A Cultural Diffusion Model for the Rise and Fall of Programming Languages

Sergi Valverde[1]* *and Ricard V. Solé*[1]

ABSTRACT

Our interaction with complex computing machines is mediated by programming languages (PLs), which constitute one of the major innovations in the evolution of technology. PLs allow flexible, scalable, and fast use of hardware and are largely responsible for shaping the history of information technology since the rise of computers in the 1950s. The rapid growth and impact of computers were followed closely by the development of PLs. As occurs with natural, human languages, PLs have emerged and gone extinct. There has been always a diversity of coexisting PLs that compete somewhat while occupying special niches. Here we show that the statistical patterns of language adoption, rise, and fall can be accounted for by a simple model in which a set of programmers can use several PLs, decide to use existing PLs used by other programmers, or decide not to use them. Our results highlight the influence of strong communities of practice in the diffusion of PL innovations.

The relevance of evolution as a universal framework to understand our biosphere is encapsulated in a famous quote by the evolutionary biologist Theodosius Dobzhansky: "Nothing in biology makes sense except in the light of evolution" (Dobzhansky 1973). When we turn our attention to cultural change, in particular the development of technology, we could ask ourselves what role is played by equivalent evolutionary forces. Both similarities and differences exist between natural and technological evolution (Solé et al. 2013), and moreover, it is often difficult to establish the importance of a given innovation and how it becomes widespread within a given social context.

Sometimes technological success (or failure) can be explained through a process of increasing returns, with little connection to rational decisions (Arthur 1994). This is the case of some inventions that had several alternatives in the market, such as the two famous video recording systems, VHS and Betamax. They entered the market almost simultaneously (Betamax appeared a year before VHS) and under similar conditions, yet despite the acknowledged advantages of Betamax, VHS eventually expanded and dominated the video recorder market, and Betamax went extinct (Arthur 1994).

How did that happen? How is it possible that the more advantageous option goes extinct while the less fit rises to full domination? The explanation comes from the nature of returns in an economic system where compatibility largely dominates the potential choices made by users and consumers. The more users a given technology has, the larger the chances that other users will adopt it. A direct consequence of this scenario is that social amplification leads to competition that necessarily ends in the extinction of the initially less popular option. Such a scenario can be easily described

[1]ICREA-Complex Systems Lab, Universitat Pompeu Fabra, Barcelona, Spain.

*Correspondence to: Sergi Valverde, ICREA-Complex Systems Lab, Universitat Pompeu Fabra, Barcelona Biomedical Research Park, Dr. Aiguader 88, Planta 4, D-491, Barcelona E-08003, Spain. E-mail: sergi.valverde@upf.edu.

KEY WORDS: CULTURAL EVOLUTION, MULTILINGUALISM, DIFFUSION, PROGRAMMING LANGUAGES, SOFTWARE.

Human Biology, Summer 2015, v. 87, no. 3, pp. 88–xx.

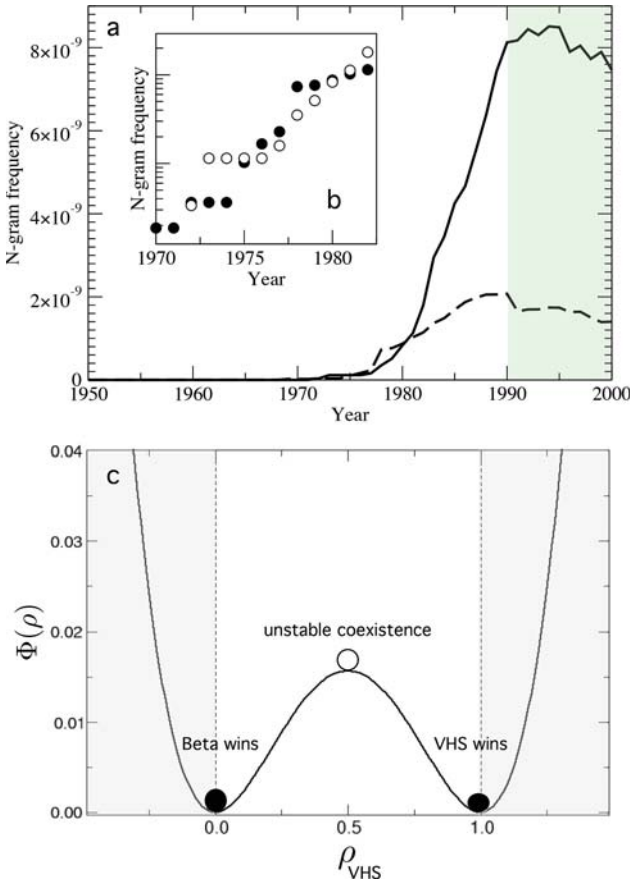

FIGURE 1. Increasing returns and symmetry breaking in technological evolution. (a and b) Time series of citations for the terms "VHS recorder" (solid line) and "Betamax" (dashed line), the two competing videotape designs that emerged almost simultaneously and became the dominant options in the market. Despite their near equivalence during the early phase (b), VHS (solid circles) eventually dominated the market. (c) Potential associated with a simple symmetric model of technological competition (see text). One unstable state (open circle) and two symmetric, stable states (solid circles) are accessible. Initial conditions and accidents can play a crucial role.

using nonlinear models where imitation dynamics provides the basic rules that drive the expansion of each option through populations of users. These examples provide a powerful illustration of how technology diffuses through society.

