Punctuated equilibrium in software evolution

A. A. Gorshenev[1] and Yu. M. Pis'mak[1,2]
[1]*Department of Theoretical Physics, State University of Saint Petersburg, Saint Petersburg 198504, Russia*
[2]*Institute for Theoretical Physics, University of Heidelberg, Heidelberg D-69120, Germany*
(Received 11 July 2003; published 23 December 2004)

An approach based on the paradigm of self-organized criticality is proposed for experimental investigation and theoretical modeling of software evolution. The dynamics of modifications is studied for three free, open source programs MOZILLA, FREE-BSD, and EMACS using the data from version control systems. Scaling laws typical for self-organized criticality found. A model of software evolution presenting the natural selection principle is proposed. Results of numerical and analytical investigation of the model are presented. They are in good agreement with data collected for real-world software.

DOI: 10.1103/PhysRevE.70.067103 PACS number(s): 89.75.Fb, 87.23.Kg, 05.65.+b

Basic self-organization mechanisms of complex systems in nature have been intensively studied in recent years. The paradigm of self-organized criticality (SOC) proposed in the pioneering paper of Bak, Tang, and Wiesenfeld [1] appears to be most fruitful here. SOC dynamics is characterized by avalanchelike changes of the system state with power law statistics of the avalanche growth. The main feature of the SOC regime is that it is an attractor of the system dynamics approached without any fine-tuning of control parameters.

Studies of fossil records have shown that biological evolution is a strong nonequilibrium process with long periods of stasis interrupted by avalanches of large changes in the biosphere. This is a main point of the punctuated equilibrium concept of biological evolution suggested by Gould and Eldredge [2,3]. A quantitative analysis of paleontological dates revealed scaling manifested in power laws of avalanche distributions in the extinction of species [4,5]. Therefore biological evolution can be considered as a kind of SOC dynamics. This has been demonstrated by Bak and Sneppen in the proposed model of Darwinian selection in ecosystems [6]. The development of computer science and engineering created a "virtual biosphere" with specific evolution laws of "virtual species"—computer programs. In this paper we propose an approach to studying software evolution in the framework of the SOC concept.

The "life" of a large computer program is a natural evolutionary process. During its creation the program often undergoes multiple internal reorganizations. New devices and platforms are supported, new features added, system tuning performed, erroneous code corrected, and a large number of cosmetic changes go on during the development of any program [7,8]. Despite the fact that the first papers on software evolution study are now decades old, universal mechanisms of computer program evolution are unclear. Most of the existing research methods in this area are based on the assumption that estimates of possible changes in a program can be obtained without taking into account the underlying dynamical laws creating this system [9,10]. In a multitude of papers authors propose statistical methods predicting the number of defects in a program using some kind of metrics describing complexity, size, volume, etc. [11,12].

From our point of view the main disadvantage of such an approach is that even the best static metric forecasting the number of improvements to be made in a computer program corresponding to a given specification becomes useless if the specification changes in time. Our approach can be considered as an elaboration of a prototype for dynamical metrics based on the use of characteristics of the SOC universality class of the system.

There are many phenomenological works about software evolution. Lehman's laws suggest that as a system grows in size, it becomes increasingly difficult to add new code unless explicit steps are taken to reorganize the overall design [13,14]. Some systems have been examined both at the system level and within top-level subsystems. It has been noted that subsystems can behave quite differently from the system as a whole [9,10]. Good metaphors such as "code decay" have been proposed to describe the continuous process that makes software more brittle over time [8,15]. Thus software evolution has many similar features to the evolution of biological species, and one can expect that evolution of large computer programs has a physical background presented by some class of universality of SOC dynamics.

To study software evolution processes it is necessary to have information about the state of the system in different moments of time. The usual sources of such data are various versions or releases of a product [9,10,13]. Unfortunately, the number of releases rarely exceeds a couple of tens. This fact significantly decreases the possibility of studying the evolution of programs. Better sources of information about changes in computer programs are version control systems. One of them is the Concurrent Versions System (CVS). It keeps information about changes happening in short time intervals.

