Punctuated Equilibrium in the Large Scale Evolution of Programming Languages

Sergi Valverde
Ricard Solé

SFI WORKING PAPER: 2014-09-030

SANTA FE INSTITUTE

Punctuated equilibrium in the large scale evolution of programming languages

Sergi Valverde[*1,2] and Ricard Solé[†1,3,2]
[1]ICREA-Complex Systems Lab, Universitat Pompeu Fabra, Dr Aiguader 88, 08003 Barcelona, Spain
[2]Institut de Biologia Evolutiva, UPF-CSIC, Psg Barceloneta 37, 08003 Barcelona, Spain
[3]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM 87501, USA

The analogies and differences between biological and cultural evolution have been explored by evolutionary biologists, historians, engineers and linguists alike. Two well known domains of cultural change are language and technology. Both share some traits relating the evolution of species, but technological change is very difficult to study. A major challenge in our way towards a scientific theory of technological evolution is how to properly define evolutionary trees or clades and how to weight the role played by horizontal transfer of information. Here we study the large scale historical development of programming languages, which have deeply marked social and technological advances in the last half century. We analyse their historical connections using network theory and reconstructed phylogenetic networks. Using both data analysis and network modelling, it is shown that their evolution is highly uneven, marked by innovation events where new languages are created out of improved combinations of different structural components belonging to previous languages. These radiation events occur in a bursty pattern and are tied to novel technological and social niches. The method can be extrapolated to other systems and consistently captures the major classes of languages and the widespread horizontal design exchanges, revealing a punctuated evolutionary path.

Keywords: Cultural evolution, punctuated equilibrium, networks, technology, programming languages, software

I. INTRODUCTION

Is cultural evolution similar to biological evolution? Darwin's theory of natural selection has been often used as a basic blueprint for understanding the tempo and mode of cultural change, particularly in relation to human language (1-3) and technological designs (4,5). Darwin himself became interested in the similarities between natural and human-driven evolutionary change and shortly after the publication of *The Origin of Species*, scholars started to speculate about the similarities between organic and man-made evolution (4). A crucial component of cultural evolution, technology has received great attention as a parallel experiment of selection, diversification and extinction (6). Tentative steps towards a theory of technological innovation have been made but the debate on the similarities versus differences between cultural and biological change remains unabated (7). In this context, it has been suggested (5) that innovations occur mainly through combination of previous technologies. But several questions remain open: Can we test such idea in a systematic way? What type of large-scale evolutionary trends are associated to technological evolution based on combination?

Most textbook examples describe historical inventions (8) as human-driven events, where a success story marks the creation of a new invention. Unfortunately, no systematic approach to extract phylogenetic relationships exist (9) and only in some cases a hand-curated tree-like structure can be inferred using human expertise and available historical records (9,10). Often, no simple trees are obtained but instead networks with merging of branches are found. Human languages are a clear exception to the rule, since it is possible to properly define distances among words or other components and reconstruct their evolutionary record (11). As it occurs with microbial species (12) languages also display high levels of horizontal transfer, but this can also be treated with the appropriate tools (13). Surprisingly, almost no attention has been dedicated to the evolution of information technology, despite its well preserved fossil record (6,14-15). Programming languages are (along with large scale hardware design and the Internet) the major players of the software revolution (16). They play, within the context of technological evolution, the same role of tongues in human language. However, instead of making communication possible among two individuals, they provide the medium to single-directed communication between humans and machines.

In this paper we use programming languages as our case study to analyse the evolutionary patterns of technological evolution. Since the appearance in 1952 of the first short written code (17) programming languages rapidly emerged, diversified and propagated through different niches (communities of users). They also coevolved with hardware (18) and can be compared to human languages, which also co-evolved with brains (19) although with a significantly different status: They are both the means of telling computers how to perform useful work (functional perspective) and a tool to share algorithmic schemes between the programmers (a communication tool). The evolutionary patterns of these systems, as shown below, is punctuated and can be reconstructed in a systematic

*corresponding author
†corresponding author

way.

Data set of programming languages In dealing with the historical record of our system, the basic elements (nodes in a network) will be programming languages (PLs) defined as artificial languages designed to communicate instructions to a computer (16,17). We have reconstructed the history of the most influential PLs from the information publicly available in Wikipedia (see SM2) which provides contents on language syntax, history, applications and, more importantly, a list of PLs that have influenced their design (see next section). Whenever the historical records were found to be inconsistent, they have been double-checked with alternative sources of information.

The data set studied here includes N=347 different PLs spanning a period of T=59 years, from 1952 to 2010. They belong to fairly different groups of architectures (see below). Some of them exist today while others have disappeared. Although the total number of different languages that have been created is much more larger than the above number, the chosen data set captures a significant fraction of the most relevant PLs. Previous studies of the history of PLs have considered separate version numbers for popular languages. For example, Fortran is an old PL that has remained in place despite experiencing changes through its technological history with slightly different versions that are easily identified as "dialects" of the original.

