ignorišući eksterne zavisnosti (reference na entitete van modula) [Bieman and Kang, 1995; Chidamber and Kemerer, 1994; Hitz and Montazeri, 1995; Lee et al., 1995]. Za razliku od njih, GCE metrike ne ocenjuju kohezivnosti modula izolovano od ostatka sistema, odnosno bazirane su na ideji da minimizacija eksternih zavisnosti povećava stepen kohezivnosti modula. Osobine GCE metrika kao softverskih metrika su ispitivane koristeći teorijski okvir koji su uveli Briand i koautori [Briand et al., 1996, 1998]. Pokazano je da sledeće GCE metrike, *conductance*, *expansion*, *cut-ratio*, *average-ODF*, *maximum-ODF* i *Flake-ODF* [Leskovec et al., 2010], zadovoljavaju dve najvažnije osobine metrika kohezivnosti (monotonost i osobinu spajanja), ali takođe

[1] Preporuke se mogu naći na *web* stranici alata – http://depfind.sourceforge.net

da poseduju izvesna ograničenja kojih korisnici moraju biti svesni prilikom upotrebe. U ovom pogledu GCE metrike nisu izuzetak, teorijska analiza prezentovana u [Briand et al., 1998] pokazuje da zapravo relativno mali broj standardnih metrika kohezivnosti zadovoljava sve osobine prethodno pomenutog teorijskog okvira. Naime, standardne metrike kohezivnosti koje zadovoljavaju sve osobine teorijskog okvira (TCC i LCC metrike definisane u [Bieman and Kang, 1995]) su bazirane na apsolutnoj gustini grafa kojeg čine interne zavisnosti između elemenata jednog modula. Sa druge strane, GCE metrike su bazirane na principu relativne gustine, kontrastu između gustine internih i eksternih zavisnosti. Stoga se GCE metrike mogu posmatrati kao komplementarne metrike standardnim metrikama kohezivnosti. Dodatno, GCE metrike imaju jasno definisane pragove koje omogućavaju klasifikaciju modula na jako kohezivne, zadovoljavajuće kohezivne i slabo kohezivne. U budućem radu će biti izvedena empirijska analiza (analiza korelacija između GCE metrika i standardnih metrika kohezivnosti) koja nadograđuje teorijsku analizu prezentovanu u disertaciji.

Koristeći SNEIPL formiran je eksperimentalni skup softverskih mreža koje su potom analizirane. Eksperimentalni skup čine 5 mreža saradnje klasa koje reprezentuju široko korišćene softverske sisteme otvorenog koda napisane u programskom jeziku Java (*Tomcat*, *Lucene*, *Ant*, *Xerces* i *JFreeChart*). Za razliku od sličnih studija, analize softverskih mreža prezentovane u disertaciji nisu čisto topološke. Naime, svaki čvor (Java klasa) u softverskoj mreži je opisan metričkim vektorom koji sadrži metrike nezavisne od domena (metrike koje se koriste u analizi kompleksnih mreža i koje su definisane na bilo kojem usmerenom grafu) i metrike iz domena (softverske metrike unutrašnje kompleksnosti i dizajna). Takođe je uveden metrički orjentisan statistički test koji poredi dve grupe čvorova u mreži pri čemu se grupe formiraju na bazi nekog topološkog kriterijuma. Test se zasniva na sekvencijalnoj (sekvencijalnoj po elementima metričkog vektora) primeni Mann-Whitney testa [Mann and Whitney, 1947] i verovatnoćama superiornosti. Na osnovu testa se identifikuju one metrike za koje ne postoje statistički značajne razlike u metričkim vrednostima poređenih grupa čvorova, metrike za koje postoje statistički značajne razlike i metrike za koje postoje drastične statistički značajne razlike (stepen drastičnosti razlika se utvrđuje na osnovu verovatnoća superiornosti).

Analiza povezanih komponenti je pokazala da softverskih mreže iz eksperimentalnog skupa imaju gigantske slabo povezane komponente koje ispoljavaju osobinu malog sveta u *Watts-Strogatz* smislu i obrasce slabog disasortativnog vezivanja. Dodatno ih karakteriše prisustvo velikih cikličnih zavisnosti koje su posledica relativno velikih jako povezanih komponenti. Analizom jako povezanih komponenti je utvrđeno da postoji jaka *Spearman*-ova korelacija između broja čvorova u jako povezanoj komponenti i prosečnog unutrašnjeg stepena čvora u komponenti. To znači da jako povezane komponente teže da se zgušnjavaju (densifikuju) sa veličinom komponente. Uočeno je da se fenomen densifikacije može dobro opisati stepenim zakonom čiji eksponent se može iskoristiti kao indikator kvaliteta dizajna softverskog sistema. Rezultati metrički orjentisanog testa poređenja grupa čvorova su pokazali da u dva ispitivana softverska sistema (*Ant* i *JFreeChart*) postoji jaka tendencija da jako povezane komponente sadrže centralne i najbitnije klase definisane u sistemu.

