

Scale-Free Nature of Java Software Package, Class and Method Collaboration Graphs

David Hyland-Wood
School of Information Technology
and Electrical Engineering
The University of Queensland
Brisbane, Australia 4072
+1 571 331 3723
dwood@itee.uq.edu.au

David Carrington
School of Information Technology
and Electrical Engineering
The University of Queensland
Brisbane, Australia 4072
+61 7 3365 3310
davec@itee.uq.edu.au

Simon Kaplan
Information Technology
Queensland University of Technology
Brisbane, Australia 4001
+61 7 3864 1913
s.kaplan@qut.edu.au

ABSTRACT

Software collaboration graphs for two Open Source software projects written in the Java programming language, the Kowari Metastore and JRDF, were analyzed for fifteen-month periods of development. Collaboration graphs were produced at the package, class and method levels.

The collaboration graphs were found to form networks which exhibited approximately scale-free properties at all three levels and during each period analyzed. This finding tends to support claims made by others that software class collaboration graphs approximate scale-free networks and provides new insights that they seem to retain the scale-free properties across different levels of granularity and over the course of their development life cycle. Significant differences between the magnitudes of power law exponents identified and those found in C and C++ applications by previous researchers are noted and discussed. Specifically, only the method-level collaboration graphs were found to have power law exponents in the most common range for scale-free networks ($2 < \gamma < 3$).

Categories and Subject Descriptors

D.2.8 [Metrics]: Complexity measures, performance measures, software science.

General Terms

Algorithms, Measurement, Theory.

Keywords

Scale-free, power law, Java, collaboration graphs.

1. INTRODUCTION

Many sets of relationships may be represented by the mathematical concept of a network. Interestingly, many of these networks may be closely approximated by a power law distribution, where the probability, $P(k)$, of a particular node

having a certain number of connections, k , decays with the power law $P(k) \sim k^{-\gamma}$, where generally $2 < \gamma < 3$. This type of network is known as scale-free. Scale free properties have been reported in networks as seemingly diverse as biological metabolisms [8] and the Internet [5].

The internal structures of software programs, such as class collaboration graphs for object-oriented systems or call graphs for procedural code, have also been declared to exhibit scale-free properties. Myers [12], for example, studied six Open Source software projects, three of which were procedural applications written in C and three of which were object-oriented applications written in C++. In all six cases, he showed that the networks formed by their class collaboration or call graphs showed approximate scale-free properties. Nearly simultaneously, de Moura et al. [2] reported identifying scale-free topologies in four other C and C++ applications.

Researchers at the Santa Fe Institute produced several papers exploring this phenomenon in 2002 and 2003. They noted the similarities between natural and engineered systems [13], claimed that these properties arose through a process of optimization [14] and later decided that "the final outcome of software evolution", regardless of design, methodology or features, is, among other things, a network with scale-free properties.

The scale-free nature of software networks transcends the work of individual teams [11]. The analysis of inter-package dependencies again found scale-free properties.

Many of these researchers [2, 11, 14 and 15] have also claimed that the software networks they examined exhibited "small-world" properties, meaning that the average distance (or "diameter") between nodes was small. Although researchers have not yet agreed on a strict definition of small-worldness, Cohen and Havlin [1] have noted that the diameter scales proportionally to the natural logarithm of the number of nodes. They have found that scale-free networks with $2 < \gamma < 3$ have a significantly smaller diameter ($d \sim \ln \ln N$), and thus consider them "ultra small-world".

This research examined temporal snapshots of source code for two Open Source, object-oriented systems written in the Java programming language to analyze the state of their software collaboration graphs over time.

Open Source software projects often store their source code in Internet-accessible revision control systems, such as the Concurrent Versioning System (CVS) or Subversion. Public access to these repositories allow researchers to mine large

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

The 5th International Symposium on Empirical Software Engineering, September 21-22, 2005, Rio de Janeiro, Brazil.

Copyright 2006 ACM 1-58113-000-0/00/0004...\$5.00.

volumes of development history in attempts to discover relationships in the data. Two such repositories were analyzed, the Kowari Metastore and JRDF.

The Kowari Metastore is an Open Source, scalable, transaction-safe database for the storage, retrieval and analysis of Resource Description Framework (RDF) and Web Ontology Language (OWL) metadata. Kowari is written in the Java programming language and has been released under the Mozilla Public License, version 1.1. Kowari is described in [16] and hosted at [10].

JRDF is a Java framework for storing, querying and creating RDF statements. The project is dual-licensed under either the Apache Software License or the GNU Lesser General Public License (LGPL). JRDF is hosted at [9].

Each project's revision control system was used to generate snapshots of the project at monthly intervals over fifteen months. Snapshots were taken on the last day of each month. Kowari snapshots were taken from 31 October 2003 to 31 December 2004. The earlier date corresponded with the first availability of the project under an Open Source license with a publicly accessible source code repository. JRDF snapshots were taken from 31 December 2003 to 28 February 2005 and represent the first fifteen months of project development. Both projects produced and released stable versions during the fifteen months analyzed.

The two projects have substantial similarities and differences. Both are written in Java. JRDF's authors also contributed to Kowari. However, Kowari is a large database (over two hundred thousand lines of code) and JRDF is a small library (less than twenty thousand lines of code).

An earlier report of our work was provided in [7] and was limited to class collaboration graphs. Kowari and JRDF class collaboration graphs for the same fifteen-month periods of development used in this paper were found to exhibit rough scale-free properties at every temporal snapshot studied. However, the power-law exponents (γ) for Kowari and JRDF were significantly lower than the exponents found by Myers and de Moura. In fact, the exponents for the data overall ($1.6 < \gamma \sim 1.8$) did not fall into the range generally accepted for scale-free networks ($2 < \gamma > 3$).