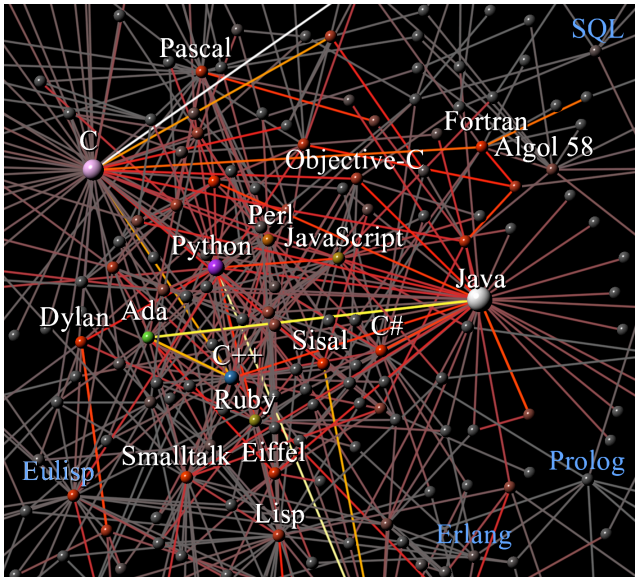

FIG. 1 The network of programming languages. We display the central core of dependencies between different PLs as gathered from our dataset (SM2). Here a directed link exist between two programming languages if the design of the later has been based in the structure of the previous one. Despite of their time-dependent nature, this is far from a simple tree. Instead, it defines a tangled, complex network. A subset of languages (including C, Java, Pyhton and Lisp) define a special group here (lighter balls). They are the core innovations within the universe of PLs (see SM1).

II. TECHNOLOGICAL EVOLUTIONARY TREES

The first goal of our study is to provide a systematic way of extracting a phylogenetic tree and identify major clades. Using our data set, connections among languages are defined in terms of their list of influences as described in Wikipedia. Here, we make a distinction between influence and usage. Our aim is to capture the relative importance of a given innovation on the evolution of future languages rather than its success. For example, Algol-68 was not widely used but it had a profound impact in subsequent developments. Similarly, ISWIM was an abstract (never implemented) language, which nonetheless had a great impact in the development of functional languages (20). Influence is a directed relation, i.e., new languages are always influenced by older ones but not the other way around. This allows defining a (directed, time-dependent) graph $G = (\Pi, E)$ formed by the set of programming languages Π and a set E of links among them (figure 1). This graph can be analysed using standard techniques (see section 1 in SM1) but it cannot used as a proper phylogenetic map. In order to obtain an evolutionary tree along with a map of horizontal exchanges, a systematic method is required.

The previous graph allows defining the so called adjacency matrix A, whose elements $a_{ij} = 1$ if a directed link $i \to j$ exists, meaning that the π_j language is based (at least in part) in the $i-$th one. Given a $\pi_i \in \Pi$, it will be one one hand based on other languages and can on the other hand influence others. The out-degree k_i^{out} of π_i is the number of edges leaving it, i. e.: $k_i^{out} = \sum_j a_{ij}$. It weights the number of times that a parent π_i has been used to build new, offspring languages. Similarly the in-degree of π_i is the number of edges entering it, $k_i^{in} = \sum_j a_{ji}$ and gives the number of previous languages that influenced the invention of π_i.

Now we need a systematic way of building a phylogeny, using for that purpose only the topological information associated to our graph along with the time coordinate. This can be done by following the approach of Gualdi, Yeung and Zhang (21). The method naturally incorporates the fact that languages sharing similar parents are likely to be technologically related. We define the the *impact* of $\alpha \in \Pi$ impact on its offspring π_i by means of:

$$I_{i\to\alpha} = \sum_{j\in\omega_\alpha - \{i\}} \lambda s_{ij}^{fol} + (1-\lambda) s_{ij}^{aut} \qquad (1)$$

where ω_α is the set of all nodes derived from α. This impact measure calculates the structural similarity between pairs of nodes in the directed network. A node α strongly influences node π_i if α has (different) offspring π_j that is itself similar to π_i. Let $\Gamma(i)$ be the set of neighbours of node i. Hereafter we use (21) $\lambda = 1/2$. The simplest measure of neighbourhood overlap among two languages