U poređenju sa sličnim studijama [de Moura et al., 2003; Hylland-Wood et al., 2006; Jenkins and Kirk, 2007; Louridas et al., 2008; Myers, 2003; Valverde et al., 2002; Wheeldon and Counsell, 2003], distribucije stepeni čvorova su testirane ne samo na stepeni zakon, nego i na eksponencijalnu i log-normalnu raspodelu upotrebom robusnog statističkog testa uvedenog u [Clauset et al., 2009]. Primećeno je da se repovi (engl. *tails*) empirijskih distribucija mogu modelovati stepenim zakonom, ali i da su alternativne distribucije takođe verodostojni ili čak i bolji modeli. Dodatno, log-normalna distribucija bolje opisuje povezanost čvorova uzimajući u obzir čitav opseg vrednosti stepena čvora.

Ovaj rezultat sugeriše da se evolucija softverskih mreža zasniva na principu skoro linearnog preferencijalnog vezivanja (engl. *nearly-linear preferential attachment*) [Redner, 2005]. Distribucije stepeni čvorova ispitivanih softverskih mreža su distribucije teškog repa (engl. *heavy-tailed distribution*). Stoga ispitivane softverske mreže sadrže habove – čvorove izrazito velikog stepena (klase koju imaju veliku vrednosti *Chidamber-Kemerer* CBO metrike [Chidamber and Kemerer, 1994]). Primećeno je da postoji disbalans između ulaznog i izlaznog stepena habova gde ulazni stepen teži da dominira nad izlaznim stepenom. Štaviše, u četiri od pet ispitivanih sistema (svi osim *Xerces*-a) nivo dominacije ulaznog stepena nad izlaznim se povećava sa ukupnim stepenom. Ovaj rezultat implicira da su jako povezane klase realnih softverskih sistema direktno prouzrokovane njihovom ponovnom internom upotrebom (engl. *internal reuse*). Kako se jako povezani entiteti smatraju negativnom pojavom u domenu softverskog inženjerstva (jer njihovo postojanje označava veliko odstupanje od *"low coupling"* principa), to navedeni rezultat nadalje povlači da prisutnost visokog stepena vezivanja može sugerisati samo negativne aspekte ponovnog iskorišćenja entiteta, ne i negativne aspekte agregacije entiteta. Primena metrički orjentisanog testa poređenja grupa čvorova je pokazala da habovi ispitivanih softverskih sistema teže da budu drastično veće (sadrže drastično veći broj linija koda) i centralnije klase u odnosu na ostatak klasa definisanih u sistemu. Sa druge strane, u većini ispitivanih sistema (svim osim *Lucene*-a) nema statistički značajnih razlika u pogledu specijalizacije habova u odnosu na "obične" klase.

U pogledu ekstrakcije ontoloških mreža predložen je ONGRAM – proširiv, jezički nezavisan pristup ekstrakciji ontoloških mreža baziran na jezički nezavisnoj eCST reprezentaciji izvornog koda [Rakić and Budimac, 2011a,b]. Slično SNEIPL-u, ONGRAM je realizovan kao jedan od zadnjih delova SSQSA okruženja [Budimac et al., 2012; Kolek et al., 2013; Rakić et al., 2013] pošto je SSQSA proširena da podržava OWL2 jezik za opis ontologija semantičkog Web-a [Savić et al., 2013]. ONGRAM omogućava proširivost na različite jezike za reprezentaciju znanja: proširivost SNEIPL-a i ONGRAM-a je direktna posledica proširivosti SSQSA okruženja i njihove baziranosti na jezički nezavisnoj eCST reprezentaciji. Povrh toga, eCST reprezentacija ontoloških opisa je omogućila definisanje novih i adaptaciju postojećih softverskih metrika unutrašnje kompleksnosti. Naime, u tezi je predložena nova metrika kompleksnosti izraza (engl. *expression complexity*) koja kvantifikuje kompleknost OWL aksioma i ontoloških modula. Takođe je pokazano kako se skup *Halstead*-ovih metrika [Halstead, 1977] i *Henry-Kafura* kompleksnost mogu adaptirati za evaluaciju ontologija [Henry and Kafura, 1981]. Na osnovu mrežne reprezentacije modularizovanih ontologija ONGRAM računa:

