6-1-2015

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

Sergi Valverde
ICREA-Complex Systems Lab, Universitat Pompeu Fabra, sergi.valverde@upf.edu

Ricard V. Solé
ICREA-Complex Systems Lab, Universitat Pompeu Fabra, ricard.sole@upf.edu

A cultural diffusion model for the rise and fall of programming languages

Sergi Valverde and Ricard V. Solé

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 allowed a 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 was followed closely by the development of programming languages. As it occurs with natural, human languages, they emerged and got extinct. There has been always a diversity of coexisting PLs that somewhat compete among them, while occupying special niches. Here we show that the statistical patterns of language adoption and the pattern of rise and fall can be accounted for by means of a simple model where a set of programmers can use several PLs, decide to use existing ones used by other programmers or decide not to use them. Our results highlights the influence of strong communities of practice in the diffusion of PL innovations.

Keywords: Cultural evolution, multilingualism, diffusion, programming languages, software.

I. INTRODUCTION

The relevance of evolution as a universal framework to understand our biosphere is encapsulated in the famous quote of Dobzhansky: "nothing makes sense except under the light of evolution". When we turn our attention towards cultural change, and in particular the development of technology, we could ask ourselves what is the equivalent role played by evolutionary forces. Both similarities and differences exist between natural and technological evolution (Solé et al, 2013) and moreover it is oftentimes 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 with 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, namely VHS and Beta. Despite that they both started to compete in the market almost simultaneously (Beta appeared a year before VHS) and under similar conditions, and despite the acknowledged advantages of Beta, VHS eventually expanded and dominated the video recorder market while Beta got extinct (Arthur, 1994).

How did that happen? How is it possible that the less advantageous option becomes 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 adopt it. A direct consequence of this scenario is that the social amplification process leads to a competition process that necessarily ends in the extinction of the initially less popular option. Such scenario can be easily described 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 what are the exact numbers of users associated to each one, some available data allow to estimate these quantities. The so called "videotape format war" provides an interesting illustration, which we can investigate by the use of so called n-grams, i. e. strings of written words that can be single items (such as "war") or more complex structure (such as "world war" or "World War I"). By using very large digitised databases incorporating all words that appeared in vast book libraries in different years, it is actually possible to obtain a compelling picture of its cultural impact (Michel et al., 2011)[1]. In figure 1 we show the outcome of this data analysis by using two specific n-grams, namely "VHS recorder" and "betamax" as surrogates of the relevant technological innovations.

The picture reveals two important trends. The larger plot (figure 1a) shows that, after their appearance in the mid 1970s, they appear more and more cited but VHS outperforms Beta over time. On the other hand, the early phase of this process reveals a remarkable coexistence of both n-grams (figure 1b) consistently with the reduced competition that should be expected when the process is starting. This is a good example of the potential value of culturomic data.

The previous example can also be analysed using a simple model of technological competition based on increasing returns. This will help illustrate the modelling approach used here to analyse 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. Additionally, users are driven by majority rules: the more users adopt a given technology, the higher is the probability that other users might decide to use it too. We indicate by ρ_1 and ρ_2 the fraction of users using each

[1] This allows to define a new field, so called “culturomics” that is a powerful quantitative way of measuring cultural impact. The culturomics approach allows to get a first glimpse into the role played by given concepts or artefacts using number of citations as a surrogate of their relative importance in a given time window.

option and we assume a normalisation $\rho_1+\rho_2 =1$. The population dynamics of this system is defined by a set of (symmetric) equations

$dt = \mu\rho_1 (\rho_1 - \rho_2) - \rho_1\Phi(\rho_1,\rho_2)$ (1)
$dt = \mu\rho_2 (\rho_2 - \rho_1) - \rho_2\Phi(\rho_1,\rho_2)$ (2)

The first term within parenthesis in the right hand side of both equations stands for the relative difference between the two populations. It introduces the sign of the choice made based on who is the current dominant option. If $\rho_1 > \rho_2$, the first population will grow while the second will decrease. The last term in both equations, $\Phi(\rho_1,\rho_2)$, 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 $\rho_i = 1 - \rho_j$:

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

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

$$\frac{d\rho_i}{dt} = \Gamma(\rho_i) = 2\mu(1 - \rho_i) (2\rho_i - 1) \rho_i \qquad (4)$$

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

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 (ρ_i) associated to the previous dynamics. The potential function is closely related to energy functions in physics and is defined by means of the relation:

[2] In general, they 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. Instead, for $\lambda > 0$ it is unstable, and a small perturbation from this point is amplified and the system abandons it.

