Scale-free networks from optimal design

To cite this article: S. Valverde *et al* 2002 *EPL* **60** 512

EUROPHYSICS LETTERS 15 November 2002

Europhys. Lett., **60** (4), pp. 512–517 (2002)

Scale-free networks from optimal design

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

[1] *ICREA-Complex Systems Lab, Universitat Pompeu Fabra (GRIB)*
Dr. Aiguader 80, Barcelona 08003, Spain
[2] *Santa Fe Institute - 1399 Hyde Park Road, NM 87501, USA*

(received 19 April 2002; accepted in final form 10 September 2002)

PACS. 05.10.-a – Computational methods in statistical physics and nonlinear dynamics.
PACS. 05.65.+b – Self-organized 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 the presence of small-world behavior) 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 (typically 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].

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 millions dollars) are associated with the software development process. In the past thirty years we have assisted to the birth and technological evolution of software engineering, whose objective is to provide methodologies and tools to design and build software efficiently.

Designed software represents a human vision or conception of a problem. Briefly, to design a software application is to decompose it into a set of disjoint, smaller, functional units also named hereafter software components([1]). There are well-defined principles that enable the engineer to find the optimal partition that solves the problem within the external constraints of available time and economic resources [20].

A software component is an abstract entity which represents everything from hardware components to entire software applications [21]. The software functionality is distributed among these components by the engineer who has to decide if a given function is implemented by a single component, or as the result of several interacting components. In the latter case, the engineer defines explicitly the communication between the components by means of relationships. There are also other types of relationships but in our study we will not make any difference between them. The number of software components and their sizes (usually measured as number of lines of source code (NLOC)) can vary greatly from one application to another.

The above design process is documented by means of the software architecture where the different software components and their relationships are explicitly depicted (this is a diagram expressed in standard graphical notation like UML [22] or OMT [23]). We define the software graph as a network obtained from an existing software architecture where every node corresponds to exactly one software component and two nodes are linked if the corresponding software components are related to each other in the software architecture. This kind of graph constitutes a very rough approximation to the fully detailed software architecture but it is of interest because the topological information is preserved([2]).

We have analysed the class diagram of the public Java Development Framework 1.2 (JDK 1.2) [24], which is a large set of software components widely used by Java applications, as well as the architecture of a large computer game [25]. 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) [20]. Although the entire plan is controlled by engineers, no design principle explicitly 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 $W_s = \{s_i\}$, $(i = 1, \ldots, N)$ 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 $s_i, s_j \in \Omega_s$

([1])In object-oriented languages (*i.e.*: C++, Java) a component is also known by the term “class”.

([2])The inner details of every class were discarded and only the class names and the names of the relationships were kept. The type and semantics of every relationship is not considered here. All the software architectures analysed here were stored in an electronic format compatible with the Rational Rose 98 specification (see `http://www.rational.com`) and converted automatically by custom software to our graph representation.

($\xi_{ij} = 1$) or that the interaction is absent ($\xi_{ij} = 0$). The average path length 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 built and gradually get connected) and is assumed to follow some standard rules of design [20, 21, 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 [26]. The development time of the application should be as short as possible because of the expensive costs involved. It is argued in the literature [20] that there is an optimum number of components so that the 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 JDK 1.2 network which has $N = 9257$ nodes and $N_{\rm c} = 3115$ 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 ($|\Omega_1| > |\Omega_2| > \ldots > |\Omega_{N_{\rm c}}|$). The largest connected component, Ω_1, has $N_1 = 1376$, with $\langle k \rangle = 3.16$ and $\gamma = 2.5$, with clustering coefficient [4] $C = 0.06 \gg C^{\rm rand} = 0.002$ and the average distance $l = 6.39 \approx l^{\rm rand} = 6.28$, *i.e.* it is a small world. The same basic results are obtained for Ω_2 (shown in fig. 1a): here we have $N_2 = 1364$, $\langle k \rangle = 2.83$ and $\gamma = 2.65$, $C = 0.08 \gg C^{\rm rand} = 0.002$ and $l = 6.91 \approx l^{\rm rand} = 6.82$.

The degree distribution for the two largest components is shown in fig. 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 [25]. This is a single, complex piece of software which consists of $N = 1989$ classes involving different

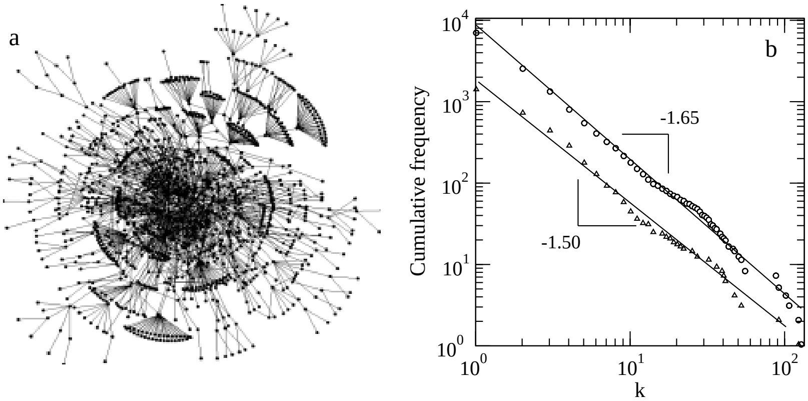

Fig. 1 – (a) One of the largest components of the java net (Ω_2, displays scale-free and small-world behavior (see text)). In (b) the cumulative frequencies $P_>(k)$ are shown for the two largest components. Here the power-law fit gives $\gamma_1 = 2.5 \pm 0.05$ and $\gamma_2 = 2.65 \pm 0.08$.