Using the CVS in our work, we studied the histories of three software projects: the MOZILLA web browser, the FREE-BSD operating system, and the Gnu EMACS text editor [16–18]. For each of these projects we analyzed only files written in the basic project language. These are C++ for MOZILLA, C for FREE-BSD, and LISP for EMACS. Header files for C and C++ were not studied. The total amounts of processed files are approximately 9000, 11 000, and 900 for MOZILLA, FREE-BSD, and EMACS. The total lengths of Revision Control System files are 1×10^7, 1×10^7, and 2×10^6 lines. The total volume of all three repositories (including not only files of basic languages, but all files) exceeds 2 gi-

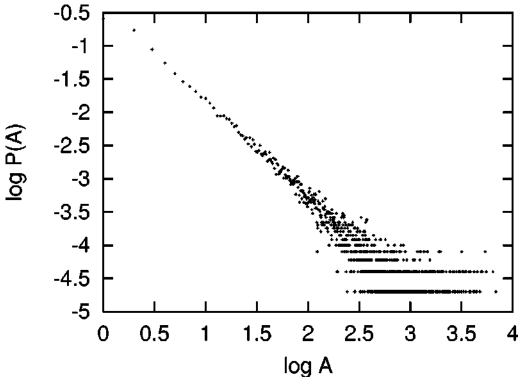

FIG. 1. Distribution $P(A)$ for FREE-BSD.

gabytes. Due to some resource limitations only part of the FREE-BSD CVS storage was processed. The histories of all three projects are stored under control of the CVS and were publicly available during our research period from the corresponding internet servers [16–18].

For each change of each file an amount D of deleted lines and an amount A of added lines were collected. Empty lines and comments were collected together with the rest of the data. The number of lines in the very first version of each file was not counted. Distributions $P(A)$ and $P(D)$ were evaluated for these two arrays A_i and D_i. As an example the data for FREE-BSD are shown in Figs. 1 and 2 on a common log-log scale ($\log x = \log_{10} x$). Results for EMACS and MOZILLA are similar. One can see that power functions are accurate approximations for $P(A)$ and $P(D): P(A) \sim A^{\mu_a}$, $P(D) \sim D^{\mu_d}$. The values of the exponents are the following: FREE-BSD, $\mu_a = -1.44 \pm 0.02$, $\mu_d = -1.48 \pm 0.02$; MOZILLA, $\mu_a = -1.43 \pm 0.02$, $\mu_d = -1.47 \pm 0.02$; EMACS, $\mu_a = -1.39 \pm 0.03$, $\mu_d = -1.49 \pm 0.04$. These scaling laws can be considered as a manifestation of SOC in the evolution of software.

One important notion being used in the description of SOC dynamics is the avalanche. The SOC process can be presented as a consequence of metastable states interrupted by avalanchelike changes in the system. For evolution of computer programs a close analog of the avalanche is the set of changes from version to version. We see that the avalanche statistic in the evolution of software is described by power functions with nontrivial exponents. The universality of SOC dynamical mechanisms allows one to hope that a simple "holistic" model can be constructed for its quantitative description [19]. To realize this idea for software evolution modeling we use the following assumptions. The peculiarity of software changes is that a programmer cannot modify a program at different points simultaneously (at least using a traditional development tool). The point of the changes is characterized as the "weakest" one in the program text: a programmer has some subjective estimation of parts of a program and makes changes in a place that is estimated as extremely nonsatisfactory. If a change is made at some point, corresponding changes must be made in some other places, i.e., in the program there is a coordination structure of its elements. We suppose that changes in the program cannot make its size less than some minimal one.