Although in most cases little is known about the population dynamics of each option and the exact numbers of associated users, some available data allow us to estimate these quantities for the so-called videotape format war between VHS and Betamax. We can investigate this interesting illustration using n-grams, that is, strings of written words that can be single items (e.g., "war") or more complex structures (e.g., "world war" or "World War I"). By using very large digitized databases incorporating all words that appeared in vast book libraries in different years, it is actually possible to obtain a compelling picture of cultural impact (Michel et al. 2011).[1]

Figure 1 shows the outcome of the data analysis of the videotape format war, with two specific n-grams for "VHS recorder" and "Betamax" as surrogates for the relevant technological innovations, which reveal two important trends. Figure 1a shows that, after their appearance in the mid-1970s, both terms increasingly appear but VHS outperforms Betamax over time. The early phase (Figure 1b) reveals a remarkable equivalence of both n-grams, consistent with the reduced competition that should be expected when the process is starting. This is a good example of the potential value of culturomics data.

This example can also be analyzed using a simple model of technological competition based on increasing returns. This helps illustrate the modeling approach used here to analyze the rise and fall of programming languages. Specifically, we will assume that two options are present in a given market and that these are the only ones available: VHS wins and Betamax looses versus VHS looses and Betamax wins. Additionally, users are driven by majority rules: the more users adopt a given technology, the higher is the probability that other users decide to use it too. We indicate by ρ_1 and ρ_2 the fraction of users using each option, and we assume a normalization $\rho_1 + \rho_2 = 1$.

The population dynamics of this system is defined by the set of (symmetric) equations

$$\frac{dp_1}{dt} = \mu\rho_1(\rho_1 - \rho_2) - \rho_1\Phi(\rho_1,\rho_2), \quad (1)$$

$$\frac{dp_2}{dt} = \mu\rho_2(\rho_2 - \rho_1) - \rho_2\Phi(\rho_1,\rho_2). \quad (2)$$

The first term in parentheses, $(\rho_i - \rho_j)$, stands for the relative difference between the two populations. It introduces the sign of the choice made based on which is the current dominant option. If $\rho_1 > \rho_2$, the first population will grow while the second will decrease. The last term, $\Phi(\rho_1, \rho_2)$, is the competition function that introduces how the two populations interact (Solé 2011).

It is not difficult to see that $\Phi(\rho) = \sum_{i\neq j}\mu\rho_i(1 - 2\rho_j)$, and thus we have, from $\rho_i = 1 - \rho_j$,

$$\frac{dp_i}{dt} = \mu\rho_i(2\rho_i - 1) - \mu\rho_i\sum_{i\neq j}\rho_i(1 - 2\rho_j), \quad (3)$$

which after some algebra gives a cubic equation for the dynamics of each population of users:

$$\frac{dp_i}{dt} = \Gamma(\rho_i) = 2\mu(1 - \rho_i)(2\rho_i - 1)\rho_i. \quad (4)$$

This model has three equilibrium (fixed) points: $\rho_i = ½$, which corresponds to a coexistence of identical numbers of users for each option; and two alternative states $\rho_i = 1$ and $\rho_i = 0$, both stable, which correspond to the extinction or success of the technological alternative.[2]

This is actually an example of symmetry breaking. An alternative, very helpful way of representing this phenomenon is to use the so-called potential function $V(\rho_i)$ associated with population dynamics. The potential function is closely related to energy functions in physics and is defined by the relation

$$\frac{d\rho_i}{dt} = -\left(\frac{\partial V(\rho_i)}{\partial \rho_i}\right), \quad (5)$$

which indicates that the dynamics of ρ "derives" from the potential. It can be shown that this potential can be obtained from

$$V(\rho_i) = -\int \Gamma(\rho_i)d\rho_i. \quad (6)$$

This function is such that the minima and maxima correspond to stable and unstable equilibrium states, respectively. In our example, the resulting potential is a quadratic function

$$V(\rho_i) = \mu(\rho_i^4 - 2\rho_i^3 + \rho_i^2). \quad (7)$$

Our potential function shows a standard two-well shape (Figure 1c). Once an initial fluctuation has favored one configuration over the other, the amplification of the original fluctuation forces a collective decision. If we think of the state of the system as a marble rolling down the potential landscape, and we start with an initial state where both VHS and Betamax are equally represented (Figure 1c, open circle) then two possible, symmetric alternatives are equally likely to happen (Figure 1c, solid circles). If we consider the potential for the VHS solution, then either it wins ($\rho_{VHS} = 1$) or it loses ($\rho_{VHS} = 0$).

The standard examples of technological innovation discussed above are often related to two-option choices (Betamax vs. VHS, or clockwise vs. anticlockwise clocks). But what about the diverse nature of multiple innovations that develop over time? This is in fact the realistic scenario that describes how technology evolves: many different innovations emerge and spread among users. This implies a highly complex dynamics, since (in principle) multiple parameters and historical factors might influence each particular innovation. Despite this, and similar to ecological systems composed of many interacting species, some models that account for evolution over time with almost no assumptions about parameters are highly successful in explaining many relevant laws. As we show below, this seems to be the case for one of the most important and influential classes of technological innovations: programming languages.