- u literaturi prethodno uvedene ontološke metrike dizajna [Orme et al., 2006; Tartir et al., 2005],
- metrike nezavisne od domena koje se primenjuju u analizi kompleksnih mreža (metrike centralnosti),
- softverske metrike dizajna adaptirane za evaluaciju ontologija [Žontar and Heričko, 2012; Zhang et al., 2010],
- hibridne metrike koje kombinuju metrike unutrašnje kompleksnosti sa metrikama dizajna (*Henry-Kafura* kompleksnost),

Dodatno, u disertaciji je pokazano da se GCE metrike (metrike za ocenu tehnika klasterisanja grafa), slično kao za softverske sisteme, mogu koristiti za evaluaciju kohezivnosti ontoloških modula. GCE metrike su takođe uključene u skup metrika koje računa ONGRAM. Stoga, ONGRAM formira ontološke mreže čiji su čvorovi obogaćeni bogatim, hibridnim skupom metrika.

Analiza ontoloških mreža nam pomaže da razumemo kompleksnost dizajna ontološkog opisa. Kao orginalni doprinos disertacije prezentovana je analiza ontoloških mreža koje na različitim nivoima apstrakcije predstavljaju SWEET ontologiju (*Semantic Web for Earth and Environmental Terminology*), javno dostupnu *NASA*-inu modularizovanu ontologiju koja definiše terminologiju u domenu geoloških nauka [Raskin and Pan, 2005]. Poseban fokus je stavljen na analizu mreže ontoloških modula jer smo hteli da razumemo modularizaciju formalizacije nekog realnog znanja, te da ocenimo kvalitet te modularizacije. Prilikom analize SWEET mreža korišćen je isti metodološki okvir kao pri analizi softverskih mreža. U odnosu na relevantne studije ontoloških mreža [Cheng and Qu, 2008; Ding et al., 2010; Ge et al., 2010; Gil and García, 2006; Ma and Chen, 2007; Theoharis et al., 2008b; Zhang, 2008; Zhang et al., 2010], analize prezentovane u disertaciji nisu čisto topološke, nego proširene bogatim skupom metrika koje su dobijene koristeći ONGRAM. Pokazano je da SWEET mreže imaju ili tačno jednu ili gigantsku slabo povezanu komponentu koja ispoljava osobine malog sveta. Međutim, na različitim nivoima apstrakcije SWEET mreže ispoljavaju različite obrasce asortativnosti i jake povezanosti:

- Mreža ontoloških modula pokazuje slabo asortativno vezivanje i ima gigantsku jako povezanu komponentu koja obuhvata više od polovine SWEET modula.
- Mreža ontoloških koncepata pokazuje značajno disasortativno vezivanje i ima veliki broj relativno malih jako povezanih komponenti,
- Mreža nasleđivanja ontoloških koncepata takođe pokazuje značajno disasortativno vezivanje, ali ima tačno jednu, minimalnu jako povezanu komponentu (komponentu koju čine dva ontološka koncepta).

Upotrebom metrički orjentisanog testa poređenja grupa čvorova u mreži je pokazano da mreža SWEET ontoloških modula ima jako povezano jezgro (engl. *strongly connected core*) koje obuhvata najvažnije SWEET ontološke module, odnosno module koji imaju najveći stepen ponovne interne iskorišćenosti. Rezultati analize distribucije stepeni čvorova ispitivanih ontoloških mreža su pokazali da mreže ne poseduju *scale-free* osobinu, ali da sadrže habove – ontološke module i koncepte sa visokim stepenom povezanosti sa drugim ontološkim modulima i konceptima, respektivno. Slično kao kod softverskih sistema, utvrđeno je da su ontološki entiteti sa visokim stepenom povezanosti dominantno prouzrokovani njihovim ponovnim iskorišćenjem. Primenom metrički orjentisanog testa poređenja grupa čvorova u mreži pokazano je da habovi teže da budu znatno veći, centralniji i važniji moduli nego oni koji nisu habovi. Na osnovu vrednosti GCE metrika je izvršena klasifikacija ontoloških modula i pokazano je da SWEET moduli ispoljavaju zadovoljavajući stepen kohezivnosti. Agregirajući rezultate analiza izveden je zaključak o kvalitetu SWEET modularizacije:

- SWEET modularizacija ne reflektuje "low coupling – high cohesion" princip. SWEET moduli ispoljavaju zadovoljavajući stepen kohezivnosti, ali takođe postoje SWEET moduli koji imaju visok stepen povezanosti. Međutim, visok stepen povezanosti SWEET modula je dominatno prouzrokovan ponovnim iskorišćenjem modula koji se u domenu ontološkog inženjerstva može smatrati dobrom praksom i ne može prouzrokovati probleme ukoliko je konzistentnost i koherentnost[2] svakog modula verifikovana prilikom njegovog uključenja u ontologiju.
- SWEET ontologija ima jako povezano jezgro koje obuhvata više od polovine ontoloških modula. Ciklične strukture su jako teške za razumevanje pošto se ne mogu topološki sortirati – nije