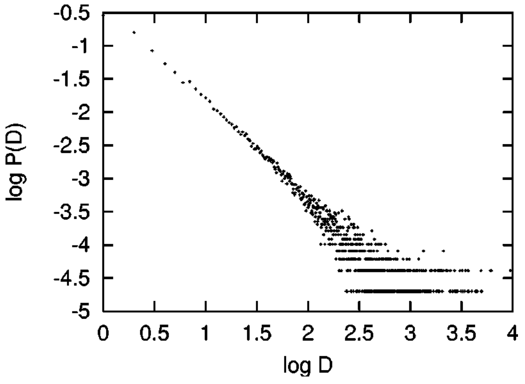

FIG. 2. Distribution $P(D)$ for FREE-BSD.

We formulate the model presenting this concept as follows. The computer program as a system constitutes a sequence of elements—lines of code. At time point t the ith line is characterized by a number $b_i(t)$, $0 < b_i(t) < 1$, representing its "fitness" in the program text or a barrier in respect to change in future stages of evolution. The state of the system of N elements is fully given by the set of barriers $B(t) = \{b_i(t), i = 1, 2, \ldots, N\}$. The evolution of the program is described in our model as the sequence of $B(t)$ for discrete time points $t = 0, 1, 2, \ldots$. The coordination structure of the program is represented by a network of its elements, where each element node conforms with its nearneighbors. The node having minimal barrier is defined as the weakest unit of the system. At each time point t we define the set $W(t)$ containing the weakest unit with all its neighbors. We call $W(t)$ the weakest component at time t.

The dynamics in the model is defined in the following way. The initial number of nodes $N(0)$ and the minimal possible number of nodes K are supposed to be given. The initial values of barriers $b_i(0)$ are chosen at random. The state $B(t)$ at time point t transforms into state $B(t+1)$ as follows. If the number of nodes $N(t)$ in the system is more than K two kinds of changes are possible. With probability α, the weakest unit is deleted from the system or with probability $1-\alpha$ a new neighbor node to the weakest unit is inserted into the system. After that the barriers of all nodes from the weakest component $W(t)$ are set randomly. So if $N(t) > K$, the size of the system decreases or increases by 1 for one time step. If $N(t) = K$, then deletion is impossible and the above described insertion is made.

Our model is a modification of the well known simple model of biological evolution suggested by Bak and Sneppen [6], and its essential specific is that the number of system elements varies in time. In our study we have considered two versions of the model: with one-dimensional (1D) and random neighbor (RN) coordination structure. In the 1D case the nodes are organized into a 1D lattice with periodic boundary conditions, and each node has two neighbors. In the RN model, there is no fixed coordination structure in the system, and at each time step k random nodes are chosen as neighbors of the weakest unit. We have considered the case $k=1$ only.

An avalanche as the elementary process of complex behavior of a nonequilibrium dynamical system can be defined in different ways. Usually in the model of SOC dynamics, λ and transient avalanches are considered [6,20,21]. In studies of our model we were interested mostly in transient avalanches. They can be defined as follows. At the time moment t_0 the minimal barrier has the value f_0. The sequence of S time steps during which the minimal barrier does not exceed f_0, $b_{\min}(t) < f_0$, $t_0 < t < t_0 + S$, is called a transient avalanche or just an avalanche if it finishes at the time point $t_0 + S$ when the value of the minimal barrier becomes larger than f_0, $b_{\min}(t+S) > f_0$. The distribution $P(S)$ of the avalanche temporal duration and the distribution $P(R)$ of the avalanche spatial volume are important characteristics of the type of dynamics. For our model it is reasonable to consider two values as characteristics of the volume of changes produced in the system by the avalanche. One of them is the number A of new elements appearing in the system at the end of the avalanche. The other is the number D of elements disappearing from the system at the end of the avalanche. In the dynamics of our model we studied mostly the distributions $P(S)$, $P(A)$, and $P(D)$ of the temporal and spatial characteristics of avalanches.