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

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

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

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

$$V(\rho_i) = \mu\left(\rho_i^4 - 2\rho_i^3 + \rho_i^2\right) \qquad (7)$$

Our potential function shows a standard two-well shape (figure 1c). Once an initial fluctuation has favoured a given configuration over the second, the amplification of the original fluctuation forces a collective decision. If we think in the state of the system as a marble rolling down on the potential landscape, and we start with an initial state where both VHS and Beta are equally represented (open circle in figure 1c) then two possible, symmetric alternatives are equally likely to happen (the two filled circles in figure 1c). If we consider the potential for the VHS solution, then either it wins (ρ_{VHS}= 1) or fails (ρ_{VHS}= 0).

The standard examples of technological innovation discussed above are often related to two-option choices (Beta against VHS or clockwise versus anti-clockwise clocks). And what about the diverse nature of multiple innovations that develop over time? This is in fact the realistic scenario that describes how technology evolves: not two, but 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. And yet, as it occurs in ecological systems composed of many interacting species, some models that account for their time evolution with almost no assumptions about parameters are highly successful in explaining many relevant laws. As will be shown below, this seems the case for one of the most important and influential class of technological innovations: programming languages (PLs).

II. THE ECOLOGY OF PROGRAMMING LANGUAGES

Several important innovations have made possible the emergence of major technological domains. But if we look to the second half of the XXth century, it would be reasonable to affirm that "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: the ENIAC, standing for "Electronic Numerical Integrator And Computer" (Burks et al., 1981) and used to address a broad variety of numerical integration problems. Among many other scientists and engineers involved with ENIAC was a genius mathematician, John von Neumann, who was a consultant and immediately became interested in the problem of finding a general framework for a more advanced, programmable computer, to be named EDVAC for "Electronic Discrete Variable Automatic Computer" (von Neumann, 1993).

Thanks to von Neumann's work, computers became programmable not by means of changes in the hardware (until then) but by means of a stored program. The proposal of a program that could be used by different computers to execute a given task was a revolutionary idea. Along with the evolution of hardware, a new class of technology that was invisible started to develop. Software marked the rise of IT technology and provided the language interface to communicate machines and humans. It had an enormous impact, beyond any expectations. It was rapidly adopted as the essential part of computation, make possible to easily interact with machines and rapidly triggered the emergence of communities of users sharing similar PLs (starting with FORTRAN). As a consequence, users invented new PLs that made possible to address problems in a variety of ways while improving the access of an increasingly larger range of users. The spread of PLs was limited right at the beginning 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.

In figure 2 we show the result of this analysis when using the terms "programming languages" and "computer hardware". We can see that, after their emergence in the 1940s, they both are rarely cited but start to rapidly grow shortly before 1960 and climb at a given rate but also experience an acceleration around 1980. They kept growing but a decline occurs after the 1990s. As we can see in figure 2A, the PL n-gram experiences a rapid growth followed be a rapid decline, particularly when compared with the much less declining n-gram associated to the growth of hardware. These time series suggest two relevant things. The first is the coupling between both 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, which influenced each other through all the history of information technology. We know that a large part of this increasing impact of PLs is related to an expansion of their number as well as to their widespread importance. Afterwards, the rapid decline of PLs suggests that they no

longer were so central. Instead, globalisation and the increasing returns associated to a limited, worldwide adoption of a limited number of computing devices made their numbers shrink (see figure 2B). In this context, PLs are similar to the inventions discussed above.

As will be shown below, a simple model of PL evolution based on cultural diffusion can explain this and other statistical patterns. They follow the same basic pattern of expansion followed by other technological artefacts. In fact, because programming languages have a special status (they are not hardware nor simple programs) their adoption is a rather important decision that is particularly sensitive to compatibility constraints. An additional component will need to be considered too in order to understand the dynamics of PLs over time: multilingualism. Very often, programmers know and use several PLs simultaneously. As it occurs with human languages, their spread and success is influenced by the presence of large numbers of users keen of using them and limited by a finite repertoire of PLs that can be adopted and usefully applied by individuals.