[2]Ontološki modul je inkoherentan ukoliko sadrži bar jedan koncept koji ne može biti instanciran. Sa druge strane, ontološki modul je inkonzistentan ako sadrži kontradiktorne aksiome.

moguće urediti SWEET module u slojeve koji se daju zasebno analizirati u svrhu razumevanja ontološkog dizajna i sadržaja ontoloških modula. Naime, ukoliko želimo da razumemo sadržaj jednog modula iz jako povezanog jezgra moramo biti u potpunosti svesni sadržaja svih modula iz jezgra. Dodatno ukoliko želimo da iskoristimo neki SWEET modul iz jako povezanog jezgra u nekoj drugoj ontologiji, onda se svi moduli iz jezgra moraju uključiti u tu ontologiju.

Analiza korelacija između ontoloških metrika je pokazala da ne postoje jake korelacije između GCE metrika i metrika kohezivnosti koje se oslanjaju samo na unutrašnje zavisnosti između entiteta u modulu (unutrašnja gustina modula i broj povezanih komponenti u grafu sačinjenom od elemenata modula). Jake korelacije takođe nisu primećene između metrike kompleksnosti izraza koja je uvedena u disertaciji i drugih metrika unutrašnje kompleksnosti (broj linija koda, *Halstead*-ove metrike). Drugim rečima, empirijska analiza je pokazala korisnost metrika uvedenih u disertaciji pošto prethodne uvedene metrike iz odgovarajućih kategorija ne mogu ukazati na onotološke module koji imaju male ili velike vrednosti GCE metrika i metrike kompleksnosti izraza.

Analiza mreža koje reprezentuju SWEET ontologiju je pokazala da mrežno bazirana analiza proširena bogatim skupom ontoloških metrika nam može pomoći ne samo da razumemo kompleksnost modularizacije nekog formalizovanog znanja, nego i da vrlo konkretno ocenimo njen kvalitet. Stoga ćemo u budućem radu primenjivati istu metodologiju analize ontoloških mreža i na druge modularizovane ontologije kako bi smo utvrdili mogućnost za njenu rasprostranjenu upotrebu u evaluaciji kvaliteta modularizovanih ontologija.

Ekstrakcija mreža saradnje istraživača je znatno drugačija vrsta problema od ekstrakcije softverskih i ontoloških mreža. Naime, čvorovi softverskih i ontoloških mreža se mogu jednostavno formirati, jer su softverski i ontološki identiteti jednoznačno određeni njihovim potpuno kvalifikovanim (engl. *fully-qualified*) imenima. Sa druge strane imena autora koja se pojavljuju u bibliografskim zapisima se ne mogu koristiti za jednoznačnu identifikaciju autora:

- Više različitih osoba može imati isto ime. Ovaj fenomen se naziva imenskom homonimijom.
- Jedna osoba može biti reprezentovana sa više imena usled ortografskih varijacija, grešaka u sricanju imena, transliteracije imena, ime se može promeniti kroz vreme (recimo usled braka), itd. Ovaj fenomen se naziva imenskom sinonimijom.

U disertaciji je predložen polu-automatski pristup ekstrakciji mreža koji je pogodan za retke i fragmentisane mreže i baziran na heuristikama za identifikaciji imenskih homonima i sinonima. Imenski homonimi se određuju na osnovu strukture ego-mreža čvorova u mreži koja je formirana pod pretpostavkom da se fenomeni imenske homonimije i sinonimije ne ispoljavaju u ulaznim podacima. Ego-mreža jednog autora je mreža koja okuplja sve njegove koautore i reflektuje saradnju između njih. Ukoliko je autor artikulacioni čvor u svojoj ego-mreži tada se ručno proverava i odlučuje (na osnovu eksternih resursa, *web* pretrage, itd.) da li on predstavlja više istraživača koji imaju isto ime. Imenski sinonimi se određuju u dve faze pošto su imenski homonimi prethodno utvrđeni. Identifikacija imenskih sinonima se obavlja na osnovu postojećih mera sličnosti stringova i klasifikacije imena na puna i skraćena. Kod punih imena i ime i prezime su dati u punom obliku, dok je kod skraćenih imena prezime dato u punom obliku, a lično ime redukovano na inicijale. U prvoj fazi se mere sličnosti stringova primenjuju bez ograničenja, odnosno bez primene neke blokirajuće funkcije koja redukuje broj parova imena za koje se računaju mere sličnosti. U drugoj fazi se primenjuje blokirajuća funkcija koja uzrokuje da se mere sličnosti računaju za one autore koji pripadaju istoj povezanoj komponenti. Ova faza je motivisana opservacijom da ukoliko dva različita čvora reprezentuju istog autora tada