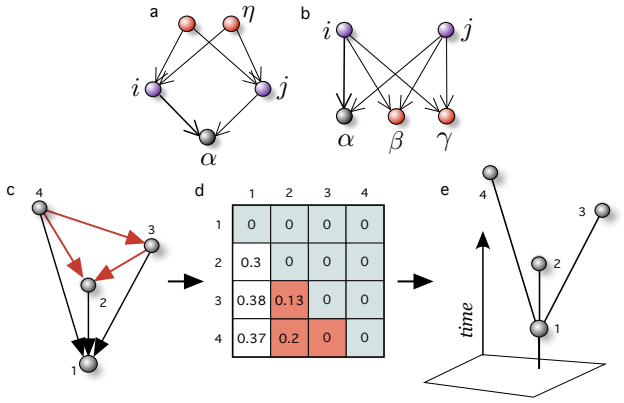

FIG. 2 Vertical and horizontal transfer of information among programming languages. In (a-b) we summarise the method followed here to compute the influence backbone. Here the balls indicate different PLs and arrows indicate influence (which languages where used to build which). Two ingredients are used when measuring the impact $I_{i\to\alpha}$ of node α on its offspring i: (a) the in-similarity or how many follower nodes (such as η) are influenced by both i and j and (b) the out-similarity between i and j, which weights how many author nodes (such as β and γ) influence both i and j (see text). (c) The influence backbone Ω keeps the links with highest impact (black links) and discards the others (red links).

π_i and π_j is:

$$s_{ij} = |\Gamma(i) \cap \Gamma(j)| = \sum_l a_{il} a_{jl} \tag{2}$$

The above measure is biased towards nodes with a large offspring. In order to correct this effect, a more balanced definition sets the degree of similarity between π_i and π_j proportional to the probability that a two-step random walk from π_i to π_j traverses any neighbour π_k (see fig 3). For directed networks, there are two naturally defined similarity definitions for every pair of nodes, namely, in-similarity and out-similarity (see below). In-similarity is defined as:

$$s_{ij}^{fol} = \frac{1}{k_j} \sum_l \frac{a_{li} a_{lj}}{p_l} \tag{3}$$

where *fol* stands for "follower". On the other hand, out-similarity is defined as follows:

$$s_{ij}^{aut} = \frac{1}{p_j} \sum_l \frac{a_{il} a_{jl}}{k_l} \tag{4}$$

where *aut* stands for "authority". Finally, these two measures are combined in (1) by taking a weighted sum where $\lambda \in \{0, 1\}$. The above measure is a good tradeoff between accuracy and complexity, see (22). The method generates a backbone based in identifying the most influential parent for each language, while we also have an additional graph that keeps all the "horizontal" exchanges among languages. As shown below, the resulting tree and its major branches capture the major groups (clades) of programming languages.

III. PROGRAMMING LANGUAGE CLADES

The result of using our algorithm is shown in figure 3, where the most relevant large-scale patterns of PL interactions are displayed, actually defining the clades of our system. In (a-b) the total number of interactions shows a two-regime behaviour (fig 3a) with an accelerated growth in the second phase, starting in the 1980s, matching the emergence of personal computers (23,24). A bursty signal (fig 3b) is observed when we plot the number of incoming links k_i for each new language. The clustered structure involves high peaks indicating that new complex languages where created out from previously existing ones. These bursts are reveal a rapid "speciation" among closely related languages. The in-degree distribution is broad, following a power law $P(k_{in}) \approx k_{in}^{\gamma_i}$ with $\gamma_i \approx 2$ whereas the out degree distribution is exponential. These features will be relevant for our interpretation of the evolutionary trees.

Our method finds two large, separated subsets of programming languages defining the major clades: imperative (or procedural) and declarative families, along with several smaller classes (see figure 2 in SM1). These trees exhibit a noticeable asymmetry and, despite our method does not include additional information beyond influence relationships, the clades accurately maps the known historic development of programming languages. The largest subtree defines 197 procedural-related languages rooted in Speedcoding, the oldest language in our database and a direct ancestor of Fortran (first widely used programming language). The design of early procedural languages was constrained by hardware costs and application requirements. The Turing machine is the theoretical model underlying this class of languages, which defines computational tasks as sequences of computer instructions that alter the machine state. Early computers were memory-limited, slow and very expensive pieces of equipment. Because of efficiency concerns, the design of procedural languages in this period mirrored the underlying Turing machine. The main purpose of early computers was scientific and data processing applications, and this required a language capable of describing sequences of complex calculations.