This paper further analyzes the collaboration graphs formed in Kowari and JRDF at the package, class and method levels and draws some additional conclusions.

2. ANALYSIS

The process used to collect, refine and analyze data was intentionally similar to that used by Myers. Fifteen snapshots of the Kowari and JRDF source code trees were collected by performing checkouts from the projects' CVS repositories hosted at Sourceforge.net. CVS snapshots were obtained for the last day of each month.

The Doxygen documentation generator [4] was run over each CVS snapshot. Doxygen was configured to generate collaboration graphs for individual methods in GraphViz [6] Dot format [3]. For each CVS snapshot, the collaboration graphs were then collected into a single, composite GraphViz file, with the names for each graph node changed to be unique within the file. Only references to methods in the Kowari and JRDF code bases were retained; references to packages, classes and third-party components were removed. A single file in GraphViz format was

thus produced which consisted of directionalized relations between methods for each temporal snapshot of each project.

Each method-level GraphViz file was then passed through a Perl script that compared node (method) names to a sorted list of class names. A list of classes was obtained from the Doxygen output for each CVS snapshot and that data used to collapse the method-level graphs appropriately. Unfortunately, a bug in Doxygen resulted in some errors in the generated lists of classes. Some package names were listed in the class list. The lists were reviewed and cleaned prior to use. A GraphViz file was then produced that provided a class-level collaboration graph representation for the given date. This process was repeated for each temporal snapshot of each project.

Similarly, the method-level GraphViz file was passed through the same perl script (with different parameters) to create collaboration graphs at the package level. The same Doxygen bug was encountered which necessitated a review of the package names used to collapse the method names appropriately.

Each relation within the final package, class and method-level collaboration graphs thus consisted of a unidirectional connection between nodes. Connections for inheritance and association references were also available from Doxygen and were hence represented in the final collaboration graphs at the package and class levels.

Additional Perl scripts were written to collect statistics regarding the collaboration graphs at all levels. The number of graph nodes and their cumulative frequency distributions were collected and output into files suitable for importing into Microsoft Excel for further analysis. Log-log graphs of the unnormalized cumulative in-degree and out-degree frequency distributions vs. the number of in-degree and out-degree connections at each level for each CVS snapshot were produced using Gnuplot.

Analysis of each graph showed clearly that distributions were approximately linear at each temporal snapshot taken. Method degree distributions were found to be generally more linear over the entire data range than distributions at the package and class levels. Kowari's method in-degree r^2 value averaged 0.99 and out-degree r^2 value averaged 0.94 over all temporal snapshots where the values were computable (see the comments regarding early JRDF data below).

By comparison, Kowari's class in-degree r^2 value averaged 0.926 and out-degree r^2 averaged 0.890 over all snapshots taken. Both JRDF's class in-degree and out-degree r^2 values averaged 0.948 over all snapshots taken. Package-level r^2 values were 0.90 for Kowari's in-degree and 0.86 for out-degree. JRDF's package in-degree was 0.97 and out-degree was 0.95.

Each graph was evaluated individually and a majority subset of data chosen which more closely approximated linearity wherever the overall r^2 values averaged less than 0.95. Linear ranges with r^2 values over 0.95 were generally readily obtained, as discussed in more detail below.

JRDF is a small project and the fifteen-month time span analyzed included the early days of its development when it had few packages and classes. Thus, some allowances had to be made in data collection and analysis. JRDF classes were collected in a single package until the seventh month of development (June, 2004) and in only two packages until the twelfth month of

development (November 2004). This resulted in a complete lack of usable package-level data for the first six months of the project and data of questionable accuracy (two points always result in a straight line!) for the following five months. JRDF package-level averages presented here therefore start in the seventh month and should be treated as less than statistically significant. The fact that they are generally in line with Kowari’s package-level data for the same time periods is comforting, but should not be taken as authoritative. The results were averaged inclusive of data starting in the seventh month, but no attempt was made to calculate “more linear ranges” for those dates which had fewer than six data points. This decision meant that additional linear ranges were not calculated for JRDF package-level in-degree distributions at all.

JRDF class-level data was also initially sparse for the first three months of development (December 2003 to February 2004). Again, the small number of data points resulted in highly linear interpolations which may not be statistically significant. The results were averaged inclusive of this early data, but no attempt was made to calculate “more linear ranges” for those dates.

The average power law exponents (γ) for Kowari and JRDF unnormalized degree distributions were found to be significantly lower than those found by Myers and de Moura et al. across all snapshots taken at the package and class levels. This is an indication that nodes (representing packages and classes) in Kowari’s and JRDF’s Java software collaboration graphs were significantly less likely to connect to other nodes than the nodes in the C and C++ graphs¹ analyzed by Myers and de Moura. Only the method levels revealed power law exponents in the ranges discussed by those researchers.

Incoming and outgoing connections were graphed separately for each project. Power law exponents were calculated for both the entire data set and for a more nearly linear subset. “More linear ranges” were obtained where necessary by reducing the data set from the extreme ends (smallest and largest k) until a range was obtained which, where possible, was linear enough for an r^2 fit of greater than 0.95. Where such a fit could not be found, a more linear range was calculated with the best possible linear fit without deviating from the above procedure.

Table 1 summarizes the average power law exponents found for package-level collaboration graphs for both projects over all temporal snapshots and provides an indication of their linear curve fits. Average power-law exponents for both the entire data range and the more linear ranges (where calculated) are also provided. Tables 2 and 3 provide the same information for class-level and method-level collaboration graphs, respectively.

Minor deviations from the procedure described above are noted in footnotes for the tables.