je jako verovatno da su ta dva čvora indirektno povezani preko jednog ili više zajedničkih koautora. Predložen pristup je potom primenjivan za ekstrakciju mreže saradnje istraživača iz bibliografskih zapisa elektronske biblioteke Matematičkog instituta Srpske akademije nauka i umetnosti – eLib [Mijajlović et al., 2010]. Ekstrahovana mreža je potom analizirana kako bi se utvrdile karakteristike socijalne strukture koju čine autori koji su svoje radove publikovali u srpskim matematičkim časopisima koje eLib indeksira [Savić et al., 2015; Savić et al., 2014].

U disertaciji je ispitivana struktura i evolucija eLib mreže u periodu od 1932 do 2011. godine. Tehnike i metrike koje se koriste u analizi socijalnih mreža su kombinovane sa metrikama produktivnosti i dugoročne zastupljenosti. Produktivnost autora je merena brojem publikovanih radova koristeći normalnu šemu [Lindsey, 1980] koja daje po 1 poen svakom autoru rada. Zastupljenost autora je merena brojem godina koje su protekle od publikovanja njegovog prvog do njegovog poslednjeg rada. Dodatno ispitivan je kontekst u kojem mreža evoluira analizom dinamike broja publikacija na godišnjem nivou, dinamike broja zastupljenih autora na godišnjem nivou, karakteristika publikovanih radova (broj autora po radu i procenat radova sa tačno jednim autorom na godišnjem nivou) i karakteristika autora (broj radova po autoru i procenat autora "povratnika" – autora koji su prethodno publikovali radove u eLib časopisima).

Ukupno 6480 radova je publikovano u eLib časopisima u ispitivanom vremenskom periodu. U pogledu dinamike broja publikovanih radova postoje vremenski periodi kada broj publikacija na godišnjem nivou raste, ali takođe i periodi kada broj publikacija na godišnjem nivou opada. Tokom i neznatno nakon Drugog svetskog rata nisu štampani radovi u eLib časopisima. Prvi duži trend rasta broja publikacija na godišnjem nivou se može uočiti u periodu 1947–1953. Period od 1953–1963 karakteriše pad načne produkcije. Nakon toga počinje drugi stabilni period rasta produkcije koji je trajao do 1980. godine. Sa smrću Josipa Broza Tita počinje period ekonomske krize i nacionalnih tenzija u zemlji koji su vodili građanskom ratu i raspadu SFRJ. Ovi događaji su evidentno uticali na produktivnost eLib zajednice autora: najduži period kontinualnog smanjenja broja publikacija na godišnjem nivou je trajao između 1980. i 1996. godine, pri čemu se tokom rata može uočiti ekstremno mala produktivnost autora. Nakon građanskog rada i relativne stabilizacije političke situacije počinje poslednji period rasta produktivnosti koji traje do kraja ispitivanog perioda. Najveći diskontinuitet u dinamici broja radova se dešava posle 2000. godine kada broj publikacija na godišnjem nivou postaje znatno veći nego u prethodnim godinama. Pomenuti diskontinuitet se može objasniti promenom politike Ministarstva nauke Republike Srbije kojom se broj publikacija istraživača uzima kao mehanizam vrednovanja njihovog kvaliteta. Dodatno, u relativno kratkom periodu posle 2000. godine osnivaju se nova četiri časopisa koja bivaju uključena u eLib. Dinamika broja zastupljenih autora na godišnjem nivou strogo korelira sa dinamikom broja publikacija.

U pogledu karakteristika publikovanih radova jasno se uočava trend rasta broja autora po radu: u prvim godinama ispitivanog perioda svi radovi su bili radovi jednog autora, prosečan broj autora po radu ima konstantan rast kako mreža evoluira da bi 2011. godine prosečan broj autora po radu bio 2.29. Konsekventno, postoji trend smanjenja broja radova sa jednim autorom: od 100% u 1932. godini do nešto ispod 40% u 2011. godini. U ovom pogledu, eLib mreža se ne razlikuje puno od mreže saradnje matematičara ekstrahovane iz MR (engl. *Mathematical Reviews*) bibliografske baze podataka [Grossman, 2002a].