Some of the early procedural languages (like Fortran) have persisted to this day but later on the evolution of programming languages adopted ideas from other paradigms. This is the case of declarative languages whose origins are associated to the launch of artificial intelligence (AI) in 1956. Researchers recognised the limitations of procedural languages like Fortran, which are more suited for engineering and mathematical calculations than to the sort of symbol manipulations associated to an intelligent computer. Lisp was introduced

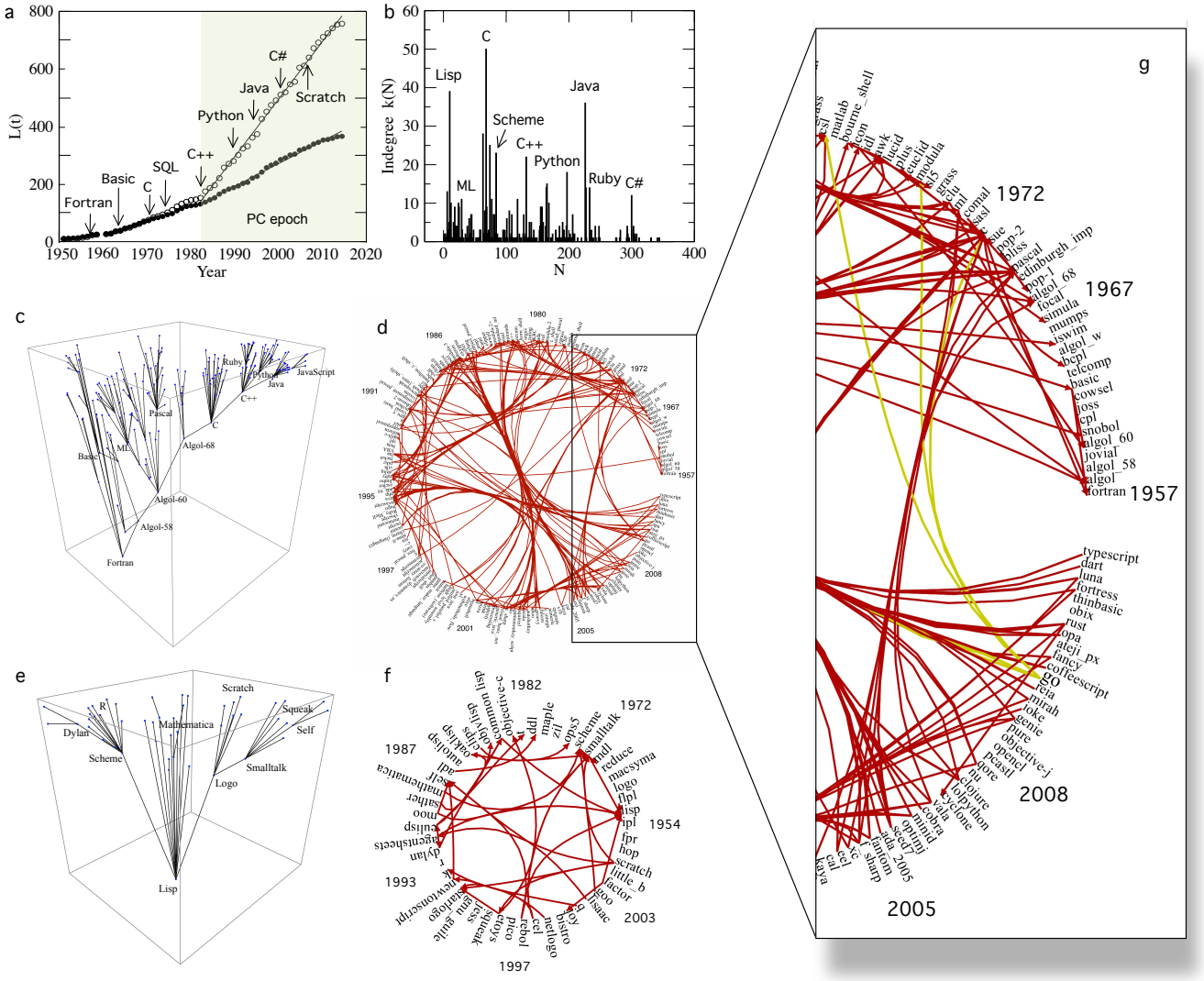

FIG. 3 Large-scale evolution of Programming Languages. In (a) we display the time series of (a) the total number of languages (N, filled circles) and of interactions (L, open circles) respectively. Notice the abrupt increase in L that takes place around 1980. In (b) number of incoming interactions against N is displayed, where PLs have been sorted chronologically. Phylogenetic and influence maps in PLs are shown in (c-f) for the two largest groups defining lineages. Within each of these lineages, we can reconstruct a phylogenetic subtree (d) with the vertical axis indicating release time. For example, the influence backbone among the family of imperative PLs (1953 - 2012) is shown in (e) which gives all the horizontal transfer among them (f). The same diagrams are shown in (g-h) for the family of functional languages (1954- 2011).

as the standard AI language, and their descendants enabled programmers to write code in terms of the problem domain instead of specific hardware details. These so-called functional languages treat computation as the evaluation of mathematical functions and avoid changes ("side-effects") in the machine state, unlike procedural languages.