Figures 1 and 2 illustrate the differences found between the package, class and method levels in both Kowari and JRDF. Since the power law exponents did not change radically for any given distribution, average exponent values are plotted for each level.

¹ Note that de Moura et al. analyzed header files from C and C++ applications. Thus, they looked at C++ classes and C files, which are taken to be rough equivalents for our purposes.

Table 1. Linear Fit and Power Law Exponent Summary for Package Collaboration Graphs

Degree	Full Data Set		More Linear Range	
	Avg r^2	Avg γ	Avg r^2	Avg γ
Kowari In	0.9049	1.4795	0.9577	1.5041
Kowari Out	0.8621	1.5307	0.9396	1.5783
JRDF In	0.9721	1.4977	N/A ²	N/A ²
JRDF Out	0.9462	1.6034	0.9720 ³	2.211 ³

Table 2. Linear Fit and Power Law Exponent Summary for Class Collaboration Graphs

Degree	Full Data Set		More Linear Range	
	Avg r^2	Avg γ	Avg r^2	Avg γ
Kowari In	0.9256	1.6862	0.9606	1.6772
Kowari Out	0.8895	1.7970	0.9650	1.8570
JRDF In	0.9476	1.6008	0.9596	1.7605
JRDF Out	0.9480	1.6194	0.9708	1.8233

Table 3. Linear Fit and Power Law Exponent Summary for Method Collaboration Graphs

Degree	Full Data Set		More Linear Range	
	Avg r^2	Avg γ	Avg r^2	Avg γ
Kowari In	0.9888	2.2666	N/A ⁴	N/A ⁴
Kowari Out	0.9439	2.3458	0.9617	2.7027
JRDF In	0.9851	2.2229	N/A ⁴	N/A ⁴
JRDF Out	0.9211	2.3159	0.9672	2.7934

In both Kowari and JRDF, method-level collaboration graphs resulted in exponents in the accepted scale-free (and even “ultra small-world”) ranges than do the same graphs when viewed at the package or class levels. This is interesting, given the obvious differences in the scale, complexity, use and design of the two projects.

It is particularly interesting to note that the power-law exponents of both projects entered the commonly-accepted range for scale-free networks, $2 < \gamma < 3$, only at the method level. The fact that this feature was repeated for both in-degree and out-degree distributions for both projects suggests that the phenomenon does, in fact, exist.

² The small number of packages in JRDF’s early history made the calculation of more linear ranges for this region impractical.

³ Based on data from only nine out of fifteen months, due to the small number of JRDF packages early in its history.

⁴ Additional linear ranges were not sought for data sets with a least squares linear fit of greater than 0.95.

Figures 3, 4, 5 and 6 present representative unnormalized cumulative degree distributions for the method level of each project. Distributions are presented for the first, fifth, tenth and fifteenth month of each project history, for brevity. The method data was chosen for presentation since it is the most clearly representative of scale-free properties across the entire data range.

Package- and class-level unnormalized cumulative degree distributions for Kowari and JRDF look similar to Figures 3-6, with sparser information and slightly less linear structures, as represented in Tables 1 and 2.

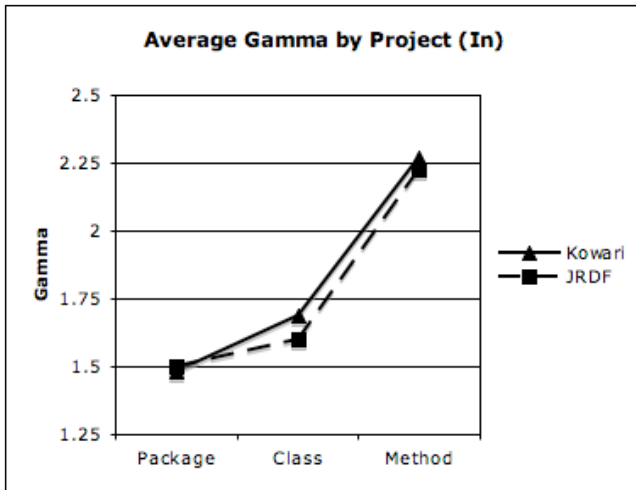


Figure 1. Average Power-Law Exponents by Project for In-Degree Distributions.

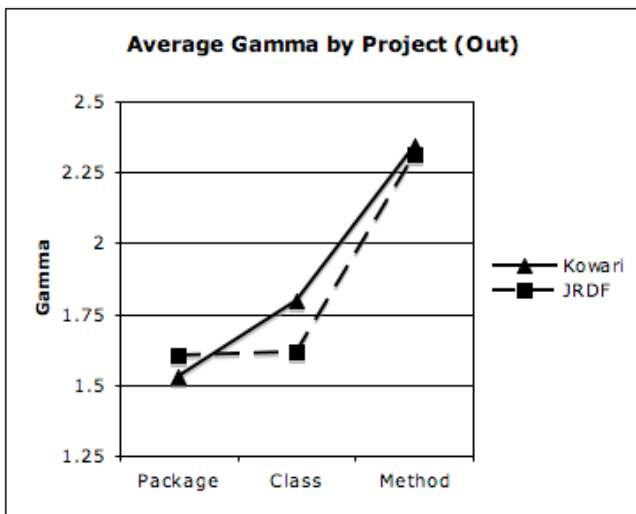


Figure 2. Average Power-Law Exponents by Project for Out-Degree Distributions.

Figure 3 shows the unnormalized cumulative in-degree distributions for Kowari methods. The average power law exponent was 2.27 for all data. No more linear range was calculated for this data because the linear fit was already very good (with an average r^2 exceeding 0.98). Exponents at time stamps varied from a low of 2.16 to a high of 2.41.