Distribucija broja radova po autoru prati stepeni zakon (engl. *power-law*) sa bržim opadanjem frakcije autora koji imaju više od 25 radova nego što to predviđa opaženi stepeni zakon (engl. *truncated power-law, power-law with sharp cut-off*). Ovaj fenomen je takođe primećen i u mrežama saradnje istraživaća ekstrahovanih iz drugih digitalnih biblioteka [Newman, 2001c] i može se objasniti uvođenjem

vremenskih ograničenja u princip kumulativne prednosti [Amaral et al., 2000]. Drugim rečima produktivni autori teže da budu produktivniji što vodi stepenom zakonu, ali takođe autori posle izvesnog vremena prestaju sa publikovanjem radova (penzionišu se ili umiru) što vodi prekidu stepenog zakona (bržem opadanju) posle određene tačke. Uočeni fenomen implicira da većina autora poseduje produktivnost koja je jednaka ili približna prosečnoj, ali takođe postoje autori čiji je stepen produktivnosti drastično veći od proseka.

Upoređivanjem koeficijenta malog sveta i koeficijenta klasterisanja eLib mreže sa predikcijama dobijenih primenom modela slučajnog grafa je ustanovljeno da eLib mreža ispoljava osobinu malog sveta u Watts-Strogatz smislu. Mreža takođe pokazuje slabo asortativno vezivanje, sa indeskom asortativnosti koji iznosi 0.115. Skoro isti stepen asortativnog vezivanja pokazuje i mreža saradnje matematičara ekstrahovana iz MR (engl. *Mathematical Reviews*) bibliografske baze podataka [Newman, 2002]. Analiza poveznih komponenti je otkrila topološku raznovrsnost u strukturi eLib mreže:

- U odnosu na većinu mreža saradnje istraživača ispitivanu u literaturi (npr. [Barabasi et al., 2002; Bettencourt et al., 2009; Liu et al., 2005; Nascimento et al., 2003; Newman, 2001c; Perc, 2010]), i druge mreže saradnje matematičara [Batagelj and Mrvar, 2000; Brunson et al., 2014; Grossman, 2002a,b], eLib mreža ne sadrži gigantsku povezanu komponentu.

- Postoji relativno veliki broj komponenti koje se povinuju stepenom zakonu u pogledu veličine komponenti i broja radova koje su publikovali autori iz komponente.

- Većinu komponenti čine izolovani autori i male trivijalne komponente (trivijalne komponente su one koje ne evoluiraju), ali takođe postoji i mali broj relativno velikih i kompleksnih komponenti povezanih autora.

Slične karakteristike se mogu uočiti i za podmreže eLib mreže koje predstavljaju saradnju istraživača u individualnim časopisima koje eLib indeksira. Dodatno, na osnovu uporedne analize globalnih mrežnih metrika uočene su razlike između eLib časopisa u pogledu kohezivnosti zajednica njihovih autora, obrazaca asortativnosti i odnosa zastupljenosti muških i ženskih autora.

Upotrebom algoritama za detekciju zajednica [Blondel et al., 2008; Girvan and Newman, 2002; Pons and Latapy, 2006; Raghavan et al., 2007; Rosvall and Bergstrom, 2007] je pokazano da najveće eLib komponente poseduju jasnu strukturu zajednica. To znači da se u najvećim eLib komponentama mogy uočiti nepreklapajuće kohezivne grupacije autora. Dodatno, većina identifikovanih grupa su jaki klasteri u smislu da autor iz klastera tešnje sarađuje sa drugim autorima iz klastera nego sa drugim autorima.

Evolutivnom analizom eLib mreže je utvrđeno da postoji 6 karakterističnih perioda u njenoj evoluciji koje se odlikuju različitim intenzitetom i tipom kolaborativnog ponašanja eLib autora. U poslednja dva karakteristična perioda (od 1975. godine) intenzitet saradnje poseduje rastući trend i netrivijalne komponente evoluiraju na način da postaju veće i kohezivnije. To znači da se ne dešava samo fenomen inkluzije novih autora u netrivijalne komponente, nego i stari autori koji imaju zajedničke koautore uspostavljaju međusobnu saradnju. Stoga naši nalazi za matematičke časopise štampane u Srbiji su slični opservacijama publikovanim u [Brunson et al., 2014; Grossman, 2002a] gde je primećen definitivan trend ka povećanju saradnje između matematičara koji su svoje radove publikovali u časopisima koje indeksira "Mathematical Reviews".

U analizi eLib mreže kombinovane su metrike koje se koriste u analizi socijalnih mreža (metrike centralnosti, koeficijent klasterisanja i koeficijent malog sveta) sa metrikama produktivnosti (broj publikacija, dugoročna zastupljenost) kako bi se numerički predstavile karakteristike eLib autora.