The Ecology of Programming Languages

Several important innovations have led to the emergence of major technological domains. If we look to the second half of the 20th century, it would be reasonable to adjust Dobzhansky's quote: "Nothing makes sense in information technology, unless under the light of programming languages." Programming languages (PLs) appeared shortly after one of the first, largest computers was built in 1946: the Electronic Numerical Integrator and Computer, or ENIAC (Burks and Burks 1981), which was used to address a broad variety of numerical

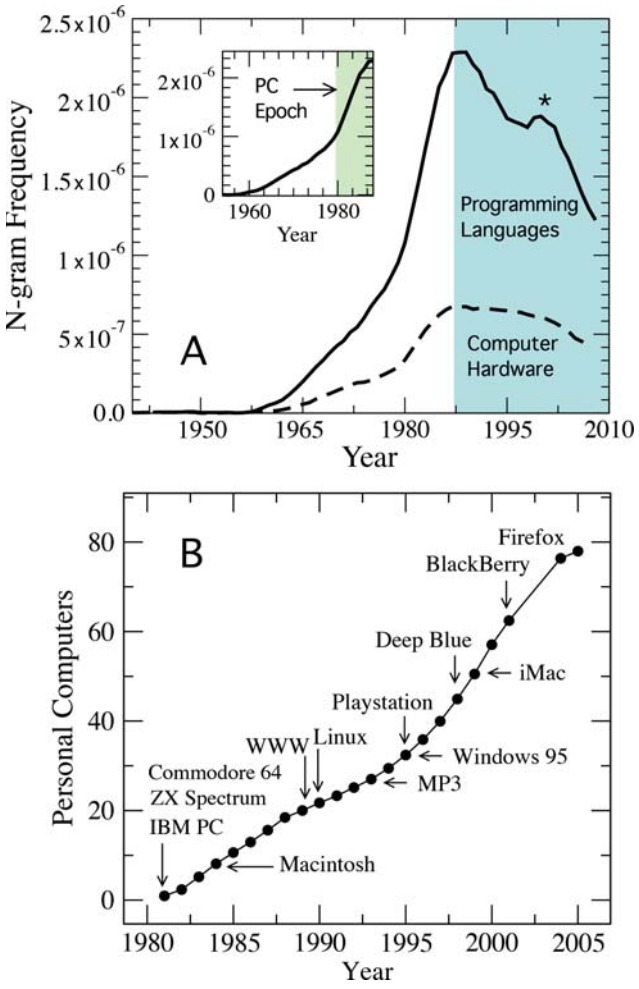

FIGURE 2. Cultural diffusion of information technologies. (A) Time series of n-gram abundance associated with the terms "programming language" (solid line) and "computer hardware" (dashed line). The curves display qualitative similar behavior, possibly mirroring the coevolution of hardware and software. A sudden increase of the abundance of "programming language" takes place around 1980 (inset). A bump (*) signals the Y2K (year 2000) problem, which affected many programming languages. The shadowed region shows a decay of "programming language" popularity with the widespread adoption of Internet technologies. (B) Sustained growth of the number of US personal computers (×100) since 1980 (data source: World Bank 2012). The arrows point to specific innovations in the history of information technology.

integration problems. Among many other scientists and engineers involved with ENIAC was the genius mathematician John von Neumann, who immediately became interested in finding a general framework for a more advanced, programmable computer, called the Electronic Discrete Variable Automatic Computer, or EDVAC (von Neumann 1945).

Thanks to von Neumann's work, computers became programmable not by changes in the hardware but by a stored program. The idea that a program could be used by different computers to execute a given task was revolutionary. Along with the evolution of hardware, a new class of "invisible" technology started to develop—software marked the rise of information technology and provided the interface for communication between machines and humans. Its enormous impact was beyond any expectations. Software was rapidly adopted as the essential part of computation, enabling humans to easily interact with machines, and triggered the emergence of communities of users sharing similar PLs (starting with FORTRAN). As a consequence, users invented new PLs that addressed problems in a variety of ways while improving access for an increasingly larger range of users. At first the spread of PLs was limited, but it quickly gained momentum as computers became smaller and cheaper. As soon as personal computers became a reality in the 1980s, multiple PLs appeared and were used by large communities of programmers, with PLs diffusing through them.

Figure 2A shows the result of analyzing the terms "programming language" and "computer hardware." Shortly after their emergence in the 1940s they both rarely appear, but just before 1960 they start to grow rapidly, then climb at a given rate, accelerate around 1980, and decline after the 1990s. The "programming language" n-gram experiences faster growth and decline compared

with "computer hardware." These time series suggest two relevant things. The first is the coupling of PLs and hardware over the growth phase of the system, including the change in growth rate that took place in the 1980s. This coupling is consistent with a coevolutionary dynamics between hardware and software that existed throughout the history of information technology. We know that a large portion of the increasing impact of PLs relates to their growing number and to their widespread importance. Afterward, the rapid decline of PLs suggests that they no longer were as central. Instead, globalization and increasing returns associated with a limited, worldwide adoption of a limited number of computing devices shrank their numbers (see Figure 2B). In this context, PLs are similar to the video recording inventions discussed above.

As we show below, a simple model of PL evolution based on cultural diffusion can explain this and other statistical patterns. PLs follow the same basic pattern of expansion as other technological artifacts. In fact, because PLs have a special status (they are neither hardware nor simple programs), their adoption is an important decision that is particularly sensitive to compatibility constraints. An additional component also needs to be considered to understand the dynamics of PLs over time: multilingualism. Very often, programmers know and use several PLs simultaneously. As occurs with human languages, the spread and success of PLs is influenced by the presence of large numbers of users keen on using them and limited by a finite repertoire of PLs that can be adopted and usefully applied by individuals.

Frequency Distributions

Although most of the early story of PLs is lost, particularly the numbers of users, the more recent record provides an interesting illustration of how populations of PL users expand or shrink over time. Using these data, usually measuring the impact of extant PLs, we capture the popularity of each PL, as well as underlying social, economic, and technological factors. The most recent historical data shows that some PLs are on the rise, for example, Objective-C used in iOS applications, which is becoming more popular thanks to the commercial success of the iPhone; meanwhile, others are decaying, such as Perl (see below). Yet others have lower impact but keep steady levels of popularity, perhaps because of their importance for specific communities (e.g., Javascript). Does any law or universal behavior drive popularity of PLs?