Figure 4 shows the out-degree distributions for Kowari methods. The average power law exponent was 2.36 (all data) and increased to 2.70 for the more linear ranges. Exponents at time stamps varied from a low of 2.25 to a high of 2.45 for the entire data set.

Figures 5 and 6 show the in- and out-degree distributions for JRDF methods. Exponents at the various time stamps varied from a low of 1.75 (for early, poorly populated distributions with only three differentiated values of k) to a high of 2.81. The average exponent value was 2.22 for in-degree distributions and 2.32 for out-degree distributions.

In some cases, such as the Kowari class out connections, smaller linear ranges could be found with steeper slopes, resulting in local power law exponents up to 2.76. An example of this is shown in Figure 7, Series 3, where the slope of the steepest linear range is -1.76 ($\gamma \sim 1 - m$). These smaller linear ranges were used as evidence by others that software projects generally contain some scale-free features in the general case, although it is clear that these small(ish) linear ranges are not representative of a project's overall collaboration graph.

Interestingly, as we looked at progressively lower levels, from packages to classes to methods, the collaboration graphs showed more clearly scale-free properties. It is yet to be determined if this finding will hold for other software projects.

It should be noted that both Myers and de Moura et al. analyzed software programs created in the C and C++ programming languages, whereas both programs analyzed in this research were written in Java. Myers created collaboration graphs for both object-oriented and procedural projects using Doxygen, and looked at class data or its procedural equivalents. deMoura, et al, used C/C++ header files to create their collaboration graphs and thus looked at the same (class or file) level.

We were able to confirm a number of the findings made by Myers. Specifically, the shape of the collaboration graphs at all levels were similar; they included significant linear ranges followed by a faster decay at large number of connections (k). Also, we determined, like Myers, that class collaboration graphs for Kowari and JRDF had power-law exponents that were larger in out-degree than in-degree. The package- and method-level collaboration graphs also exhibited this property.

Both Kowari and JRDF were developed using modern software methodologies. Kowari was created using a test-first variant of Extreme Programming, a so-called "agile" methodology. JRDF was created by a smaller team using a test-driven approach. Both are similar in impact on a software application. Both projects were written in Java, and so both were object-oriented. Both development teams attempted to hold to accepted object-oriented design and development styles. These factors lead to the creation of small blocks of code, with significant reuse of components. Continual refactoring during the periods studied reinforced these tendencies and resulted in higher in-degree distributions. In-degree distributions therefore showed a higher number of connections (k). This feature is evident in any particular time stamp in Tables 3 and 5 when compared to the same time stamp in Tables 4 and 6 for the same project.

Surprisingly, we did not see a great deal of difference between the collaboration graphs of Kowari and JRDF. In spite of the different designs, sizes and uses of the two projects, their collaboration graphs showed very similar properties. These