Analizom metrika na nivou autora je ustanovljeno da je ugnježdenost autora (engl. *betweenness centrality*) bolji indikator produktivnosti i dugoročne zastupljenosti nego što je to stepen autora u mreži (broj najbližih suseda, odnosno koautora). Povrh toga, evolutivna studija korelacija između metrika na nivou autora je pokazala da jačina korelacija između ugnježdenosti i metrika produktivnosti raste kako mreža evoluira sugerišući da se čak i jače korelacije mogu očekivati u budućnosti.

U pogledu analize softverskih i ontoloških mreža fokus disertacije je bio na njihovoj strukturi. U budućem radu planiramo da ispitujemo njihovu evoluciju. Analiza evolucije jako povezanih komponenti i habova softverskih i ontoloških mreža može pružiti vredne uvide relevatne za održavanje softverskih i ontoloških sistema. Takođe ćemo ispitivati kako evolucija softverskih i ontoloških sistema utiče na strukturu mreža koje ih reprezentuju na različitim nivoima apstrakcije kako bi identifikovali evolutivne obrasce koji mogu voditi prediktivnim modelima evolucije softverskih i ontoloških sistema. U pogledu mreža saradnje istraživača planiramo studije koje se bave pokrivenošću, pozicijom i važnošću srpskih istraživača u mrežama saradnje ekstrahovanih iz masivnih, javno dostupnih bibliografskih baza podataka. Dodatno, planirane studije će obuhvatiti i mreže citata kako bi se ocenio uticaj srpskih istraživača u savremenoj nauci.

Kratka biografija kandidata

Miloš Savić je rođen u Sarajevu 12. oktobra 1984. godine. Osnovnu školu "Miša Dudić" u Valjevu je završio kao đak generacije, nakon čega upisuje Valjevsku gimnaziju, specijalno-matematičko odeljenje. Osnovne studije informatike Prirodno-matematičkog fakulteta, Univerziteta u Novom Sadu završava 2010. godine sa prosečnom ocenom 8.55 i diplomskim radom na temu "Struktura i evolucija kompleksnih mreža ekstrahovanih iz softverskih sistema". Master studije informatike na istom fakultetu završava 2011. godine sa prosečnom ocenom 9.0 i master radom na temu "Analiza i modelovanje softverskih mreža". Nakon toga upisuje doktorske studije informatike na istom fakultetu. Sve ispite predviđene planom i programom studija je položio sa prosečnom ocenom 10.0. Od 2012. godine je zaposlen na Prirodno-matematičkom fakultetu, Univerziteta u Novom Sadu, prvo kao istraživač pripravnik, a potom kao asistent na Katedri za računarske nauke. Drži ili je držao vežbe iz više predmeta za studente informatike: Uvod u programiranje, Strukture podataka i algormiti I, Objektno-orjentisano programiranje, Programski jezici, Veštačka inteligencija II i Inženjerstvo zahteva. Učesnik je na projektu "Inteligentne tehnike i njihova integracija u sisteme za podršku odlučivanju sa širokim poljem primene" koje finansira Ministarstvo obrazovanja, nauke i tehnološkog razvoja Republike Srbije. Bio je član organizacionih odbora tri međunarodne radionice. Pomoćnik je glavnog urednika ComSIS (Computer Science and Information Systems) časopisa. Saradnik je na programima računarstva Istraživačke stanice Petnica. Dobitnik je fakultetske nagrade "Aleksandar Saša Popović" za izuzetan istraživački rad iz informatike za 2011. godinu. Koautor je 14 radova publikovanih u međunarodnim časopisima i zbornicima međunarodnih konferencija u oblastima softverskog inženjerstva, analize kompleksnih mreža i elektronskog učenja.

Univerzitet u Novom Sadu
Prirodno-matematički fakultet
Ključna dokumentacijska informacija

Redni broj: RBR	
Identifikacioni broj: IBR	
Tip dokumentacije: TD	Monografska dokumentacija
Tip zapisa: TZ	Tekstualni štampani materijal
Vrsta rada: VR	Doktorska disertacija
Autor: AU	Miloš Savić
Mentor: MN	dr Mirjana Ivanović
Naslov rada: NR	Ekstrakcija i analiza kompleksnih mreža iz različitih domena
Jezik publikacije: JP	engleski
Jezik izvoda: JI	srpski/engleski
Zemlja publikovanja: ZP	Srbija
Uže geografsko područje: UGP	Vojvodina
Godina: GO	2015
Izdavač: IZ	autorski reprint
Mesto i adresa: MA	Novi Sad, Trg D. Obradovića 4
Fizički opis rada: (broj poglavlja/strana/lit. citata/tabela/slika/grafika/priloga) FO	6/215/316/56/47/0/1
Naučna oblast: NO	Informatika
Naučna disciplina: ND	Računarske nauke
Predmetna odrednica/ Ključne reči: PO	Kompleksne mreže, ekstrakcija, analiza, softverske mreže, ontološke mreže, mreže saradnje istraživača
UDK	
Čuva se: ČU	