Ranking programming popularity is a very difficult task. The measurement of popularity is affected by common problems similar to most market studies. Popular measures estimate PL impact as a weighted combination of the reported number of hits reported by search engines. Several measures have been published, such as the TIOBE programming community index (http://www.tiobe.com), the PYPL Popularity of Programming Language index (http://pypl.github.io), and the Transparent Language Popularity Index analyzed here (http://lang-index.sourceforge.net). These measures capture different economic and social factors affecting the popularity of specific PLs reflected on the Internet. For example, Figure 3 shows the frequency-rank distribution of PL popularity, ordering all the measured PLs from the most abundant (rank $r = 1$) to the least frequent. In this way we obtain a decreasing distribution that appears highly skewed. This distribution provides a snapshot of the popularity changes experienced by PLs.

Our distribution is consistent with a discrete generalized beta distribution (DGBD) (Martinez-Mekler et al. 2009) that well fits frequency-rank plots of PL popularity:

$$f(r) = \frac{A}{r^a}(r + 1 - r)^b, \tag{8}$$

where r is the PL rank, R is the maximum rank value, A is a normalization constant, and a and b are two fitting exponents. Martinez-Mekler et al. (2009) presented a growth probabilistic model that generates data complying with this distribution. The model represents a competition between two processes: permanence (expansion) and change (point mutations). This model provides an intuitive interpretation for the DGBD parameters: when $a > b$, point mutations are rare and expansion is favored; $a < b$ corresponds to the opposite situation, with prevalent disorder. However, their model does not predict the values observed here: $(a, b) = (1.44, 0.46)$ (see Figure 3).

The DGBD pattern shown in Figure 3 provides

a statistical, global view of PLs at a given point in time (July 2013). It reveals an overabundance of rare PLs and the presence of a limited number of highly abundant PLs. This means that rarity is the rule: most extant PLs are used by a small fraction of users. On the other hand, this pattern is described by a continuous distribution; that is, it is characterized by a long tail and does not display a two-regime pattern between common and rare PLs. This type of scaling behavior is known to occur in a vast number of systems displaying power laws (Solé and Goodwin 2001). It reveals the presence of amplification phenomena that often can be explained by very simple rules (Ball 2004). More important, scaling laws are also well known in the distribution of human languages (Solé et al. 2010).

Human languages exhibit enormous diversity, but they also display the skewed distribution of abundances shown by PLs. This pattern reveals again the uneven use of tongues in our planet, largely dominated by a small number of languages, with an enormous number of rare ones defining the tail of the distribution. The rarity of human languages reveals an important trait: many extant languages will face extinction in the next decades. Languages, as with species, disappear once they are no longer propagated. Despite their differences (Solé et al. 2010), both languages and species need to confront the conflict associated with survival under competition for resources. For both human languages and PLs, "resources" means users. Once a tongue is no longer spoken, we consider it extinct. Similarly, if no more programs are written in a given PL, it also is extinct.

The next section presents a simple model of PL evolution that allows us to recover the statistical distribution of PLs at a given step characterized by high diversity, as well as the time evolution of the total number of PLs.

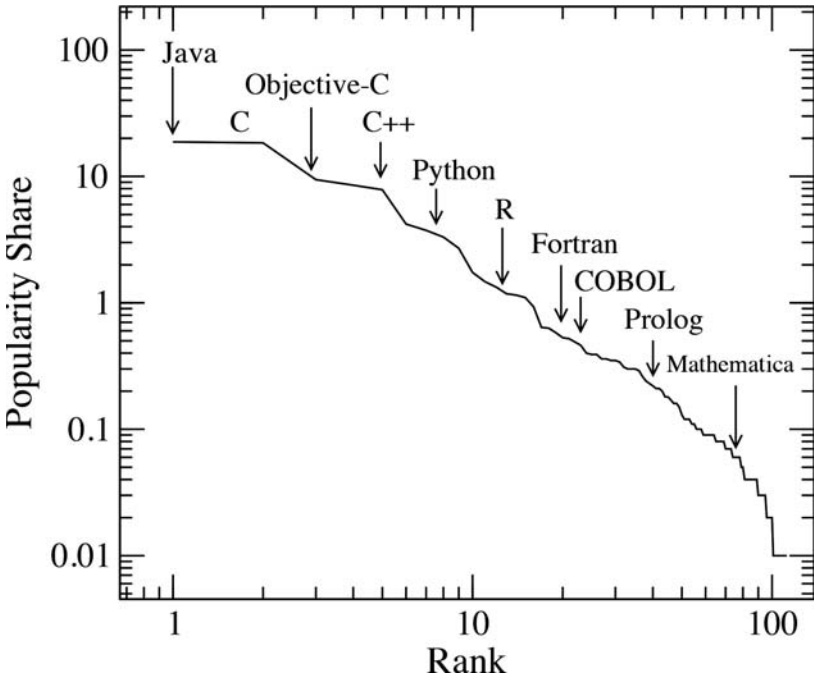

FIGURE 3. Log-log plot of frequency rank, ordered by descending popularity, for programming languages (PLs) listed in the Transparent Language Popularity Index (http://lang-index.sourceforge.net). This distribution follows the discrete generalized beta distribution, with parameters (a, b) = (1.44, 0.46). The arrows point to the rank position of well-known PLs that differ greatly in nature and duration.