The functional subtree is rooted in IPL (parent of Lisp) and consists of 47 languages (see figure 3e). Procedural languages are more popular than declarative ones presumably because they are associated to a more natural style of programming (see below). Declarative languages like Lisp are related to mathematical abstractions which, in general, are harder to grasp by beginners. Our analysis suggests that AI has been a strong influence: the sum of horizontal influences from functional to imperative languages is much greater than the other way around. Other members in the declarative family, such as logic and database paradigms define segregated small subtrees correctly mapping specific application domains (see section 2 in SM1).

The influence graphs describe a very interesting situation: far from observing links relating languages close in time, bundles of links reveal very large time windows

connecting modern and old languages (figs 3d, f). The zoomed diagram shown in figure 3g provides some illustration of this: links exist between languages created within the first and last decades of programming history. These graphs support the combinatorial rule of technological evolution (5) but point to a richer picture: there are groups of time-close languages whose properties are recruited to build new clusters of languages far in the future.

Modern programming languages can been understood as combinations ("hybrids") of different paradigms. In the 1960's, an international consortium designed Algol, a successor to Fortran that incorporated functional elements taken from Lisp. By the 1970's, there was the perception that the hundreds of languages derived from Fortran and Lisp were flawed and provided limited support when developing complex software. A partial response to this "software crisis" was the convergence between the procedural an declarative approaches in the (so-called) object-oriented (OO) paradigm. This approach recognises that programming is neither the breaking down of calculations (imperative approach) nor the definition of functions (functional approach) but the definition and manipulation of domain objects (like numbers, strings, arrays, files, graphs).

At the clade level, the organisation of the influence backbone transitions from the historical separation between the procedural and declarative paradigms to this pattern of convergent evolution. In many of the linages, we have examples of languages showing clear OO traits. For example, the best know OO language is Smalltalk (1976), which was strongly influenced by Lisp (see fig. 3e). On the other hand, the simulation language Simula (1960) extended Algol with concepts borrowed from the OO paradigm.

IV. MODELING PROGRAMMING LANGUAGE EVOLUTION

How are these patterns generated dynamically? Does a purely combinatorial model explain the previous observations? In order to define a framework accounting for the large-scale features of our system, we need a null model that generates nodes (inventions) and makes new inventions to connect with previous ones leading to a network from which the phylogenetic tree can be extracted. Moreover, if a previous invention is chosen, which is related with others, it is likely that the new node gets also linked with those related inventions with some probability. In other words, the model must include a process of growth by combination incorporating the "inheritance" of existing correlations among technologies and must provide both vertical and horizontal relations. Such requirements discard standard models of cladogenesis based on simple branching rules (25,26).

Here, we use a growing network model that fulfils the previous requirements and leads to the observed in- and

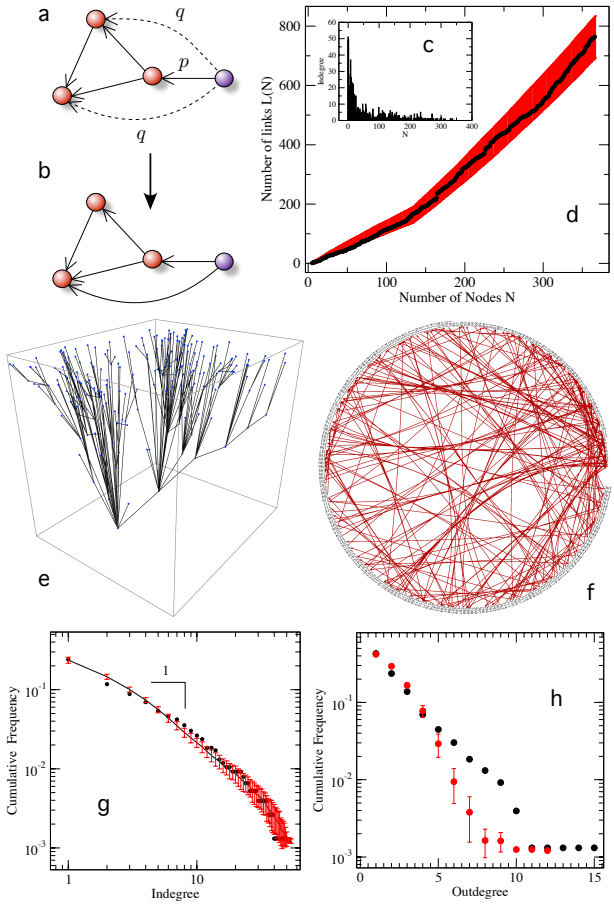

FIG. 4 Modeling the growth of the programming language network. Each time step, a new node (a) is introduced. The new node (blue) attaches to a target node (red) with probability p. This new node also inherits every link (b) from the target node (dashed links), with probability q. Both parameters can be estimated. The model correctly predicts the two-stage time evolution of $L(N)$, as shown in (c) where the real data (filled circles) is compared with the predicted one (red) using 10^2 different replicas starting from the same initial condition. A phylogenetic tree (e) and the horizontal influence map (f) can also be constructed. The tree is somewhat similar, but much less asymmetric than the ones shown in figure 3. The influence graph (f) displays heterogeneity, but far from the one exhibited by real data. The in- and out-degree distributions for model (red) and data (black) are shown in figures g and h, respectively. Our model predicts different saturation constants for each stage, i.e., $k_M^1 = 20$ and $k_M^2 = 40$.