Važna napomena:
VN

Izvod: Skoro svaki kompleksan sistem se može predstaviti mrežom koja opisuje interakcije izmedju entiteta od kojih je sistem komponovan. Fokus ove disertacije je na kompleksnim mrežama iz tri domena: (1) mreže ekstrahovane iz izvornog koda računarskih programa koje reprezentuju dizajn softverskih sistema, (2) mreže ekstrahovane iz ontologija semantičkog web-a koje opisuju strukturu deljenog znanja pogodnog za višekratnu upotrebu, i (3) mreže ekstrahovane iz bibliografskih zapisa koje opisuju saradnju istraživača. U okviru disertacije predložene su nove metode za ekstrakciju mreža iz pomenutih domena. Drugo, na nekoliko studija slučaja ilustrovani su benefiti mrežno orjentisane analize konkretnih sistema iz domena obuhvaćenih disertacijom. U poredjenju sa prethodnim relevantim istraživanjima, analize prezentovane u disertaciji nisu čisto topološke, nego kombinuju tehnike i metrike razvijene u okviru teorije kompleksnih mreža sa metrikama iz konkretnog domena.

IZ
Datum prihvatanja teme od strane
NN veća: 4. septembar 2014.
DP
Datum odbrane:
DO
Članovi komisije:
(Naučni stepen/ime i prezime/zvanje/fakultet)
KO

Predsednik:	dr Zoran Budimac, redovni profesor, Prirodno-matematički fakultet, Univerzitet u Novom Sadu
Mentor:	dr Mirjana Ivanović, redovni profesor, Prirodno-matematički fakultet, Univerzitet u Novom Sadu
Član:	dr Miloš Radovanović, docent, Prirodno-matematički fakultet, Univerzitet u Novom Sadu
Član:	dr Bojana Dimić Surla, docent, Prirodno-matematički fakultet, Univerzitet u Novom Sadu
Član:	dr Zoran Ognjanović, naučni savetnik, Matematički institut Srpske akademije nauka i umetnosti, Beograd

University of Novi Sad
Faculty of Science
Key Words Documentation

Accession number: NO	
Identification number: INO	
Document type: DT	Monograph documentation
Type of record: TR	Textual printed material
Contents code: CC	Doctoral dissertation
Author: AU	Miloš Savić
Advisor: MN	dr Mirjana Ivanović
Title: TI	Extraction and analysis of complex networks from different domains
Language of text: LT	English
Language of abstract LA	Serbian/English
Country of publication: CP	Serbia
Locality of publication: LP	Vojvodina
Publication year: PY	2010
Publisher: PU	Author's reprint
Publ. place: PP	Novi Sad, Trg D. Obradovića 4
Physical description: (no. of chapters/pages/bib. refs/tables/figures/graphs/appendices) PO	6/215/316/56/47/0/1
Scientific field: SF	Informatics
Scientific discipline: SD	Computer Science
Subject/Key words: SKW	Complex networks, extraction, analysis, software networks, ontology networks, co-authorship networks
UC	
Holding data: HD	

Note:
N
Abstract: Almost any large-scale system can be viewed as a network that shows interactions among entities which are constituent parts of the system. The focus of this dissertation is on complex networks from three domains: (1) networks extracted from source code of computer programs that represent design of software systems, (2) networks extracted from semantic web ontologies that describe the structure of shared and reusable knowledge, and (3) networks extracted from bibliographic records that depict collaboration in science. We proposed new methods for the extraction of networks from mentioned domains. Secondly, on several case studies we demonstrated benefits of network-based analysis of concrete systems from those domains. In contrast to the previous work on the subject, analyses presented in this dissertation are not purely topological, but combine techniques and metrics developed under the framework of complex network theory with domain-dependent metrics.
AB
Accepted by Scientific Board on: September 4, 2014
AS
Defended:
DE
Dissertation Defense Board:
(Degree/first and last name/title/faculty)
DB
President: Dr. Zoran Budimac, full professor, Faculty of Sciences, University of Novi Sad

Advisor: Dr. Mirjana Ivanović, full professor, Faculty of Sciences, University of Novi Sad

Member: Dr. Miloš Radovanović, assistant professor, Faculty of Sciences, University of Novi Sad

Member: Dr. Bojana Dimić Surla, assistant professor, Faculty of Sciences, University of Novi Sad

Member: Dr. Zoran Oganjanović, research professor, Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade