Scale-free Networks from Optimal Design

S. Valverde[1], R. Ferrer Cancho[1] and R. V. Solé[1,2]

[1] *ICREA-Complex Systems Lab, IMIM-UPF, Dr Aiguader 80, Barcelona 08003, SPAIN*
[2] *Santa Fe Institute, 1399 Hyde Park Road, New Mexico 87501, USA.*

PACS. 05.10.-a – Computational methods in statistical physics.
PACS. 05.65.+b – Self-organizing systems.

Abstract. – A large number of complex networks, both natural and artificial, share the presence of highly heterogeneous, scale-free degree distributions. A few mechanisms for the emergence of such patterns have been suggested, optimization not being one of them. In this letter we present the first evidence for the emergence of scaling (and smallwordlness) in software architecture graphs from a well-defined local optimization process. Although the rules that define the strategies involved in software engineering should lead to a tree-like structure, the final net is scale-free, perhaps reflecting the presence of conflicting constraints unavoidable in a multidimensional optimization process. The consequences for other complex networks are outlined.

Two basic features common to many complex networks, from the Internet to metabolic nets, are their scale-free (SF) topology [1] and a small-world (SW) structure [2, 3]. The first states that the proportion of nodes $P(k)$ having k links decays as a power law $P(k) \sim k^{-\gamma}\phi(k/\xi)$ (with $\gamma \approx 2-3$) [1, 4, 5] (here $\phi(k/\xi)$ introduces a cut-off at some characteristic scale ξ). Examples of SF nets include Internet topology [4,6], cellular networks [7,8], scientific collaborations [9] and [10] lexical networks. The second refers to a web exhibiting very small average path lengths between nodes along with a large clustering [2, 3].

Although it has been suggested that these nets originate from preferential attachment [4], the success of theoretical approximations to branching nets from optimization theory [11, 12] would support optimality as an alternative scenario. In this context, it has been shown that minimization of both vertex-vertex distance and link length (*i.e.* Euclidean distance between vertices) [13] can lead to the SW phenomenon. In a similar context, SF networks have been shown to originate from a simultaneous minimization of link density and path distance [14]. Optimal wiring has also been proposed within the context of neural maps [15]: 'save wiring' is an organizing principle of brain structure. However, although the analysis of functional connectivity in the cerebral cortex has shown evidence for SW [16], the degree distribution is clearly non-skewed but single-scaled (i. e. ξ is very small).

The origin of highly heterogeneous nets is particularly important since it has been shown that these networks are extremely resilient under random failure: removal of randomly chosen nodes (tipically displaying low degree) seldom alters the fitness of the net [17]. However, when nodes are removed by sequentially eliminating those with higher degree, the system rapidly experiences network fragmentation [17, 18].

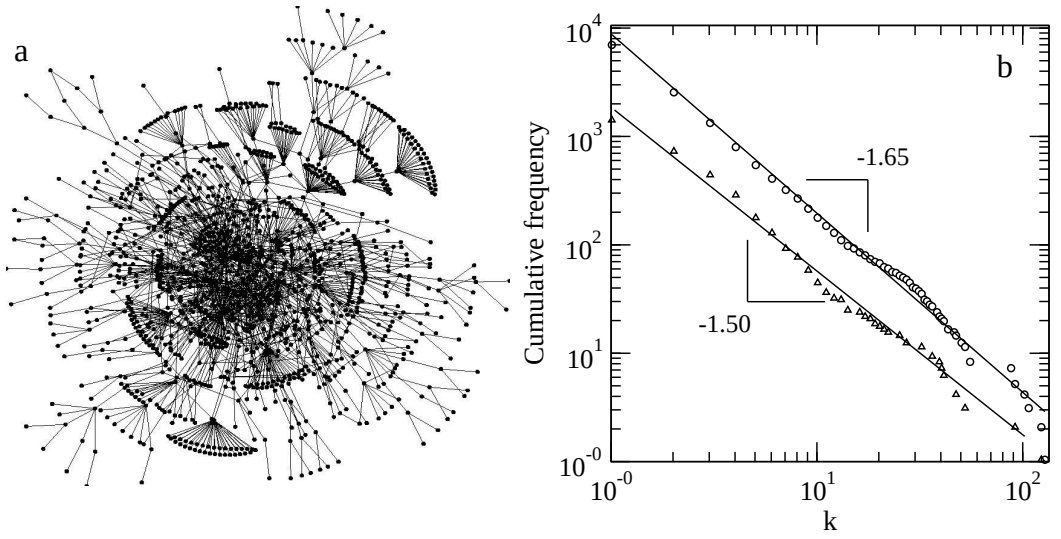

Fig. 1 – (a) One of the largest components of the java net (Ω_2, displays scale-free and small world behavior (see text). In (b) the cumulative frequencies $P_>(k)$ are shown for the two largest components. We have $P_>(k) \sim k^{-\gamma-1}$, with $\gamma_1 = 1.5 \pm 0.05$ and $\gamma_2 = 1.65 \pm 0.08$.

Artificial networks offer an invaluable reference when dealing with the rules that underlie their building process [19]. Here we show that a very important class of networks derived from software architecture maps, displays the previous patterns as a result of a design optimization process.

The importance of software and understanding how to build efficiently software systems is one of our major concerns. Software is present in the core of scientific research, economic markets, military equipments and health care systems, to name a few. Expensive costs (thousands of millions of dollars) are associated with the software development process. In the past 30 years we have assisted to the birth and technological evolution of software engineering, whose objective is to provide methodologies and tools to control and build software efficiently. Software engineers conceive programs with graphs as architects use plans for buildings. The software architecture is the structure of the program. The building blocks are software components and links are relationships between software components. The interactions between all the components yields the program functionality. *Class diagrams* constitute a well-known example of such graphs [20]. In this case, software components are also known by the technical term *class*. We have analysed the class diagram of the public Java Development Framework 1.2 (JDK1.2) [21], which is a large set of software components widely used by Java applications, as well as the architecture of a large computer game [22].

These are examples of highly optimized structures, where design principles call for diagram comprehensibility, grouping components into modules , flexibility and reusability (i.e. avoiding the same task to be performed by different components) [23]. Although the entire plan is controlled by engineers, no design principle explicitely introduces preferential attachment nor scaling and small-worldness. The resulting graphs, however, turn out to be SW and SF nets.

The software graph is defined by a pair $\Omega_s = (W_s, E_s)$, where $W_s = \{s_i\}, (i = 1, ..., N)$ is the set of $N = |\Omega|$ classes and $E_s = \{\{s_i, s_j\}\}$ is the set of edges/connections between classes. The *adjacency matrix* ξ_{ij} indicates that an interaction exists between classes $s_i, s_j \in \Omega_s$

($\xi_{ij} = 1$) or that the interaction is absent ($\xi_{ij} = 0$). The average path lenght l is given by the average $l = \langle l_{min}(i,j) \rangle$ over all pairs $s_i, s_j \in \Omega_s$, where $l_{min}(i,j)$ indicates the length of the shortest path between two nodes. The clustering coefficient is defined as the probability that two classes that are neighbors of a given class are neighbors of each other. Poissonian graphs with an average degree $\bar{k}$ are such that $C \approx \bar{k}/N$ and the path length follows [3]:

$$l \approx \frac{\log N}{\log(\bar{k})} \tag{1}$$

C is easily defined from the adjacency matrix, and is given by:

$$C = \left\langle \frac{2}{k_i(k_i-1)} \sum_{j=1}^{N} \xi_{ij} \left[\sum_{k \in \Gamma_i} \xi_{jk} \right] \right\rangle_{\Omega_s} \tag{2}$$

It provides a measure of the average fraction of pairs of neighbors of a node that are also neighbors of each other.

The building process of a software graph is done in parallel (different parts are build and gradually get connected) and is assumed to follow some standard rules of design [20,23]. None of these rules refer to the overall organization of the final graph. Essentially, they deal with optimal communication among modules and low cost (in terms of wiring) together with the rule of avoiding hubs (classes with large number of dependencies, that is, large degree). The set of bad design practices, such as making use of large hubs, is known as *antipatterns* in the software literature: see [24]. The development time of the application should be as short as possible because the expensive costs involved. It is argued in literature [23] that there is an optimum number of components so that cost of development is minimized, but it is not possible to make a reliable prediction about this number. Adding new software components involves more cost in terms of interconnections between them (links). Conversely, the cost per single software component decreases as the overall number of components (nodes) is increased because the functionality is spread over the entire system. Intuitively, a trade-off between the number of nodes and the number of links must be chosen.

However, we have found that this (local) optimization process results in a net that exhibits both scaling and small-world structure. First, we analyzed JDK1.2 network has $N = 9257$ nodes and $N_c = 3115$ connected components, so that the complete graph Ω_s is actually given by $\Omega_s = \cup_i \Omega_i$, where the set is ordered from larger to smaller components ($|\Omega_1| > |\Omega_2| > ... > |\Omega_{N_c}|$). The largest connected component, Ω_1, has $N_1 = 1376$, with $< k >= 3.16$ and $\gamma = 2.5$, with clustering coefficient [4] is $C = 0.06 \gg C^{rand} = 0.002$ and the average distance $l = 6.39 \approx l^{rand} = 6.28$, i.e. it is a small-world. The same basic results are obtained for Ω_2 (shown in fig. 1a): here we have $N_2 = 1364$, $< k >= 2.83$ and $\gamma = 2.65$, $C = 0.08 \gg C^{rand} = 0.002$ and $l = 6.91 \approx l^{rand} = 6.82$.

The degree distribution for the two largest components is shown in figure 1b, where we have represented the cumulative distribution

$$P_>(k;\Omega_i) = \sum_{k' \geq k}^{N(\Omega_i)} p(k', \Omega_i) \tag{3}$$

for $i = 1, 2$. We can see that the largest components display scaling, with estimated exponents $\gamma \approx 2.5 - 2.65$.

Similar results have been obtained from the analysis of a computer game graph [22]. This is a single, complex piece of software which consists of $N = 1989$ classes involving different

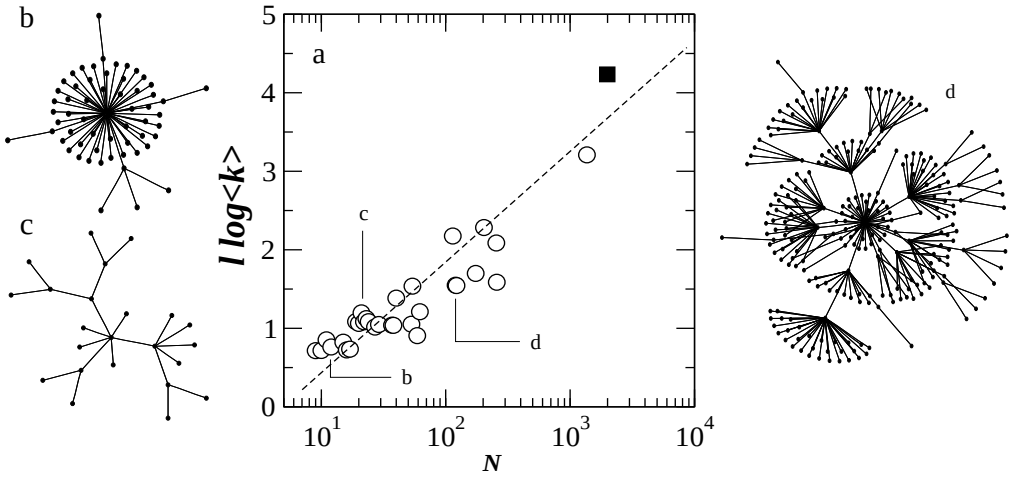

Fig. 2 – (a) Using the 32 connected components with more than 10 classes (nodes), the $l\log(\bar{k}) - N$ plots is shown. As predicted from a SW structure, the components follow a straight line in this linear-log diagram. Three subwebs are shown (c-d), displaying hubs but no clustering (their location is indicated in (a)). The black square corresponds to the computer game graph.

aspects like: real-time computer graphics, rigid body simulation, sound and music playing, graphical user interface and memory management. The software architecture graph for the game has a large connected component that relates all subsystems. The cumulative degree frequency for the entire system is scale-free, with $\gamma = 2.85 \pm 0.11$. The network also displays SW behaviour: the clustering coefficient is $C = 0.08 \gg C^{rand} = 0.002$ and the average distance is $l = 6.2$, close to $l^{rand} = 4.84$.

These results reveal a previously unreported global feature of software architecture which can have important consequences in both technology and biology. This is, as far as we know, the first example of a scale-free graph resulting from a local optimization process instead of preferential attachment [4] or duplication-rewiring [25, 26] rules. Since the failure of a single module leads to system's breakdown, no global homeostasis has been at work as an evolutionary principle, as it might have occured in cellular nets. In spite of this, the final structure is very similar to those reported from the analysis of cellular networks. Second, our results suggest that optimization processes might be also at work in the latest, as it has been shown to occur in transport nets [11].

Complex biosystems are often assumed to result from selection processes together with a large amount of tinkering [27]. By contrast, it is often assumed that engineered, artificial systems are highly optimized entities, although selection would be also at work [28]. Such differences should be observable when comparing both types, but the analysis of both natural and artificial nets indicates that they are often remarkably similar, perhaps suggesting general organization principles. Our results support an alternative scenario to preferential attachment based on cost minimization together with optimal communication among units [14] process. The fact that small-sized software graphs are trees (as one would expect from optimization leading to hierarchical structures, leading to stochastic Cayley trees [6]) but that clustering emerges at larger sizes might be the outcome of a combinatorial optimization process: As the number of modules increases, the conflicting constraints that arise among different parts of

the system would prevent reaching an optimal structure [29]. Concerning cellular networks, although preferential linking might have been at work [30], optimization has probably played a key role in shaping metabolic pathways [31–33]. We conjecture that the common origin of SF nets in both cellular and artificial systems such as software might stem from a process of optimization involving low cost (sparse graph) and short paths. For cellular nets (but not in their artificial counterparts) the resulting graph includes, for free, an enormous homeostasis against random failure.