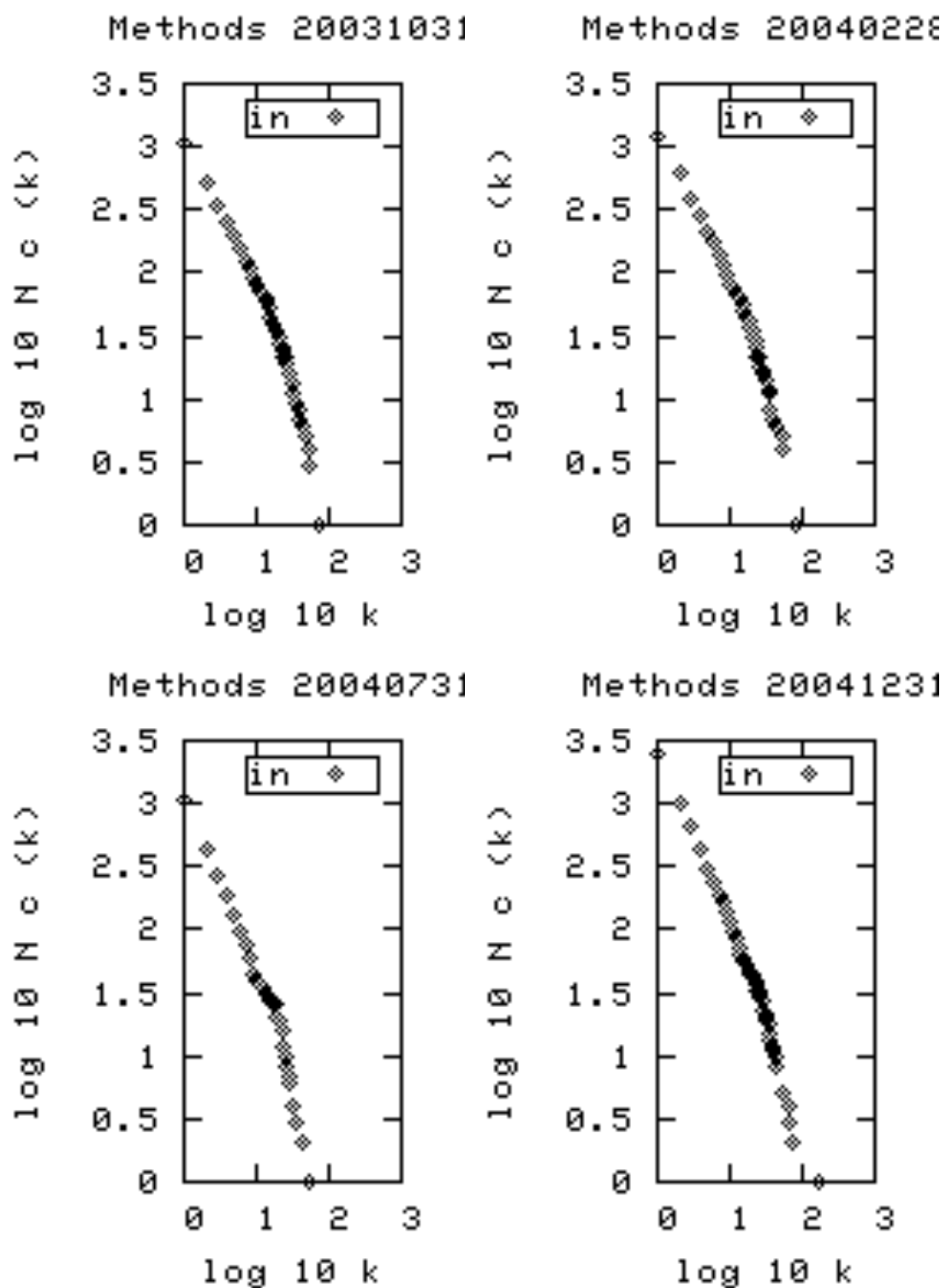


Figure 3. Representative Kowari In-Degree Method Distributions.

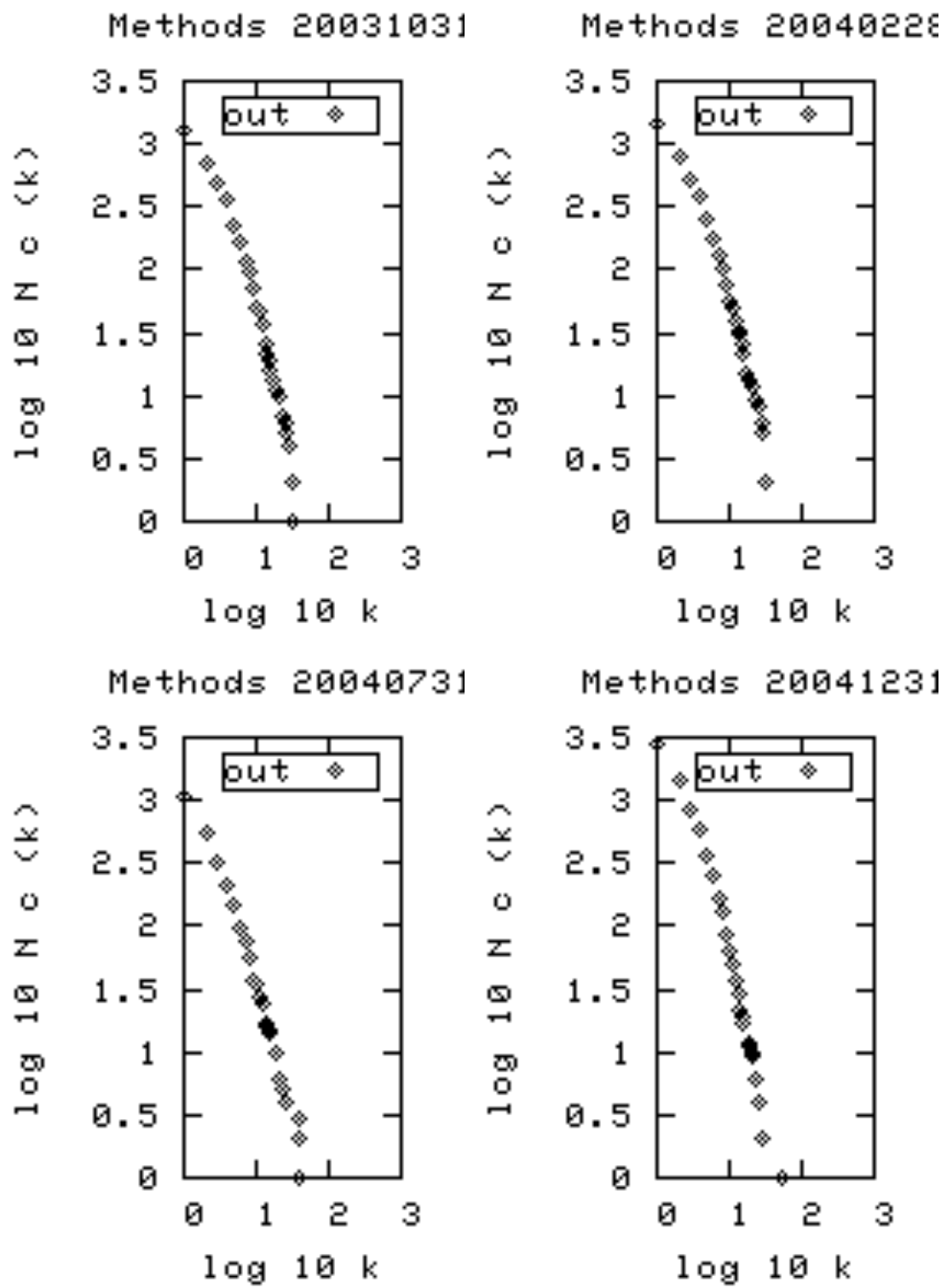


Figure 4. Representative Kowari Out-Degree Method Distributions.