We studied numerically the 1D and RN versions of the model for $\alpha = \frac{1}{2}$. The initial size of the system was 8000 elements. The experiment went on until one million avalanches were registered. We got the following results. The $P(S)$, $P(A)$, and $P(D)$ distributions can be sufficiently approximated by the power functions $P(S) \sim S^{\tau}$, $P(A) \sim A^{\mu_a}$, and $P(D) \sim D^{\mu_d}$ with exponents $\tau = -1.358 \pm 0.005$, $\mu_a - 1.45 \pm 0.01$, and $\mu_d = -1.47 \pm 0.02$ for the 1D model and $\tau = -1.901 \pm 0.008$, $\mu_a - 1.98 \pm 0.01$, and $\mu_d = -2.10 \pm 0.02$ for the RN model.

For the RN model it is possible to obtain an analytical description in the framework of the master equation formalism. To do it one can use the method of construction of the master equation proposed for analysis of SOC dynamics of the random neighbor version of the Bak-Sneppen model [22]. If we denote by $P_{n,N}(t)$ the probability that at time point t there are N nodes in the system, and n have barriers less than λ, where $0 < \lambda < 1$, then the dynamical rules of the RN model result in the following master equation:

$$P_{n,N}(t+1) = (\alpha + \beta\delta_{N,K+1})P^a_{n,N}(t) + \beta P^d_{n,N}(t), \qquad (1)$$

where $\beta = 1 - \alpha$, and in terms of $\mu = 1 - \lambda$, $\rho_{n,N} = (n-1)/(N-1)$, and $\sigma_{n,N} = 1 - \rho_{n,N}$ the quantities $P^d_{n,N}(t)$ and $P^a_{n,N}(t)$ can be presented by the following relations:

$$\begin{aligned}P^a_{n,N}(t) &= A^a_{n+2,N-1}P_{n+2,N-1}(t) + B^a_{n+1,N-1}P_{n+1,N-1}(t)\\ &\quad + C^a_{n,N-1}P_{n,N-1}(t) + D^a_{n-1,N-1}P_{n-1,N-1}(t)\\ &\quad + E^a_{n-2,N-1}P_{n-2,N-1}(t) + (\mu^3\delta_{n,0} + 3\lambda\mu^2\delta_{n,1}\\ &\quad + 3\lambda^2\mu\delta_{n,2} + \lambda^3\delta_{n,3})P_{0,N-1}(t),\end{aligned}$$

$$\begin{aligned}P^d_{n,N}(t) &= A^d_{n+2,N+1}P_{n+2,N+1}(t) + B^d_{n+1,N+1}P_{n+1,N+1}(t)\\ &\quad + C^d_{n,N+1}P_{n,N+1}(t) + (\mu\delta_{n,0} + \lambda\delta_{n,1})P_{0,N+1}(t),\end{aligned}$$

$$A^a_{n,N} = \mu^3\rho_{n,N}, \quad B^a_{n,N} = 3\lambda\mu^2\rho_{n,N} + \mu^3\sigma_{n,N},$$

$$C^a_{n,N} = 3\lambda\mu^2\sigma_{n,N} + 3\lambda^2\mu\rho_{n,N}, \quad D^a_{n,N} = 3\lambda^2\mu\sigma_{n,N} + \lambda^3\rho_{n,N},$$

$$E^a_{n,N} = \lambda^3\sigma_{n,N}, \quad A^d_{n,N} = \mu\rho_{n,N}, \quad B^d_{n,N} = \mu\sigma_{n,N} + \lambda\rho_{n,N},$$