out-degree distributions. It is based on a previous model of tinkered graph evolution (27-29). At every step, the network grows by introducing a new node (fig 4a-b) which links to m randomly chosen target node π_j with probability p as well as to all ancestor nodes of each target, with probability q (see fig. 4). As defined, mp is the mean number of ancestor languages that influence new languages and mq is the mean number of influences also inherited. By estimating these parameters out from our

empirical data set we obtain networks that are statistically close to our original graphs. In order to take into account the fact that the number of links reaches a saturation (probably because there is a limited number of features that can be reused in further innovations) we have modified the original model by including a Boltzmann saturation term to the probabilities of attachment, namely $\mathcal{P}$ is replaced by

$$\mathcal{P}(k_j) = \frac{\mathcal{P}}{1 + \exp\left(-\beta(k_j - k_M)\right)} \tag{5}$$

with $\mathcal{P} = p, q$ and $\beta = 0.1$. Since we have a two-regime scenario (fig 3a) we estimated two pairs of values, namely: $mp_1 = 0.92$ and $mp_2 = 2.2$ where $m = 2$ is the average number of randomly selected targets and p_1 and p_2 are the probability to attach to any target in stage 1 and stage 2, respectively. We can in particular match the evolution of L of links against N (fig 4c-d). From the final network, both the phylogenetic tree (fig 4e) and the horizontal transfer graph (fig 4f) can be obtained.

Another structural component is well fitted by the model. The final degree distributions (figures 4g-h) fit very well the observed asymmetry between in- and out degree. Two main observations can be made by looking at the reconstructed tree and horizontal graphs: the first is much more symmetric than the original one, while in the second we can see widespread recombination but less local and long-distance clusters of correlations.

V. ADAPTIVE RADIATIONS AND TREE IMBALANCE

The trees extracted from the model are much less asymmetric than their observed counterparts. This is deeply tied to the burstiness displayed by our system (as can be appreciated in figure 3c). Such asymmetric growth seems to be characteristic of evolution in living systems, where adaptive radiations and strong differences between clades are known to exist (30). In order to measure this asymmetry, we use standard measures of tree imbalance (31-32). Tree imbalance measures allow to study how species diversity is arranged through different branches. This can be addressed using structural measurements of tree shape (34). Here, we will focus in the average depth $\langle d \rangle$ of a tree with N nodes, defined as follows:

$$\langle d \rangle = \frac{1}{N} \sum_i d(r, i) \tag{6}$$

where $d(r, i)$ is the path length or number of intermediate nodes relating the root r with any other node i. Notice that here we compute the path length for every node, not only tree leaves (which yields a different, but related measure, e.g., see (32).

Equation [5] is a measure of tree imbalance, i.e., the degree to which subtrees are divided in groups of unequal size (31). Here, the average path length is lower-bounded by $d_{min} = 1 - 1/N$. On the other hand, the maximum average distance $d_{max} = (N^2 - 1)/4N$ corresponds to a fully imbalanced binary tree. Notice that, for large N, we have $d_{min} \approx 1$ and $d_{max} \approx \log N$. We can compare our data with the simplest null model of stochastic tree growth. At every time step, the Equal-Rates Markov or Yule's model attaches two new descendant nodes to a randomly chosen leaf node (33,34). This previous rule is performed until a tree with N nodes is obtained.

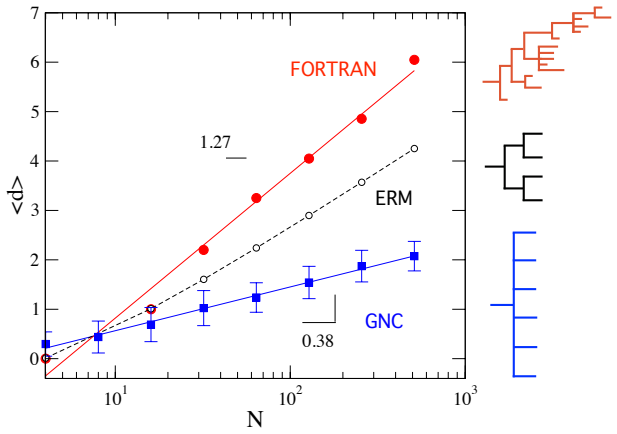

FIG. 5 Logarithmic scaling of average depth with subtree size. We compare the different scaling behaviour exhibited by the evolutionary tree of Fortran-related languages (open circles) with two models. One is the ERM model (solid line) and the other is the model of network growth by copying (GNC, red squares) (see text).