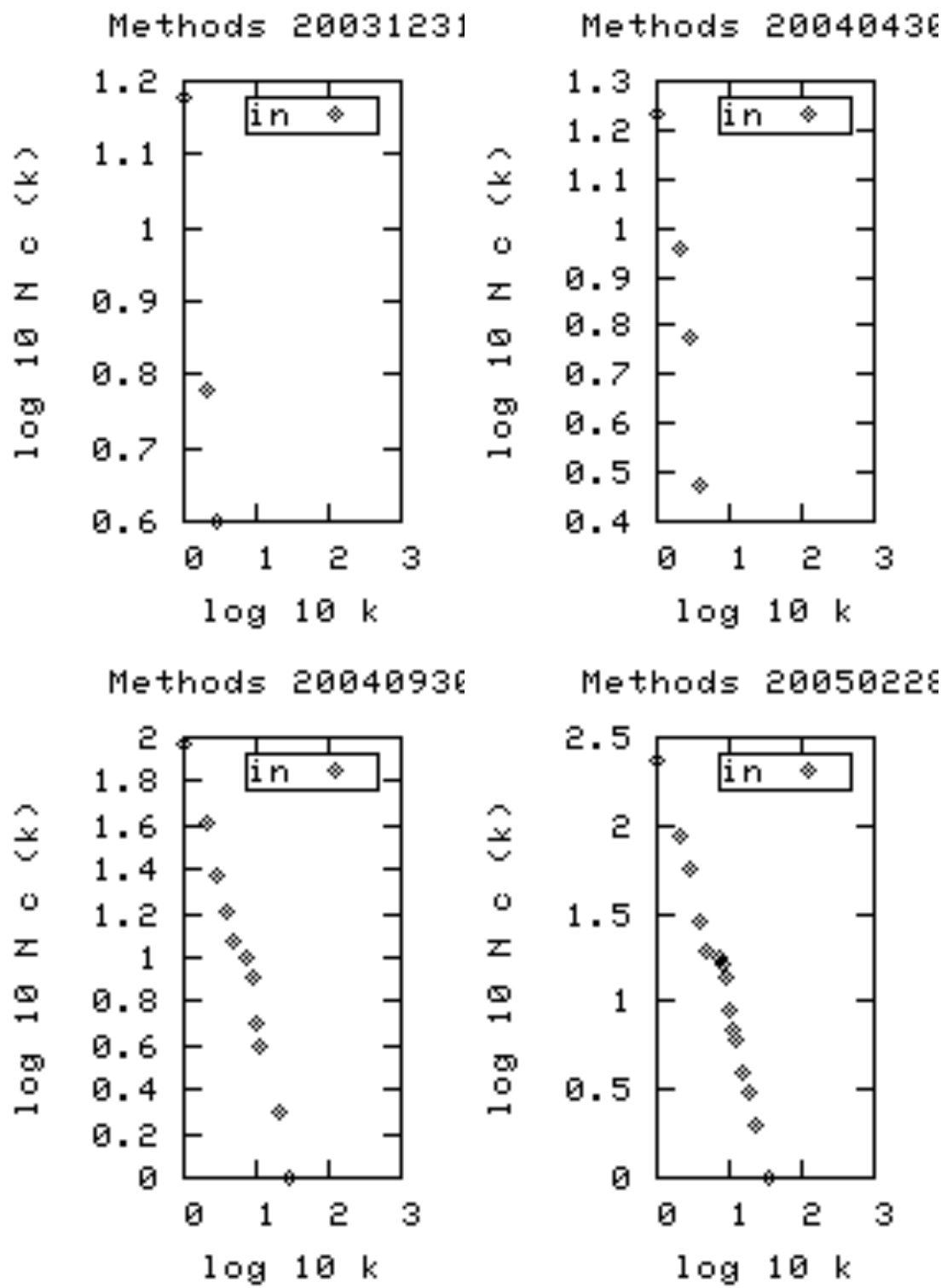


Figure 5. Representative JRDF In-Degree Method Distributions.