Discrete Diffusion Model of Programming Languages

This section examines cellular automata that allow us to simulate how cultural diffusion takes place in programmer communities. (A Netlogo implementation of this model is freely available at http://svalver.github.io/Proglang/pldiffusion.html.) Our approach does not focus on specific details but, rather, considers the way that any PL is likely to emerge, spread, and disappear. Modeling PL evolution requires a number of simplifications and strong assumptions. PL evolution is marked by changes over time as PLs rise and fall. We will assume that users are completely identical, thus ignoring the presence of different types of communities. We will also ignore the fact that PLs and the machines they interact with them coevolve. These assumptions might seem too strong, but they allow simple models to be built and, very often, succeed. Despite such oversimplification, minimal models that describe the evolution and statistical patterns of both languages and species in ecosystems are able to explain observed regularities.

Perhaps the best example is the so-called neutral theory of ecology (Hubbell 2001; Solé et al. 2004; Alonso et al. 2006), which assumes that all species within a given habitat are identical in all their birth and death characteristics. The seemingly universal pattern of rank abundance of species in a given ecosystem is obtained in a robust manner. In this context, neutral models can also explain the distribution of cultural variants (Bentley et al. 2004).

As with systems mentioned above, we can reproduce the empirical frequency-rank distribution of PL popularity with a small number of

assumptions (see below). The success of these approximations is grounded in the universal dynamics exhibited by systems allowing the amplification of fluctuations. Despite the roles played by memory, space, hierarchies, social dynamics, or demographic parameters, most of these properties have little impact on the statistical patterns of organization. First, the popularity of PLs depends only on the presence or absence of other PLs. Popular PLs are more frequently adopted than rare ones, which are more likely to be forgotten and disappear. Second, we simulate a homogeneous population of software developers that can adopt several PLs simultaneously and have a reasonable understanding of PL features, including design principles and implementation details.

The model consists of a static population of programmers located on an $L \times L$ lattice and a fixed pool of PLs indexed by a set

$$\Sigma = \{0, \ldots, \mu\} \tag{9}$$

(0 denotes the null language). Similar to Axelrod's (1997) model of cultural dissemination, the programming culture of a developer is described by a vector that holds a certain number of PLs. The state of the programmer located at site (i,j) is

$$\overrightarrow{s_{ij}} = (s_{ij}^1, s_{ij}^2, \ldots, s_{ij}^M) \in \Sigma^M, \tag{10}$$

where $s_{ij}^k \in \Sigma$ indicates the index of the kth PL known by the programmer. Every developer can adopt up to M different PLs, that is, for all $1 \le k \ne l \le M$. In addition, we will count how many programmers know the rth PL:

$$N_p(r) = \sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{k=1}^{M}\delta(s_{ij}^k, r), \tag{11}$$

where $\delta(x, y) = 1$ if $x = y$, and 0 otherwise. At the beginning of the simulation, which corresponds to the year 1958, there are no PLs, and thus $\vec{s_{ij}} = \vec{0}$ for all programmers. At every time step t, choose one random site (i,j) of the lattice and apply rules for innovation, adoption, and forgetting, as follows.

Innovation Rule

This rule is associated with the "discovery" by a programmer of a previously unused PL. It thus acts as an external input into the system, similar to the immigration of an individual into a given ecosystem (a similar rule was proposed in Bentley et al. 2004). Randomly choose the PL index $r \in \Sigma$ and $1 \le k \le M$, and set $s_{ij}^k(t+1) = r$ with probability α provided that the rth PL does not belong to $\vec{s_{ij}}$; that is, it is unknown to the programmer located at site (i,j).

Adoption Rule

This is the rule that introduces the diffusion of PLs in our model. It is a contact-like process, where two programmers that interact with each other (here simply by being neighbors in our two-dimensional lattice) allows an "infection" to occur. The Moore neighborhood is defined by $\Omega(i,j) = \{(u,v) : |u-i| \le 1, |v-j| \le 1\}$. For each programmer, we choose one random neighbor site $(u,v) \in \Omega(i,j)$ and set $s_{ij}^q = s_{uv}^r$ with probability

$$P[s_{uv}^r \to s_{ij}^q] = \eta\left(\frac{N_p(s_{uv}^r)}{Z}\right), \tag{12}$$

where Z is the normalization constant

$$Z = \sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{k=1}^{M}\theta(s_{ij}^k),$$

$1 \le q \le M$ is a memory slot, and $\eta > 0$ is the adoption rate.

Forgetting Rule

For different reasons, users might eventually discard a given PL from their list of potential PLs. This rule corresponds to the extinction of an individual of a given species but differs in a fundamental way: the user will retain other PLs. Randomly choose $1 \le r \le M$, and set $s_{ij}^r(t+1) = 0$ with probability

$$P[s_{ij}^r \to 0] = \delta\left[1 - \frac{N_p(s_{uv}^r)}{Z}\right], \tag{13}$$

where Z is a normalization constant and $\delta > 0$ is the forgetting rule.

Our model assumes that PLs are discovered at a constant, small rate α (innovation rule). Very popular PLs, with $N_p(s_{ij}^r)/Z \approx 1$, tend to be adopted more frequently (adoption rule), while rare PLs are easier to forget and abandon by the community (forgetting rule).