III. FREQUENCY DISTRIBUTIONS

Although most of the early story of PLs is lost, particularly the one associated to numbers of users, the more recent record is available and provides an interesting illustration of how PLs expand or shrink their populations of users over time. Using these data, usually measuring the impact of extant PLs, we capture the popularity of each PL as well as underlying social, economical and technological factors. If we look at the most recent historical data, it is possible to see that some languages are on the rise, e.g., Objective-C used in iOS applications is becoming more popular thanks to the commercial success of the iPhone, while others are decaying, like Perl (see below). Others have lower impact but keep steady levels of popularity perhaps because of their importance for specific communities (Javascript). Is there any law or universal behaviour driving popularity of programming languages?

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 programming language impact as a weighted combination of the reported number of hits reported by search engines. Several measures have been published, e.g., the *TIOBE programming community index* (http://www.tiobe.com), the *PYPL PopularitY of Programming Language* (http://pypl.github.io) and the *Transparent Language Popularity Index* (http://lang-index.sourceforge.net) (analysed here). These measures capture different economic and social factors affecting the popularity of specific programming languages in the internet. For example, figure 3 shows the frequency-rank distribution of PL popularity. Here we order all the measured PLs from the most abundant (rank $r = 1$), the second most frequent ($r = 2$) and so on until the least

frequent. In this way we obtain a decreasing distribution that appears highly skewed. This distribution provides a frozen picture of the popularity changes experienced by PLs.

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

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

where r is the PL rank, R is the maximum rank value, A is a normalisation constant and (a,b) two fitting exponents. (Martinez-Mekler et al, 2009) presented a growth probabilistic model that generates data complying 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 and $a < b$ 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).

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

Human languages exhibit an 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 and with an enormous amount 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 it occurs with species, disappear once they are not propagated anymore. Despite the differences (Solé et al., 2010) both languages and species need to confront the conflict associated to survival under competition for resources. For both human and programming languages, resources means users. Once a tongue is not spoken anymore, we consider it extinct. Similarly, if no more programs are written in a given PL, it is also gone.

In the next section, we present a simple model of PL evolution that allows us to recover the statistical distribution of languages at a given step characterised by a high diversity, but also the time evolution of the total number of PLs.

IV. DISCRETE DIFFUSION MODEL OF PROGRAMMING LANGUAGES

In this section, we will study a cellular automata that allows us to simulate how cultural diffusion takes place in programmer communities. A Netlogo implementation of this model is freely available at the following web site http://svalver.github.io/Proglang/pldiffusion.html. Our approach does not focus on specific details but rather considers the way that any language is likely to emerge, spread and disappear. Modelling the evolution of PLs requires a number of simplifications and strong assumptions. As we saw at the beginning, language evolution was marked by some changes over time as they rose and fell. 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 interacted with them co-evolved together. These might seem too strong assumptions, but these are the simplifications that allow simple models to be built and, very often, succeed. Despite the oversimplification, it is known that minimal models that describe the evolution and statistical patterns of both languages and species in ecosystems are able to explain the observed regularities. Perhaps the best example is the so-called neutral theory of ecology (Hubbell, 2001; Solé et al., 2004; Alonso et al., 2006) that 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 it occurs with systems mentioned above, we are capable of reproducing 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 role played by memory, space, hierarchies, social dynamics or demographic parameters, it can be shown that most of these properties have little impact on the statistical patterns of organisation. First, the popularity of PLs only depends on the presence or absence of other PLs. Popular languages are more frequently adopted than rare languages, which are more likely to be forgotten and disappear. And second, we simulate an homogeneous population of software developers that can adopt several languages simultaneously and have a reasonable understanding of PLs features, including design principles and implementation details.