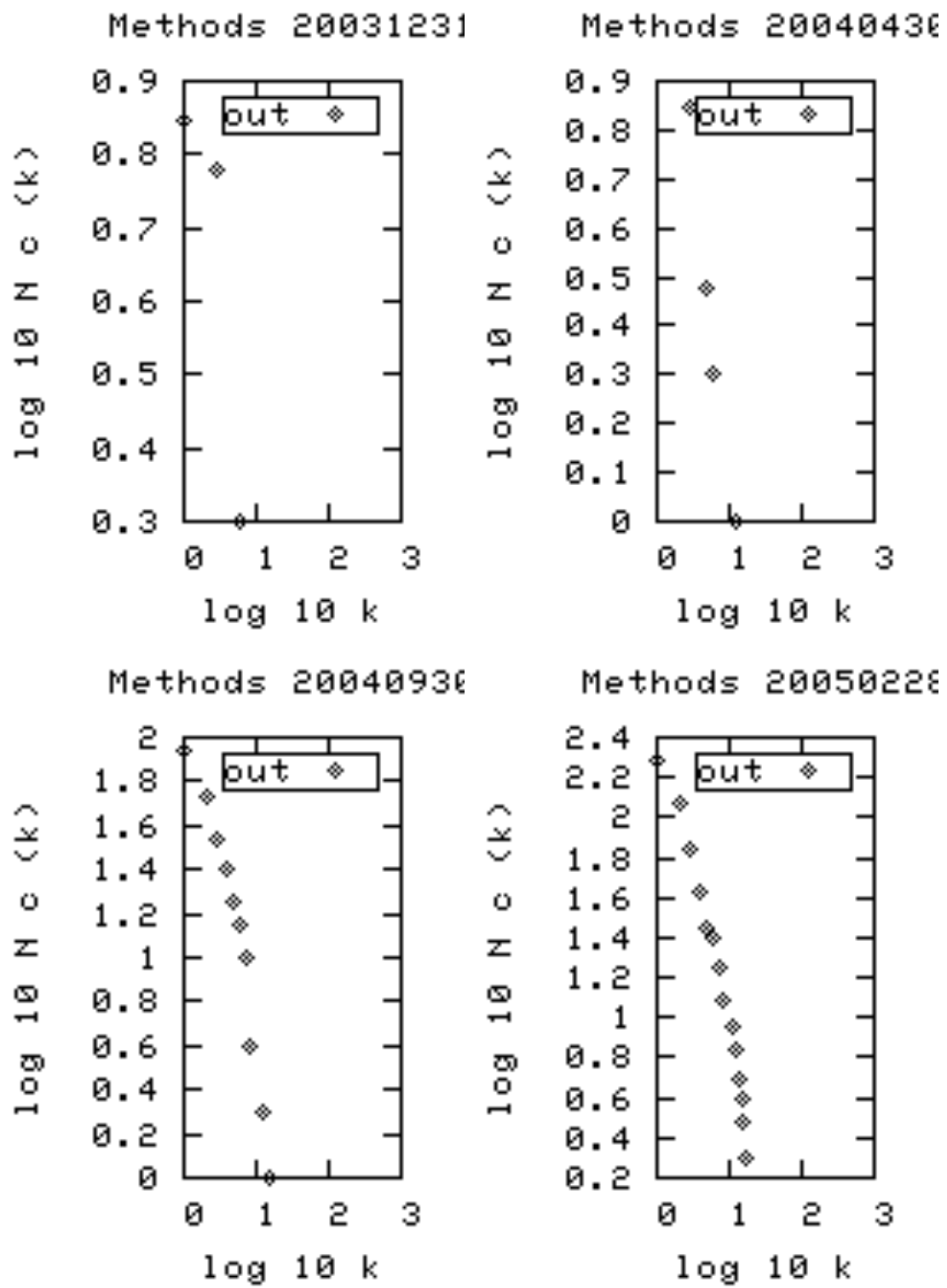


Figure 6. Representative JRDF Out-Degree Method Distributions.

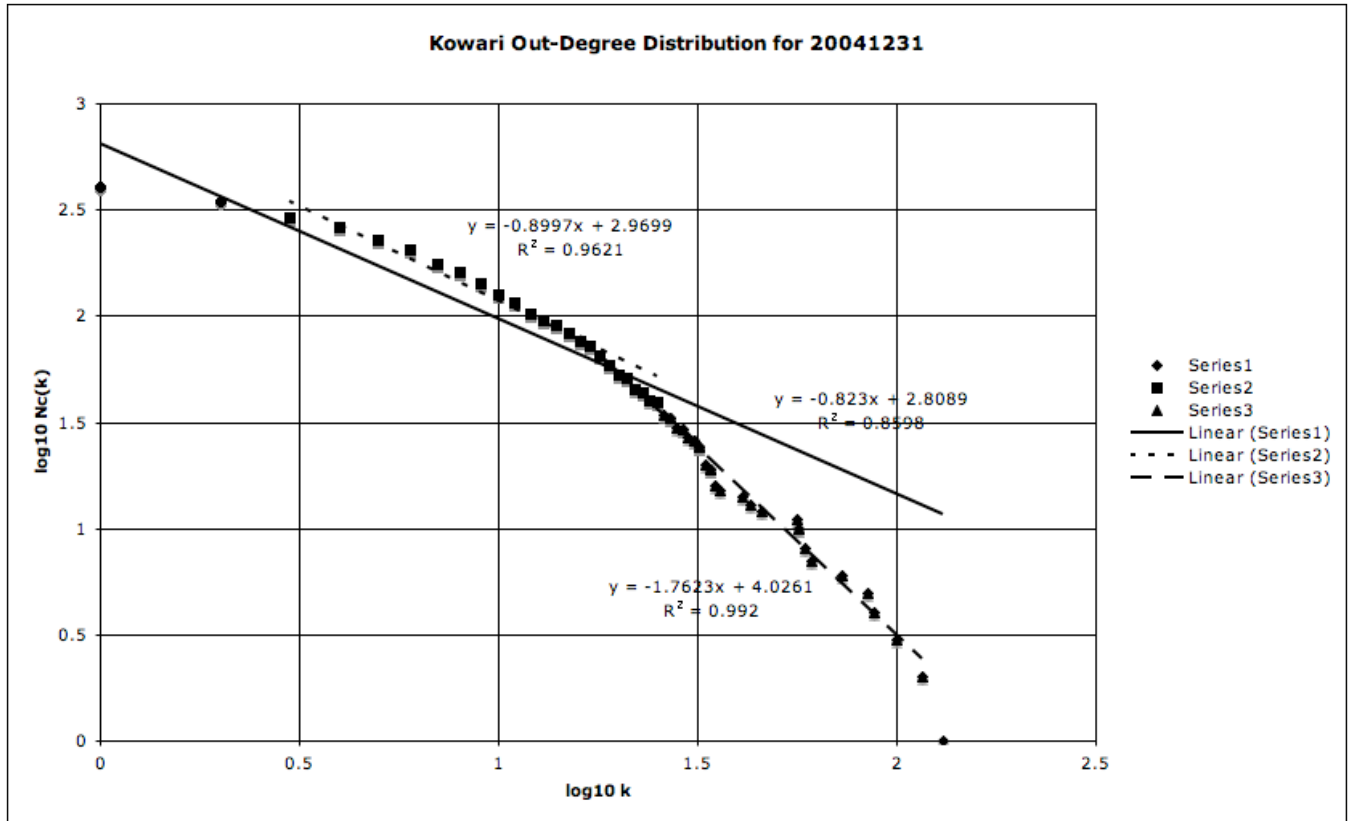


Figure 7. Kowari Out-Degree Class Distribution for 31 December 2004.

properties were also time-invariant. That is, the properties developed with the projects and stayed relatively constant throughout the lifecycle of both projects.

Since the Java projects we analyzed both showed class collaboration graphs with substantially lower power-law exponents than Myers' or de Moura's, we are confident that there is a systemic reason for such behavior. It is possible, although speculative at this time, that the smaller values for power law exponents observed at the class and package levels may be due to some combination of the language or the design style. If so, that finding could call into question the broader statements of scale-free applicability to general software structures claimed by Valverde and Solé [15].