The model consistently reproduces the patterns displayed by PLs, including the ups and downs of abundances that somewhat map into the "popularity" of PLs. Common PLs have smoother time fluctuations, whereas more rapid fluctuations

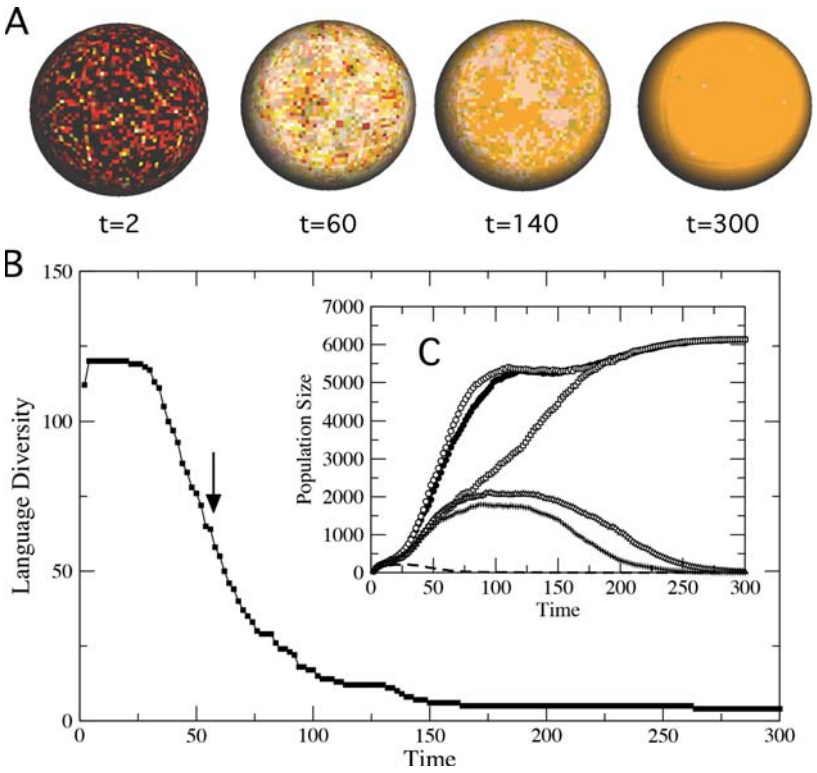

FIGURE 4. (A) Four snapshots of the cellular automaton model of programming language (PL) spread on a two-dimensional lattice. Each site represents a programmer, who can use/store up to $M = 3$ PLs. The RGB color scheme encodes the multiple (up to $M = 3$) languages adopted by a programmer. Different colors represent combinations of different PLs, which captures the PL population diversity. (B) Time series of the number of different PLs present in the population. The number grows very quickly at the beginning ($t = 2$), where most programmers only know one (A, dark red) or zero (A, black) PLs. High diversity of PLs plateaus at $t \approx 50$, which is represented in A by different shades of red and yellow sites (at $t = 60$). With time, $M = 3$ dominant PLs emerges, and a significant loss of diversity follows. According to this model, we are now located in the decay phase (arrow at $t = 60$ as in A). (C) Times series of number of adopters for five individual PLs (represented by different symbols), including three that survive at the end of the diffusion process. Individual behavior in our model is typical of competition dynamics in other technological systems (see Figures 1 and 2).

are associated with rare PLs that often end in extinction. Figure 4A displays four spatial snapshots of our cellular automaton model corresponding to the PL time series. In the first stages of the simulation ($t = 2$), relatively small patches indicate the diffusion of PLs among limited numbers of users. However, as time proceeds, larger patches can be observed resulting from the successful propagation of some PLs and their combinations. This process is also visualized in Figure 4C, where the population size for some of the most abundant PLs is plotted against time. We can see that all these PLs start by growing, but some eventually decline and become extinct, whereas others succeed.

The model exhibits a marked increase in PL diversity, and it also reveals the presence of scaling laws. Figure 5 shows how our model predicts the frequency-rank distribution observed for PLs. We have used an evolutionary algorithm to obtain the best-fitting parameters $\alpha = 0.495987$, $\eta = 0.905577$, and $\delta = 0.000195$. The maximum PL diversity is $\mu = 120$ because of the limitations of the data set. Other sources (e.g., *Wikipedia*) report many more PLs (Valverde and Solé 2015). Still, our experiments show that the behavior of our model is robust to a wide range of μ, M, and L values. Here, we have fixed programmer capacity $M = 3$ and lattice size $L = 32$. As observed with the popularity measures for extant PLs, at a given step a large number of PLs are represented by a small number of users, whereas a limited set of PLs are adopted by a majority.

These results reveal that the fate of our PL system, despite the transient richness that matches the previously presented scaling laws, is to reduce the number of PLs as the spread rule dominates. Inevitably, a low diversity of PLs will be observed in the future. The three snapshots and frequency

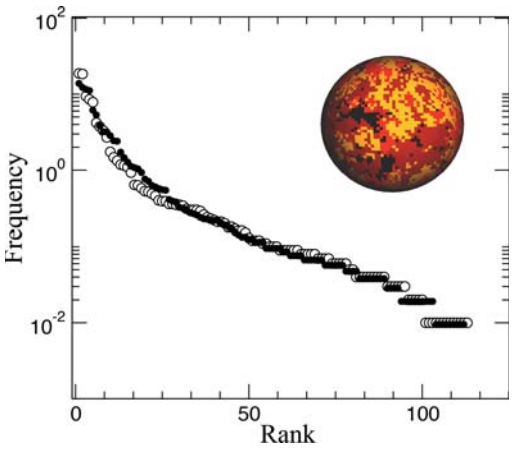

FIGURE 5. Frequency-rank distribution in programming language (PL) popularity: log-linear plot of PLs ordered by observed popularity (open circles; data from Figure 3) and as predicted by our model (solid circles). Our model fits the observed distribution and predicts the existence of spatially heterogeneous communities (color inset; see Figure 4A). Parameters: $\alpha = 0.495987$, $\eta = 0.905577$, $\delta = 0.000195$, $\mu = 120$, $M = 3$, $L = 32$.