* * *

The authors thanks Javier Gamarra, Jose Montoya, William Parcher, Charles Herman and Marcee Herman for useful comments. This work was supported by the Santa Fe Institute (RFC and RVS) and by grants of the Generalitat de Catalunya (FI/2000-00393, RFC) and the CICYT (PB97-0693, RVS).

1. Albert, R. and Barabási, A.-L., cond-mat/0106096.
2. Watts, D. J. & Strogatz, S. H. *Nature* **393** (1998) 440.
3. Newman, M. E. J. *J. Stat. Phys.* **101** (2000) 819.
4. Barabási, A.-L. & Albert, R. *Science* **286** (1999) 509.
5. Amaral, L. A. N., Scala, A., Barthélemy, M. & Stanley, H. E. *Proc. Natl. Acad. Sci. USA* **97** (2000) 11149.
6. Caldarelli, G., Marchetti, R. Pietronero, L. (2000) *Europhys. Lett.* **52**, 304.
7. Jeong, H., Mason, S., Barabási, A. L. and Oltvai, Z. N. *Nature* **411** (2001) 41.
8. Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N. and Barabasi, A.-L. *Nature* **407** (2000) 651.
9. Newman, M. E. J. *Proc. Natl. Acad. Sci. USA* **84** (2001) 404.
10. Ferrer i Cancho, R. and Solé, R. V. *Procs. Roy. Soc. London B*, **268** (2001) 2261.
11. West, B. and Brown, J. *Scaling in Biology*, Oxford, New York (2000).
12. Rodriguez-Iturbe, I. and Rinaldo, A. *Fractal River Basins*, Cambridge U. Press, Cambridge (1997).
13. Mathias, N. and Gopal, V. *Phys. Rev. E* **63** (2001) 1.
14. Ferrer Cancho, R. and Solé, R. V., *SFI Working paper* 01-11-068.
15. Cherniak, C. *Trends Neurosci.* **18**, 522-527 (1995).
16. Stephan, K. A., Hilgetag, C.-C., Burns, G. A. P. C, O'Neill, M. A., Young, M. P. and Kötter, R. *Phil. Trans. Roy. Soc. B* **355** (2000) 111.
17. Albert, R. A., Jeong, H. and Barabási A.-L. *Nature*, **406** (2000) 378.
18. Solé, R. V. and Montoya, J. M. (2001) Procs. Royal Soc. London B **268**, 2039.

19. Ferrer Cancho, R. Janssen, C. and Solé, R. V. (2001) *Phys. Rev. E*, **63** 32767.

20. Gamma, E., Helm R., Johnson R., Vlissides J. (1994) *Design Patterns Elements of Reusable Object-Oriented Software* (Addison-Wesley, New York)

21. Sun, Java Development Kit 1.2. Web site: http://java.sun.com/products/java/1.2/

22. UbiSoft ProRally 2002: http://ubisoft.infiniteplayers.com/especiales/prorally/

23. Pressman, R. S. (1992) *Software Engineering: A Practitioner's Approach*, (McGraw-Hill)

24. Brown, W. H., Malveau, R., McCormick, H., Mowbray, T., and Thomas, S. W. (1998) *Antipatterns: Refactoring Software, Architectures, and Projects in Crisis*, (John Wiley & Sons, New York)

25. Solé, R. V., Pastor-Satorras, R., Smith, E. D. and Kepler, T. (2002). Adv. Complex Syst. (in press)

26. Vazquez, A., Flammini, A., Maritan, A. and Vespignani, A. (2001) cond-mat/0108043.

27. Jacob, F. (1976) *Science* **196**, 1161-1166.

28. Monod, J. (1970) *Le hasard et la nécessité*, Editions du Seuil, Paris.

29. Kauffman, S. A. (1993) *Origins of Order*, Oxford, New York.

30. Wagner. A. and Fell, D. A. *Proc. Roy. Soc. London B* 268 (2001) 1803.

31. Mittenthal, J.E., A. Yuan, B. Clarke, and A. Scheeline (1998) Bull. Math. Biol. **60**, 815-856.

32. Melendez-Hevia, E. Waddell, T. G. and Shelton, E. D. Biochem. J. **295**, 477.

33. Melendez-Hevia, E. Waddell, T. G. and Montero, F. *J. Theor. Biol.* **166** (1994) 201.