3. DISCUSSION AND CONCLUSIONS

Package, class and method-level collaboration graphs for the Kowari Metastore and JRDF projects form networks with nearly scale-free properties at each level and snapshot in time analyzed. This finding tends to support the claims made by Myers and Valverde et al. that relationships between software objects form networks with scale-free properties. It should be highlighted that these properties are only approximately scale-free in most cases. Although linear ranges may be found, the overall network structure fails to meet to the scale-free definition in the general case when a linear approximation is applied overall.

Method-level collaboration graphs did unquestionably form scale-free networks overall, being both highly linear and demonstrating power-law exponents solidly in the accepted

range of $2 < \gamma < 3$. The method-level in-degree distributions were especially linear for both projects, with r^2 values in excess of 0.98 over the entire range.

Previous studies have focused on relatively small linear ranges in class collaboration graphs. Our work extended that approach to both higher (package) and lower (method) levels for the two Java-based projects studied.

All distributions studied here included significant linear ranges followed by a faster decay at large number of connections (k). This confirms Myers' finding of the same phenomenon. We were also able to confirm Myers' finding that class collaboration graphs exhibited power-law exponents that were larger in out-degree than in-degree and extend this finding to the package and method levels.

In-degree distributions were generally more linear than out-degree distributions, and tend toward a higher number of connections (k). This is perhaps expected, since the software development practices used for the development of both Kowari and JRDF encouraged the creation of small blocks of code (classes and methods) and significant reuse of those blocks within an application.

Package, class and method collaboration graphs were found to exhibit the same general properties throughout significant periods of development. The properties did not appear to be impacted by scale of the software project or by maturity of development for the systems and time periods studied. Thus, the collaboration graphs seem to retain their approximately scale-free properties over the course of their development life cycle.

4. FURTHER WORK

Validation of this work for a larger variety of software projects is considered necessary. This work only addressed two software projects, both of which were object-oriented and written in the same language. No work was done to validate that projects written in other languages or styles exhibit scale-free properties over lengthy periods of development. Further analysis could be categorized by software type (e.g. procedural, object-oriented, functional) and language.

It is possible that collaboration graphs contain properties which may relate a software project to its development and maintenance strategies more completely than the pure phenomenology discussed here. Further work by the authors will explore the topological properties of software collaboration graphs in order to map these relationships. It may be possible to both relate these properties to both existing software metrics and to develop new metrics to assist software developers and maintainers.

5. ACKNOWLEDGMENTS

The authors would like to thank the developers of the Kowari Metastore and JRDF for the production and release of their applications under Open Source licenses. Other software which contributed to this effort included the Open Source projects Doxygen, CVS and Perl and the freely available Gnuplot.

Chris Diehl of Johns Hopkins University's Advanced Physics Laboratory deserves thanks for useful discussions during the preparation of this paper.

David Hyland-Wood's efforts were partially funded by the University of Maryland Information and Network Dynamics (MIND) Laboratory, in turn funded by Fujitsu Laboratory of America College Park, NTT Corporation and Lockheed Martin Corporation.

6. REFERENCES

- [1] Cohen, R. and Havlin, S., Scale-Free Networks are Ultrasmall, *Physical Review Letters*, Vol. 90, 058701, 2003.
- [2] de Moura A.P., Lai Y.C., Motter A.E., Signatures of small-world and scale-free properties in large computer programs, *Physical Review E* 68, 017102 July 2003.

- [3] Dot format. <http://www.graphviz.org/cvs/doc/info/lang.html>
- [4] Doxygen. <http://www.stack.nl/~dimitri/doxygen/>
- [5] Faloutsos, M., Faloutsos, P. and Faloutsos, C., On Power-Law Relationships of the Internet Topology, *ACM SIGCOMM Computer Communication Review*, Proceedings of ACM SIGCOMM 1999, Vol. 29, pg. 251, 1999.
- [6] GraphViz. <http://www.graphviz.org>
- [7] Hyland-Wood, D., Carrington, D. and Kaplan, S., Scale-Free Nature of Java Software Class Collaboration Graphs, submitted to Mining Software Repositories workshop at the International Conference on Software Engineering 2006.
- [8] Jeong, H., Tombor, B., Albert, R., Oltvai, Z.N. & Barabási, A.-L. The large-scale organization of metabolic networks. *Nature* Vol. 407, pg. 651-654, 2000.
- [9] JRDF, <http://jrdf.sourceforge.net/>
- [10] Kowari Metastore, <http://kowari.org/>
- [11] LaBelle, N., Wallingford, E., Inter-Package Dependency Networks in Open-Source Software, Submitted to *Journal of Theoretical Computer Science*, 2004.
- [12] Myers, C.R., 2003, Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs, *Physical Review E* 68, 046116 2003.
- [13] Solé, R., Cancho, R.S., Montoya, J.M. and Valverde, S., Selection, Tinkering, and Emergence in Complex Networks, *Complexity*, 8(1) (2002) 20-33.
- [14] Valverde, S., Ferrer Cancho, R. and Solé, R.V., Scale-free networks from optimal design, *Europhys. Lett.*, 60 (4), pp. 512-517, 2002.
- [15] Valverde, S., Solé, R., Hierarchical Small Worlds in Software Architecture, Santa Fe institute Working Paper, 03-07-044, 2003, <http://www.santafe.edu/research/publications/wpabstract/200307044>.
- [16] Wood, D., Gearon, P., Adams, T. Kowari: A Platform for Semantic Web Storage and Analysis, *Proc. of XTech 2005*, <http://www.idealliance.org/proceedings/xtech05/papers/04-02-04/>