It can be shown that the average depth for trees obtained with this model is $d_{Yule} \sim log_2(N)$ for large N. Figure 5 compares the scaling of $\langle d \rangle$ for the Fortran subtree with the predictions obtained with the ERM and the GNC models. In all systems, $\langle d \rangle$ scales linearly with $\log N$. The slope of the scaling law in the Fortran subtree is larger than model predictions indicating a great tree imbalance in our phylogenies. One of the Fortran subfamilies (including descendants of Algol languages like C, Java, and C++, see below) is much more developed than lateral branches. On the other hand, the GNC prediction is much less steep. The scaling for the ERM model is in-between the Fortran lineage and the GNC model.

As it occurs with the tree of life (30-33) technological trees are highly imbalanced, largely a consequence of accelerated diversification events tied to innovations. This pattern has also been found in the diversification pattern of human languages (35,36) which exhibited strong imbalances too. The asymmetries have been proposed to be evidence of punctuated equilibrium (37,38). In our system, we do identify these shifts as major innovations associated to novel forms of engineering programming languages. The tree imbalance, but also the bundles observed in the horizontal transfer interactions are consistent with such bursts of rapid modifications.

VI. DISCUSSION

The study cultural evolutionary patterns, particularly when dealing with artifacts, is usually constrained by a lack of powerful quantitative methods. The absence of a "genome" is a great challenge, since it prevents us from exploiting some type of metric defining the distance among inventions. Only human languages allowed to sistematically reconstruct phylogenies while taking into account lateral transfer (13). In this paper we have shown that a simple network approach allows to construct phylogenetic trees from existing databases that include information on who influenced whom in a given branch of technological development. We have used this method to study one important area of information technology, namely the large-scale evolution of programming languages. Given the low level of details required to extract our networks, we predict that it can be applied to other technological webs, including other software systems, hardware development, specific tech fields (such as aircraft industry) and patent citation networks.

Our study is the first full systematic characterisation of phylogenetic patterns in a cultural evolving system beyond the human language case study. It reveals that the evolutionary dynamics displayed by programming languages fits the combination metaphor while reveals the presence of a non-uniform rates of change. Such correlations cannot be accounted for by our simple model. The unbalanced phylogenetic trees and complex horizontal transfer tells us that the underlying dynamics is rich and nontrivial. It actually supports a punctuated pattern of technological evolution. This concept has been previously explored by means of theoretical models (39,40) and supported by available historical information (41) and has been validated by our systematic method from available data of technological dependencies.

It is often said that human language coevolved with brains (19) by infecting the mind of its hosts, thus acting as a sort of viral entity. Programming languages emerged as a much needed interface to communicate human brains and programmable machines (42). They are actually virtual machines that make possible to use diverse hardware systems and thus "infect" a large variety of devices. Their improvement made possible to use more powerful machines but also to design even more powerful ones. Our work provides a rationale to rigorously explore the evolution of these virtual machines and how they coevolved with both computers and human programmers. Future work should allow to test this hypothesis by considering the parallel evolutionary changes experienced by computer hardware.

Acknowledgments We thank M. Rosas-Casals, S. Kauffman, N. Eldredge and D. Farmer with early discussions on technological evolution. This paper is dedicated to the defenders of the last barricade before the Eglise Sant-Merri. Our work has been supported by the Botin Foundation. We also thank the Santa Fe Institute, where most of this research was done.

VII. REFERENCES

1. Pagel, M (2009) Human language as a culturally transmitted replicator. Nat. Rev. Gen. 10: 405-415.

2. Mufwene S (2001) *The ecology of language evolution.* Cambridge University Press.

3. Solé R, Corominas-Murtra B, Fortuny J (2010) Diversity, competition, extinction: the ecophysics of language change. J Roy Soc Interface 7: 1647-1664.

4. G. Basalla (1989) The Evolution of Technology, Cambridge Univ. Press.

5. Arthur B (2009) *The Nature of Technology. What it is and How it Evolves.* Free Press.

6. Sole R, Valverde S, Rosas-Casals M, Kauffman SA, Farmer D, Eldredge N (2013) The evolutionary ecology of technological innovation. Complexity 18: 15-27.