The model consists of a static population of programmers located on a L x L lattice and a fixed pool of programming languages indexed by a set

$$\Sigma = \{0,...,\mu\} \qquad (9)$$

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

$$\vec{s_{ij}} = (s_{ij}^1, s_{ij}^2, ..., s_{ij}^M) \in \Sigma^M \qquad (10)$$

where $s_{ij}^k \in \Sigma$ indicates the index of the k-th language known by the programmer. Every developer can adopt up to M different languages, i.e., $s_{ij}^k \neq s_{ij}^l$ for all $1 \leq k \neq l \leq M$. In addition, we will count how many programmers know the r-th language:

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

where δ(x,y)=1 if x =y and 0, otherwise. At the beginning of the simulation, which corresponds to the year 1958, there are no programming languages 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 the following rules:

Innovation rule: This rule is associated to the "discovery" by a programmer of a previously not used 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 language index r ∈ Σ and 1 ≤ k ≤ M and set $s^k_{ij}(t+1) = r$ with probability ☐ provided that the r-th language does not belong to $\vec{s_{ij}}$, i.e., it is unknown to the programmer located at the 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 neighbours in our 2-D lattice) allows an "infection" to occur. The Moore neighbourhood is defined by $\Omega(i,j) = \{(u,v) : |u-i| \leq 1, |v-j| \leq 1\}$. For each programmer $\vec{s_{ij}}$, we choose one random neighbour site $(u, v) \in$ ☐(i, j) and set $s^q_{ij} = s^r_{uv}$ with probability

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

where the factor Z in the right hand is:

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

i.e, a normalisation constant, $1 \leq q \leq M$ 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 is different in a fundamental way: the user will retain other PLs. Randomly choose $1 \leq r \leq M$ and set $s^{r}_{ij}(t+1) = 0$ with probability

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

where the factor Z in the right hand is a normalisation constant and $\delta > 0$ is the forgetting rule.

Our model assumes that PLs are discovered at a constant, small rate □. The last two rules represent the process of PL adoption and forgetting. Very popular languages, i. e. those such that $N_p(s_{ij}^{r})/Z \approx 1$, tend to be adopted more frequently while rare languages are easier to forget and abandon by the community.

The model consistently reproduces the patterns displayed by PLs, including the ups and downs of abundances that somewhat map into the "popularity" of the language. Common languages have smoother time fluctuations, whereas more rapid fluctuations are associated to rare PLs that often end in extinction. In figure 4A we also display four spatial snapshots of our cellular automaton model corresponding to the previous time series. In the first stages of the simulation, relatively small patches indicate the diffusion of PLs on limited numbers of users. However, as time proceeds, larger patches can be observed resulting from the successful propagation of some PLs and combinations of them. This process is also visualised in figure 4C, where the population size for some of the most abundance PLs is plotted against time. We can see that all these languages start growing but some eventually decline after a very long transient until they become extinct. Other succeed and invade the 2-D lattice.

The model exhibits a marked increase in PL diversity and it also reveals the presence of scaling laws. This is illustrated in figure 5 where we can see how our model predicts the observed frequency-rank distribution. We have used an evolutionary algorithm to obtain the best-fitting parameters $\alpha = 0.495987$, $\eta = 0.905577$ and $\delta = 0.000195$. The maximum language diversity is $\mu = 120$ because of the limitations of the dataset. Other sources (like Wikipedia) report many more languages (Valverde et al., 2015). Still, our experiments show that the behaviour of our model is

robust to a wide range of μ, *M* and *L* values. Here, we have fixed programmer capacity *M*=3 and *L*=32. As observed with the popularity measures used on extant PLs, at a given step we can see that a vast majority of PLs are represented by a small number of users whereas a bunch of them dominates and is adopted by a great majority.

The previous results reveal that the fate of our system, despite the transient richness that matches the previously presented scaling laws, is to reduce the number of PLs as the spread rule dominates over the others. Inevitably, a rather low diversity of PLs will be observed in the future. The three snapshots and frequency 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) we can see after a transient period the emergence of a power-law behaviour of a virtual world sustaining many PLs. Eventually, the same process concludes with a stark impoverishment of our language set, with just a handful of surviving languages.

The model thus makes a strong prediction: in the future, as globalised communication and increasing returns accelerate, few programming languages will survive. In this context, many examples exist of once successful and widespread languages that become extinct. We are constantly facing this pattern, that is also common to many spoken languages. A recent case is provided by Perl, a programming language invented in 1987, that became the essential web writing language during the early days of the Internet. Despite its great impact, Perl is fading away towards extinction. As pointed out by Doug Barry: *"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"*.