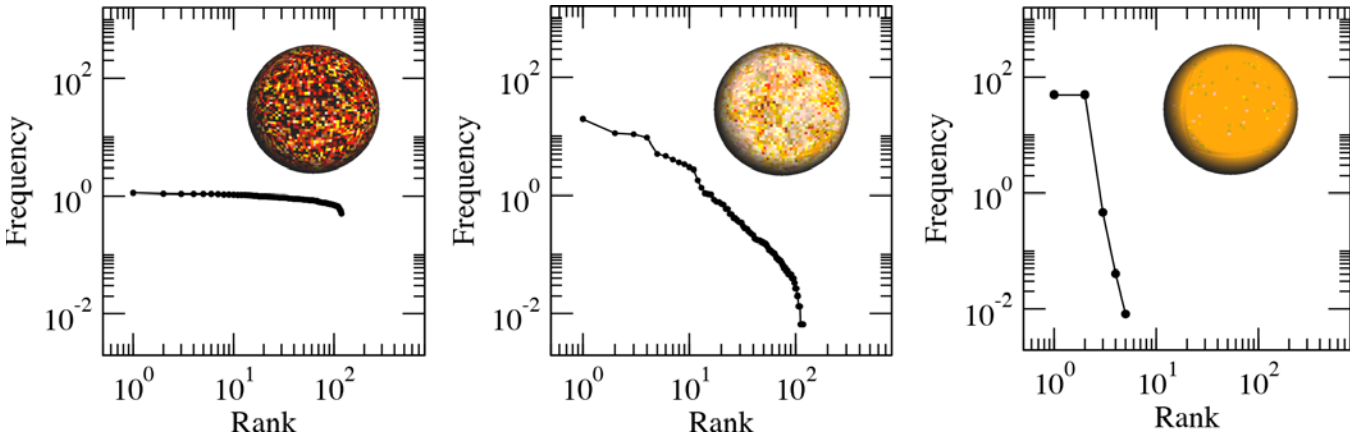

FIGURE 6. Time sequence of programming language (PL) diffusion using our model. At the beginning ($t = 4$; left) we have PL coexistence without constraints, which corresponds to an almost flat rank-frequency distribution. Dark colors indicate that programmers have free memory slots. At $t \approx 60$ (middle), we recover the observed frequency-rank distribution because different PLs are competing for programmer infection. This corresponds to a heterogeneous spatial structure. At longer time scales ($t = 250$; right) the system stabilizes in a homogeneous configuration with a few ($M = 3$) dominant PLs (many sites display the same color). Indeed, limited memory slots and feedback effects have destroyed the initial PL diversity. Parameters: $\alpha = 0.495987$, $\eta = 0.905577$, $\delta = 0.000195$, $\mu = 120$, $M = 3$, $L = 32$. We have assumed that significant communities have at least 20 members.

distributions displayed in Figure 6 indicate what can be expected under our assumptions. Starting from a more or less homogeneous distribution (other conditions can be used), after a transient period the emergence of power-law behavior of a virtual world sustains many PLs. Eventually, the same process concludes with a stark impoverishment of our PL set, with just a handful of surviving PLs.

The model thus makes a strong prediction: in the future, as globalized communication and increasing returns accelerate, few PLs will survive. In this context, many examples exist of once successful and widespread PLs that become extinct. As with many spoken languages, we are constantly facing this pattern with PLs. A recent case is provided by Perl, a PL invented in 1987 that became the essential web-writing PL during the early days of the Internet. Despite its great impact, Perl is fading toward extinction. As pointed out by Doug Barry (2014): "For programmers who already know Perl, the consensus seems to be that it's still a useful skill to have. However, for programmers coming of age in the IT industry right now, taking the trouble to learn a difficult language . . . is akin to learning Latin in a world increasingly filled with other, newer versions of Latin like French, Spanish, and Italian. Perl may not 'go extinct' . . . , but it will mostly become an anachronism." The comparison here between PLs and spoken languages here is interesting—in several ways they are similar: they both face the impact of competition among coexisting options and the law of increasing returns. However, as discussed below, PLs are more affected by these amplification phenomena, and their fate and future diversity reduction are likely to happen in a much shorter time scale than spoken languages.

Discussion

People's lives are increasingly dependent on the correct working of software, because as wireless networks proliferate, many portable devices become interconnected. The increasing popularity of the personal computer has been followed by a consistent trend toward more complex software systems. In that respect, the expansion of PLs over decades of change shares some traits with human languages and differs in some important aspects. Different human languages arise, coexist, and fall, but they are all characterized the essentially same complexity provided by syntax. The diffusion of human languages correlates directly with the number of speakers, but they are all essentially equal in structure. PLs, in contrast, can be classified in two major categories, procedural and declarative, and because of their distinct goals and underlying

hardware to be executed, they can display different levels of complexity (Scott 2009).

PLs are one of the main drivers of successful software development. They are a means both to tell computers how to do useful work and to exchange ideas and algorithms between people. The evolution of PLs is linked to both the software environment (hardware) and socioeconomic factors (and the exponential decline in hardware costs). Computers have transitioned from costly, restricted machines to inexpensive, mass-market products. This transition has been associated with deep changes in PL design. The first PLs were specifically designed for expensive machines and not for human beings. However, rapid advances in hardware technology called for more efficient ways of dealing with the (increasing) complexity of computer programming. In this context, designers of high-level PLs are much more concerned with language expressiveness and simplicity than with hardware efficiency. That is, cognitive factors and the social dynamics of PL adoption appear to be much more important than hardware aspects for the future evolution of PLs.