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 $C = 0.08 \gg C^{\mathrm{rand}} = 0.002$ and the average distance is $l = 6.2$, close to $l^{\mathrm{rand}} = 4.84$.

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 [27, 28] 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 occurred in cellular nets. In spite of this, the final

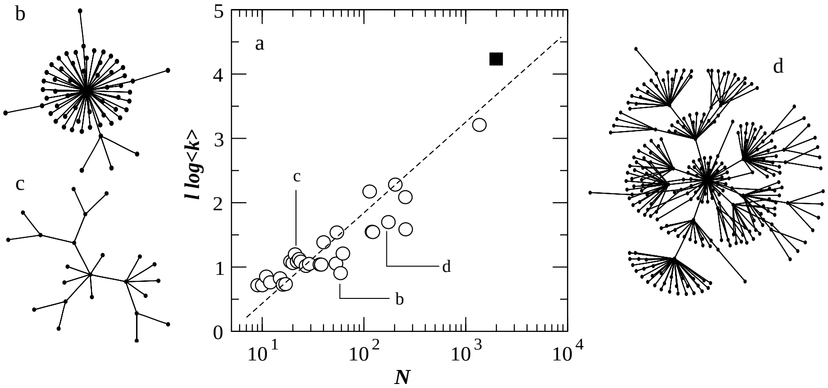

Fig. 2 – (a) Using the 32 connected components with more than 10 classes (nodes), the $l\log(\bar{k})$-N plots is shown. As predicted from a SW structure, the components follow a straight line in this linear-log diagram. Three subwebs are shown (c-d), displaying hubs but no clustering (their location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

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 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 [29]. By contrast, it is often assumed that engineered, artificial systems are highly optimized entities, although selection would be also at work [30]. 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 (see fig. 2) 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 [31]. Concerning cellular networks, although preferential linking might have been at work [32], optimization has probably played a key role in shaping metabolic pathways [33–35]. 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 thank J. Gamarra, J. Montoya, W. Parcher, C. Herman and M. 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).

REFERENCES

[1] Albert R. and Barabási A.-L., cond-mat/0106096.

[2] Watts D. J. and Strogatz S. H., *Nature*, **393** (1998) 440.

[3] Newman M. E. J., *J. Stat. Phys.*, **101** (2000) 819.

[4] Barabási A.-L. and Albert R., *Science*, **286** (1999) 509.

[5] Amaral L. A. N., Scala A., Barthélemy M. and Stanley H. E., *Proc. Natl. Acad. Sci. USA*, **97** (2000) 11149.

[6] Caldarelli G., Marchetti R. and Pietronero L., *Europhys. Lett.*, **52** (2000) 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 Barabási A.-L., *Nature*, **407** (2000) 651.

[9] Newman M. E. J., *Proc. Natl. Acad. Sci. USA*, **84** (2001) 404.

[10] Ferrer Cancho R. and Solé R. V., *Proc. R. Soc. London, Ser. 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 University 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** (1995) 522.

[16] Stephan K. A., Hilgetag C. C., Burns G. A. P. C., O'Neill M. A., Young M. P. and Kötter R., *Philos. Trans. R. 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., *Proc. R. Soc. London, Ser. B*, **268** (2001) 2039.

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

[20] Pressman R. S., *Software Engineering: A Practitioner's Approach* (McGraw-Hill, New York) 1992.

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

[22] Booch G., Rumbaugh J. and Jacobson I., *Unified Modeling Language User Guide* (Addison-Wesley, New York) 1999.

[23] Rumbaugh J., Blaha M., Premerlani W., Eddy F. and Lorensen W., *Object-Oriented Modeling and Design* (Prentice-Hall, Englewood Cliffs) 1991.

[24] Sun, Java Development Kit 1.2. Web site: `http://java.sun.com/products/java/1.2/`.

[25] UbiSoft ProRally 2002: `http://ubisoft.infiniteplayers.com/especiales/prorally/`.

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

[27] Solé R. V., Pastor-Satorras R., Smith E. D. and Kepler T., *Adv. Complex Syst.*, **5** (2002) 43.

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

[29] Jacob F., *Science*, **196** (1976) 1161.

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

[31] Kauffman S. A., *Origins of Order* (Oxford, New York) 1993.

[32] Wagner A. and Fell D. A., *Proc. R. Soc. London, Ser. B*, **268** (2001) 1803.

[33] Mittenthal J. E., Yuan A., Clarke B. and Scheeline A., *Bull. Math. Biol.*, **60** (1998) 815.

[34] Melendez-Hevia E., Waddell T. G. and Shelton E. D., *Biochem. J.*, **295** (1993) 477.

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