$$C^d_{n,N} = \lambda\sigma_{n,N}.$$

Here, the coefficients

$$A^a_{n,N}, B^a_{n,N}, C^a_{n,N}, D^a_{n,N}, E^a_{n,N}, A^d_{n,N}, B^d_{n,N},$$

and $E^d_{n,N}$ are defined for $0 < n \leqslant N$. For $n \leqslant 0$ and $n > N$ they are assumed to be zero. The master equation (1) enables one to find $P_{n,N}(t)$ for $t > 0$, if the initial values $P_{n,N}(0)$ are given. Based on this equation one can obtain analytical results for the characteristics of the dynamics in the RN model. With that end in view it is convenient to use the formalism of the generating function that appeared to be very effective for construction of the exact solution for the master equations of the RN version of the Bak-Sneppen model [23–26]. The dynamics of the RN model is more complex than that of the Bak-Sneppen model, and solution of the master equation for the RN model for software evolution appears to be not an easy problem. Here, we present only the exact result for $P_N(t) = \Sigma_{n=0}^{N} P_{n,N}(t)$, which is the probability that the system has N elements with barriers less than λ at time point t. Let us denote as $\mathcal{N}(y,u)$ the generating function for probabilities $P_N(t) : \mathcal{N}(y,u) = \Sigma_{N=K,t=0}^{\infty} P_N(t) y^{N-K} u^t$. From Eq. (1) we obtain the following equation for $\mathcal{N}(y,u)$:

$$\mathcal{N}(y,u)[y - u(\alpha y^2 + \beta)] = y\mathcal{N}(y,0) + u\beta[y^2 - 1]\mathcal{N}(0,u).$$

It describes one-dimensional discrete diffusion with reflection and can be solved by the methods used in [23–26]. The result has the form

$$\mathcal{N}(y,u) = \frac{y\mathcal{N}(y,0)(u-\tau) + u\alpha\tau\mathcal{N}(\tau,0)(y^2-1)}{(u-\tau)[y - u(\alpha y^2 + \beta)]}$$

where $\tau = (1 - \sqrt{1 - 4u^2\alpha\beta})/2u\alpha$ is the analytical solution of the equation $[\tau - u(\alpha\tau^2 + \beta)] = 0$ at the point $u = 0$. The mean value $n(t) = K + \Sigma_N P_N(t) N$ of the system element number at the time point t has the following asymptotic for large $t : n(t) \approx (2\alpha - 1)t$ for $\alpha > 1/2$, $n(t) \approx \sqrt{2t/\pi}$ for $\alpha = 1/2$, and if we denote by $p_{ev}(p_{od})$ the probability that the initial number $N(0)$ of nodes is even (odd), then

$$n(t) \approx K + [1 + (-1)^{t+K}(1 - 2\alpha)^2(p_{ev} - p_{od})]/[2(1 - 2\alpha)]$$

for $\alpha < 1/2$. The corrections to the leading terms of the asymptotic are of the form $n(0) - 2\alpha\beta/(\alpha^2 - \beta^2) + f(t)g_1(t)$ for $\alpha > 1/2$, $t^{-1/2}g_3(t)$ for $\alpha = 1/2$, and $f(t)g_2(t)$ for $\alpha < 1/2$. Here $f(t) = [4\alpha\beta]^{t/2}/t^{-3/2}$ and $g_i(t)$, $i = 1, 2, 3$, are bounded for large t, i.e., there are constants T, M such that $|g_i(t)| < M$ if $t > T$. Since $4\alpha\beta < 1$ for $\alpha \neq 1/2$, the function $f(t)$ decreases exponentially fast for large t.

The asymptotic behavior of $n(t)$ demonstrates the dynamical phase transition at the point $\alpha = 1/2$. For $\alpha < 1/2$ the volume of the system remains finite, but for $\alpha \geqslant 1/2$ it can became as large as one likes. At the point $\alpha = 1/2$, the dynamics of the system is critical.