It is actually interesting to see that the comparison is made with spoken languages. 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.

V. DISCUSSION

People's lives are increasingly dependent on the correct working of software. This dependency is increasing because, as wireless networks proliferate, many portable devices become interconnected. The increasing popularity of the personal computer has been followed by a consistent trend towards more complex software systems. In that respect, the expansion of PLs

over decades of change share some traits with human languages and differ in some important aspects. Different languages coexist, come and go but they are all characterised the essentially same complexity provided by syntax. The diffusion of languages correlates directly with the number of speakers, but they are all essentially equal in structure. PLs instead can be classified in two major categories, i.e., procedural and declarative, and because of their distinct goals and underlying hardware to be executed, they can display different levels of complexity (Scott, 2009).

Programming languages are one of the main drivers of successful software development. They are both a means of telling computers how to do useful work as well as the means of exchanging ideas and algorithms between people. The evolution of programming languages is linked both to the software environment (hardware) and socio-economic 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 to 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 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 globalised world and in particular the ecosystem formed by PL users is finite and spreading of innovations is very fast, candidate languages rapidly experience a strong selection process. Users will adopt a PL provided that is is not too difficult to learn, that is shared by other users (the more, the better) and that is compatible with existing hardware requirements. Since increasing returns play here a very important role and that homogenisation is growing with globalisation, PLs are doomed to be less and less diverse. As opposed to a spoken language, where several factors, such a special status, might favour their persistence despite competitive forces (Solé et al., 2010) PLs are not so crucial for the programmer as a kind of strong social component. Such lack of special status and a rapidly changing hardware might contribute to this decline and accelerate it.

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 only depends on language popularity. However, expert developers can learn new languages faster than novices. Computer programming, like reading, involves a number of different, interrelated skills. Some languages have been designed teaching (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 et al., 1984) affect the dynamics of PL adoption. In this context, extensions of modelling approaches like the ones discussed here will help us to explore these questions and more generally, towards understanding the societal impact of information technologies.

ACKNOWLEDGMENTS

We thank Mark Watney and the members of the CSL for useful discussions. This work has been supported by Fundación Botin, the Spanish Ministry of Economy and Competitiveness, Grant FIS2013-44674-P and FEDER (SV) and by the Santa Fe institute.

REFERENCES

1. Solé, R.V., Valverde, S., Rosas-Casals, M., Kauffman, S. A., Farmer, J. D. & Eldredge, N (2013) The evolutionary ecology of technological innovation. Complexity 18, 15-27.

2. Arthur, B. (1994) Increasing Returns and Path Dependence in the Economy. Ann Arbor, MI: Michigan University Press.

3. Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., The Google Books Team, Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak M. A., Aiden, E. L. (2011) Quantitative Analysis of Culture Using Millions of Digitized Books. Science 331 (6014), 176-182.

4. Solé, R.V. (2011) Phase Transitions. Princeton University Press.

5. Burks, A. W., and Burks, A.R. (1981) The ENIAC: First General-Purpose Electronic Computer. Annals of the History of Computing (3) 4.

6. von Neumann, J. (1993) First Draft of a Report on the EDVAC, Reprinted in: IEEE Annals of the History of Computing 15 (4), 27-75.

7. Martinez-Mekler, G., Martínez, R. A., del Ro, M. B., Mansilla R., Miramontes, P., and Cocho, G. (2009) Universality of Rank-Ordering Distributions in the Arts and Sciences PLoS ONE 4(3): e4791.

8. Solé, R. V., and Goodwin, B. (2001) Signs of life: how complexity pervades biology. New York, NY: Basic Books.

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

10. Solé, R. V., Corominas-Murtra, B., Fortuny, J. (2010) Diversity, Competition, Extinc- tion: the Econophysics of Language Change. J. R. Soc. Interface 7, 1647-1664.

11. Hubbell, S.P. (2001) The Unified Neutral Theory of Biodiversity and Biogeography. Princeton University Press, Princeton, NJ.

12. Solé, R. V., Alonso, D., and Saldaa, J. (2004) Habitat Fragmentation and Biodiversity Collapse in Neutral Communities. Ecological Complexity 1, 65-75.