Here we have discussed a very simple model of cultural diffusion capable of reproducing the empirical rank distribution of PL popularity. Since our globalized world and, in particular, the ecosystem formed by PL users are finite, and since the spread of innovations is very fast, candidate PLs rapidly experience a strong selection process. Users will adopt a PL that it is not too difficult to learn, is shared by other users (the more the better), and is compatible with existing hardware requirements. Because increasing returns play a very important role here, and because homogenization is growing with globalization, PLs are destined to be less and less diverse. Compared with a spoken language, where several factors, such a special status, might favor persistence despite competitive forces (Solé et al. 2010), the social components of PLs are not as crucial to the programmer. Such lack of special status and rapidly changing hardware might contribute, indeed accelerate, this decline in diversity.

Future work should extend this approach to understand the effect of population heterogeneity and specific features of PLs. Our model assumes that all PLs are equally learnable and that adoption rate depends only on PL popularity. However, expert developers can learn new PLs faster than can novices. Computer programming, like reading, involves a number of different, interrelated skills. Some PLs have been designed for teaching (e.g., BASIC), while others are purposely complicated (so-called esoteric languages, like Ook!). An important (and largely unexplored) question is how the cognitive demands of learning (Pea and Kurland 1984) affect the dynamics of PL adoption. In this context, extensions of modeling approaches like the ones discussed here will help us to explore these questions and, more generally, help us understand the societal impact of information technologies.

ACKNOWLEDGMENTS

We thank Mark Watney and the members of the Complex Systems Lab for useful discussions. This work has been supported by Fundación Botin, by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund (grant FIS2013-44674-P to S.V.), and by the Santa Fe Institute.

NOTES

1. This defines a new field, "culturomics," which is a powerful quantitative way to measure cultural impact. The culturomics approach allows us to glimpse the role played by given concepts or artifacts using numbers of citations as surrogates for their relative importance in a given time window.
2. In general, the three equilibrium points are obtained from the condition $d\rho_i/dt = 0$ or, equivalently, from $F(\rho_i^*) = 0$. The type of stability is defined by the sign of $\lambda = \partial\Gamma(\rho_i^*)/\partial\rho_i$, to be calculated for each fixed point. If $\lambda < 0$ the point is stable, meaning that the system will return to the point if a small perturbation is applied. If $\lambda > 0$ it is unstable; a small perturbation from this point is amplified, and the system abandons it.

Received XXX; revision accepted for publication XXX.

LITERATURE CITED

Alonso, D., R. S. Etienne, and A. J. McKane. 2006. The merits of neutral theory. *Trends Ecol. Evol.* 21:451–457.

Arthur, B. 1994. *Increasing Returns and Path Dependence in the Economy.* Ann Arbor: Michigan University Press.

Axelrod, R. 1997. The dissemination of culture: A model with local convergence and global polarization. *J. Conflict*

Resolut. 41:203–226.

Ball, P. 2004. *Critical Mass: How One Thing Leads to Another.* New York: Farrar, Straus, and Giroux.

Barry, D. 2014. The slow, stubborn extinction of the Perl programming language. *Tech of the Town* (blog), 7 August, http://resolvit.com/Editor/NewsandBlogs/NewsandBlogs-FullArticle/tabid/427/ArticleId/701/language/en-US/The-Slow-Stubborn-Extinction-of-the-Perl-Programming-Language.aspx.

Bentley, R. A., M. W. Hahn, and S. J. Shennan. 2004. Random drift and culture change. *Proc. Biol. Sci.* 271:1,443–1,450.

Burks, A. W., and A. R. Burks. 1981. The ENIAC: First general-purpose electronic computer. *IEEE Ann. Hist. Comput.* 3:310–389.

Dobzhansky, T. 1973. Nothing in biology makes sense except in the light of evolution. *Am. Biol. Teacher* 35:125–129.

Hubbell, S. P. 2001. *The Unified Neutral Theory of Biodiversity and Biogeography.* Princeton, NJ: Princeton University Press.

Martinez-Mekler, G., R. A. Martínez, M. B. del Ro et al. 2009. Universality of rank-ordering distributions in the arts and sciences *PLoS One* 4:e4791.

Michel, J.-B., Y. K. Shen, A. P. Aiden et al. 2011. Quantitative analysis of culture using millions of digitized books. *Science* 331:176–182.

Pea, R. D., and D. M. Kurland. 1984. On the cognitive effects of learning computer programming. *New Ideas Psychol.* 2:137–168.

Scott, M. L. 2009. *Programming Language Pragmatics*. 3rd ed. Burlington, MA: Morgan Kaufmann.

Solé, R. V. 2011. *Phase Transitions.* Princeton, NJ: Princeton University Press.

Solé, R. V., D. Alonso, and J. Saldaña. 2004. Habitat fragmentation and biodiversity collapse in neutral communities. *Ecol. Complex.* 1:65–75.

Solé, R. V., B. Corominas-Murtra, and J. Fortuny. 2010. Diversity, competition, extinction: The econophysics of language change. *J. R. Soc. Interface* 7:1,647–1,664.

Solé, R. V., and B. Goodwin. 2001. *Signs of Life: How Complexity Pervades Biology*. New York: Basic Books.

Solé, R. V., S. Valverde, M. Rosas-Casals et al. 2013. The evolutionary ecology of technological innovation. *Complexity* 18:15–27.

Valverde, S., and R. V. Solé. 2015. Punctuated equilibrium in the large-scale evolution of programming languages. *J. R. Soc. Interface* 12:20150249.

von Neumann, J. 1945. First draft of a report on the EDVAC. Reprinted in *IEEE Ann. Hist. Comput.* 15:27–75, 1993.

World Bank. 2012. World DataBank, http://databank.worldbank.org.