7. Eldredge, N. (2011) Paleontology and cornets: Thoughts of material cultural evolution. Evo Edu Outreach 4: 364-373.

8. Johnson S. (2010) *Where Good Ideas Come From: A Natural History of Innovation.* Riverhead Press.

9. Lipo CP, O'Brien MJ, Collard M. Shennan SJ (2009) *Mapping our Ancestors.* Aldine Transaction Publishers.

10. Tëmkin I, Eldredge N (2007) Phylogenetics and material cultural evolution. Curr Antrop 48: 146-153.

11. Dunn M, Terrill A, Reesink G, Foley RA, Levinson SC (2005) Structural phylogenetics and the reconstruction of ancient language history. Science 309: 2072-2075.

12. Dagan T, Martin W (2009) Getting a better picture of microbial evolution en route to a network of genomes. Phil Trans R Soc Lond B Biol Sci 364: 2187-2196.

13. Nelson-Sathi S, List JM, Geisler H, Fangerau H, Gray RD, Martin W, Dagan T (2011) Networks uncover hidden lexical borrowing in Indo-European language evolution. Proc Royal Soc London B 278: 1794-1803.

14. Green T (2010) *Bright Boys: The Making of Information Technology.* CRC Press.

15. McNerney J, Farmer JD, Redner S, Trancik JE (2011) Role of design complexity in technology improvement. Proc Natl Acad Sci USA 108: 9008-9013.

16. Sammet JE (1969) *Programming Languages: History and Fundamentals.* Prentice-Hall, Englewood Cliffs, N.J.

17. Sammet JE (1972) Programming languages: history and future. Comm. ACM 15: 601-610.

18. O'Regan G (2008) *A brief history of computing.* Springer, London.

19. Deacon TW (1997) *The symbolic species: the coevolution of language and brain.* Norton, New York.

20. Landin PJ (1965) The next 700 programming languages. Comm. ACM 9(3),157- 65.

21. Gualdi S, Yeung CH, Zhang YC (2011) Tracing the evolution of physics on the backbone of citation networks. Physical Review E 84: 046104.

22. Lu L, Jin C-H, Zhou T (2009) Similarity index based on local paths for link prediction of complex networks. Physical Review E 80: 046122.

23. Cambell-Kelly M, Aspray W (2004) *Computer: A history of the information machine.* Perseus Books. Cambridge MA.

24. Ensmenger N (2010) *The computer boys take over: Computers, Programmers, and the Politics of Technical Expertise.* MIT Press.

25. Raup DM (1985) Mathematical models of cladogenesis. Paleobiology 11: 42-52.

26. Mace R, Holden CJ (2005) A phylogenetic approach to cultural evolution. Trends Ecol Evol 20: 116-121.

27. Krapivsky, P. L., and Redner, S. (2005) Network growth by copying, Physical Review E 71, 036118.

28. Valverde S, Solé R (2005) Logarithmic growth dynamics in software networks. Europhys Lett 72: 858-864.

29. Valverde S, Solé R (2005) Network motifs in computational networks: A case study in software architecture. Phys Rev E 72:026107.

30. Rabosky DL, Slater GJ, Alfaro ME (2012) Clade Age and Species Richness Are Decoupled Across the Eukaryotic Tree of Life. Plos Biol 10 : e1001381.

31. Kwang-Tsao, S., Sokal R. R., Tree Balance, Systematic Zoology 1990, 39(3): 266-276.

32. Kirkpatrick, M, Slatkin, M, Searching for the evolutionary patterns in the shape of a phylogenetic tree, Evolution 1993, 47: 1171-1181.

33. Cotton JA, Page RDM, The Shape of human gene family phylogenies, BMC Evol Biol 2006, 6:66.

34. Cavalli-Sforza, L. L., Edwards, A. W. F., Phylogenetic analysis: models and estimation procedures, Am. J. Hum. Genet 1967, 19: 233-257.

35. Atkinson QD, Meade A, Venditti C, Greenhill SJ, Pagel M. (2008) Languages evolve in punctuational bursts. Science 319 : 588.

36. Venditti C, Pagel M. (2008) Speciation and bursts of evolution. Evo. Edu. Outreach 1:274?280.

37. Eldredge N, Gould SJ. (1972) Punctuated equilibria: an alternative to phyletic gradualism. In: Schopf TM, editor. Models in palaeobiology. San Francisco: Freeman Cooper, pp. 82?115.

38. Gould SJ, Eldredge N. (1977) Punctuated equilibria: the tempo and mode of evolution reconsidered. Paleobiology 3 : 115?51.

39. Mokyr J. (1990) Punctuated equilibrium and technological progress. Am. Econ. Rev. 80 : 350-354.

40. Loch CH and Huberman BA (1999) A punctuated-equilibrium model of technology diffusion. Manag. Sci. 45: 160-177.

41. Levine, H. and Rheingold, H. (1987) *The cognitive connection: Thought and Language in Man and Machine.* Prentice Hall, New York.