13. Alonso, D., Etienne,R. S., and McKane, A. J. (2006) The merits of neutral theory Trends in ecology evolution 21 (8), 451-457.

14. Bentley, R. A., Hahn, M. W., and Shennan, S. J. (2004) Random Drift and Culture Change. Proc. Biol. Sci. 271(1547), 1443-1450.

15. Axelrod, R. (1997) The Dissemination of Culture: A Model with Local Convergence and Global Polarization, J. Conflict Resolution 41: 203-226.

16. Valverde, S., and Solé, R. V. (2015) Punctuated Equilibrium in the Large-Scale Evo- lution of Programming Languages. J. Roy. Soc. Interface 12: 20150249.

17. Scott, M. L., (2009) Programming Language Pragmatics (3rd Edition). Morgan Kauf- mann Publishers.

18. Pea, R. D., and Kurland, D. M. (1984) On the Cognitive Effects of Learning Computer Programming, New Ideas Psychol. 2(2), 137-168.

FIG. 1. Increasing returns and symmetry breaking in technological evolution. In (A) we display the time series of citations for "VHS recorder" and "betamax", the two competing videotape designs that emerged almost simultaneously and became the dominant options in the market. Despite their close coexistence (as indicated in the inset figure B) VHS eventually dominated the market. This phenomenon can be understood using a symmetry breaking model, with two alternative solutions (VHS wins and Beta looses, or VHS looses and Beta wins in an alternative history). In (C) we display the potential associated to a simple, symmetric model of technological competition (see text). One unstable state (open circle) and two symmetric, stable states (filled circles) are accessible. Initial conditions and accidents can play a crucial role.

FIG. 2. Cultural diffusion of IT technologies. (A) Time series of n-gram abundance associated to two relevant n-grams. Here we used "programming languages" and "computer hardware" which correspond to the upper (continuous) and lower (discontinuous) curves, respectively. The curves display qualitative similar behaviour and possibly mirroring the co-evolution between hardware and software (see text). A sudden increase of the abundance of "programming languages" takes place around 1980 (inset). The bump (*) signals the year 2000 (Y2K) problem, which affected many programming languages. In the second part (shadowed region), there is a decay of "programming languages" popularity with (B) the widespread adoption of IT technologies. The plot shows the sustained growth of the number of US personal computers (x100) since 1980 (data source: World Data Bank).

FIG. 3. Log-log plot of the frequency-rank, ordered by descending popularity, which which the programming languages appear in the *Transparent Language Popularity Index* (2013). This distribution follows the discrete generalised beta distribution (DGBD) with parameters (a, b)= (1.44, 0.46) (see text). The arrows point to the rank position of popular PLs: there is a huge diversity of languages of very different nature and antiquity.

FIG. 4. (A) Four snapshots of the cellular automaton model of programming language spread on a 2-D lattice. Here each site represents a programmer, who can use/store up to M=3 PLs. Colours encode the population diversity of PLs. (B) Time series of the number of different languages present in the population grows very quickly at the beginning t=2, where most programmers only know one (dark red) or zero (black sites) languages. High diversity of PLs plates until $t \approx 50$, which is represented by different shades of red and yellow sites. With time, M=3 dominant languages

emerge and a significant loss of diversity follows. According to this model we are now located in the decay phase (pointed by the arrow). (C) The inset shows the times series of number of language adopters for some of the individual languages, including in particular those that finally get established at the end of the diffusion process. Individual behaviour in our model is typical of competition dynamics in other technological systems (see figure 1 and 2).

FIG. 5. From left to right, time sequence of PL diffusion using our model. At the beginning $t = 4$ we have language coexistence without constraints, which corresponds to an almost flat rank-frequency distribution. Dark colours indicate that programmers have free memory slots. At $t \approx 60$, we recover the observed frequency-rank distribution because different languages are competing for programmer infection. This corresponds to an heterogenous spatial structure. At longer timescales ($t = 250$) the system stabilises in an homogeneous configuration with a few M=3 dominant languages (many sites display the same colour). Indeed, limited memory slots and feedback effects have destroyed the initial language diversity. Parameters: $\alpha = 0.495987$, $\eta = 0.905577$, $\delta = 0.000195$, $\mu = 120$, $M = 3$, and $L = 32$. We have assumed that significant communities have at least 20 members.