In the above formulated model we tried to present elementary mechanisms of software changes. They are made by a programmer locally in the place where these changes are most needed. But a program changed in one place often must be changed in other places in some way connected to the first one. For example, in order to change the number of arguments of a subroutine call, one needs to change not only the line containing the call operator but the definition of the subroutine also. This would lead to some subsequent changes of all the calls to the subroutine in the whole program. If one adds a line in which some data are read from a disk one should add some lines to check whether the data have been read successfully, and this in turn can require some change in the list of modules included, which in turn can cause a name conflict, which in turn can cause other changes, etc. Thus, avalanchelike processes seem to be natural for modifications of programs. The avalanche ends when all the parts of the program code more or less satisfy some subjective and implicit criterion of the programmer. Naively speaking, the program as a whole becomes "a little bit better." In the model it can be presented as a process terminating when the value of the minimal barrier becomes greater than the initial one. This was the reason we studied transient avalanches of the self-organization period and not the λ avalanches of the stationary mode. The obtained statistical characteristics of avalanches make it possible to conclude that SOC is the dominating dynamical regime in the evolution of free software. Our results demonstrate that natural selection can create this type of "punctuated equilibrium" of such complex "virtual beings" in the information sphere. We believe that in the framework of the proposed approach modern methods of investigation of SOC dynamics can be very effective for studies of physical aspects of software evolution. Our results could be seen also as a theoretical prerequisite for the development of new tools and methods for advanced measures of software engineering quality.

The work of Yu.M.P. was supported in part by the Russian Foundation for Basic Research (Grant No. 03-01-00837), the Swiss National Science Foundation (SCOPES Grant No. 7SUPJ062295), and a Nordic Grant for Network Cooperation with the Baltic Countries and Northwest Russia (No. FIN-6/2002).

[1] P. Bak, C. Tang, and K. Wiesenfeld, Phys. Rev. A **38**, 364 (1988).

[2] S. Gould and N. Eldredge, Paleobiology **3**, 115 (1977).

[3] S. Gould and N. Eldredge, Nature (London) **332**, 211 (1988).

[4] D. M. Raup and I. J. Sepkoski, Jr., Science **215**, 1501 (1982).

[5] D. M. Raup and G. E. Boyajian, Paleobiology **14**, 109 (1988).

[6] P. Bak and K. Sneppen, Phys. Rev. Lett. **71**, 4083 (1993).

[7] N. E. Fenton and M. Neil, IEEE Trans. Software Eng. **25**, 675 (1999).

[8] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, IEEE Trans. Software Eng. **27**, 1 (2001).

[9] M. W. Godfrey and Q. Tu, in Proceedings of the International Conference on Software Maintenance, 2000 (unpublished), pp. 131–142.

[10] H. Gall, M. Jazayeri, R. R. Klösch, and G. Trausmuth, in Proceedings of the International Conference on Software Maintenance, 1997 (unpublished), pp. 160-166.

[11] F. Akiyama, Inf. Process. 71, 353 (1971).

[12] M. H. Halstead, *Elements of Software Science* (Elsevier, New York, 1977).

[13] M. M. Lehman, D. E. Perry, J. F. Ramil, W. M. Turski, and P. D. Wernick, in *Proceedings of the Fourth International Symposium on Software Metrics, 1997* (IEEE Computer Society, Washington, D.C., 1997), pp. 20–23.

[14] M. M. Lehman and L. A. Belady, *Program Evolution: Process of Software Change* (Academic Press, New York, 1985).

[15] D. L. Parnas, In *Proceedings of the 16th International Conference on Software Engineering, 1994* (IEEE Computer Society Press, Los Alamitos, 1997), pp. 279–287.

[16] www.freebsd.org

[17] www.mozilla.org

[18] www.gnu.org

[19] P. Bak, *How Nature Works: The Science of Self-Organized Criticality* (Copernicus Press, New York, 1996).

[20] M. Paczuski, S. Maslov, and P. Bak, Phys. Rev. E **53**, 414 (1996).

[21] S. Maslov, Phys. Rev. Lett. **77**, 1182 (1996).

[22] H. Flyvbjerg, K. Sneppen, and P. Bak, Phys. Rev. Lett. **71**, 4087 (1993).

[23] Yu. M. Pis'mak, J. Phys. A **28**, 3109 (1995).

[24] Yu. M. Pis'mak, Phys. Rev. E **56**, R1326 (1997).

[25] G. L. Labzovski and Yu. M. Pis'mak, Phys. Lett. A **246**, 377 (1998).

[26] Yu. M. Pis'mak, Phys. Rev. Lett. **83**, 4892 (1999).