Murphy's law, the fitness of evolving species, and the limits of software reliability

Robert M. Brady (1), Ross J. Anderson (2), and Robin C. Ball (3)
1: Brady plc, Cambridge; r.brady@bradyplc.co.uk
2: Computer Laboratory, Cambridge; rja14@cl.cam.ac.uk
3: Cavendish Laboratory, Cambridge; rcb1@phy.cam.ac.uk

Abstract

We tackle two problems of interest to the software assurance community. Firstly, existing models of software development (such as the waterfall and spiral models) are oriented towards one-off software development projects, while the growth of mass market computing has led to a world in which most software consists of packages which follow an evolutionary development model. This leads us to ask whether anything interesting and useful may be said about evolutionary development. We answer in the affirmative. Secondly, existing reliability growth models emphasise the Poisson distribution of individual software bugs, while the empirically observed reliability growth for large systems is asymptotically slower than this. We provide a rigorous explanation of this phenomenon. Our reliability growth model is inspired by statistical thermodynamics, but also applies to biological evolution. It is in close agreement with experimental measurements of the fitness of an evolving species and the reliability of commercial software products. However, it shows that there are significant differences between the evolution of software and the evolution of species. In particular, we establish maximisation properties corresponding to Murphy's law which work to the advantage of a biological species, but to the detriment of software reliability.

1 Introduction

The traditional models of software development assume a project approach, in which a large system is developed from scratch. This development may be top down and driven by successive refinements of a specification (the waterfall model) or, where the requirements are unclear, may involve a process of iterative prototyping (the spiral model). But, since these models were developed, the world has changed; nowadays most software consists of packages. Some of these are general retail products, such as word processors, spreadsheets and small business accounts, while others are specialised. For example, the first author's company sells a package used by investment banks to track the exposure of a portfolio of investments to changes in parameters such as interest and exchange rates.

But whether mass market or specialised, software packages share an evolutionary model of development. Products are developed by modifying previous versions, and over the years, they become so complex that they could simply not be developed (or redeveloped) from scratch. Indeed, Microsoft has tried more than once to rewrite Word, and has given up each time [1].

Although there are useful books written by practitioners of evolutionary software development (e.g., [2]), there is little on the topic in the research literature, and this led us to ask whether we can say anything that is not just empirically useful but also scientifically interesting. In particular, can we develop a reliability growth model that accurately reflects the experience of companies that develop large software packages?

This led us to confront one of the outstanding puzzles in reliability growth theory. The behaviour of systems which contain a single bug, or a small number of them, is known to be governed by Poisson survival statistics [3]. For example, the probability p_i that a particular defect remains undetected after t statistically random tests is given by $p_i = e^{-E_i t}$. The quantity E_i is sometimes called the 'virility' of the defect, and depends on the proportion of the input space that it affects. Many systems that adapt to events in their environment can be described quantitatively in a similar way, and this gives us some hope that a useful analogy might be developed between software engineering and biological evolution.

The problem is that extensive empirical investigations have shown that in a large and complex system, the likelihood that the t-th test fails is not proportional to e^{-Et} but to k/t for some constant k. This was first measured by Adams, who reviewed the bug history of IBM mainframe operating systems [4]. An equivalent formulation is that in order for software to have a mean time to failure of (say) 10,000 hours, it must be tested for at least that much time (see, e.g., Butler and Finelli [5]).

In the rest of this paper, we present a reliability growth model which explains this apparent discrepancy. We prove that, given reasonable assumptions, the $e^{-E_i t}$ statistics of the individual bugs sum to k/t for the whole system over a wide range of values of t. Our initial insight was that if we define a 'temperature' $T = 1/t$, then the Poisson statistic becomes $p_i = e^{E_i/T}$. This is now a thermal (Boltzmann) distribution, and it suggests that we might be able to use methods inspired by statistical thermodynamics. As we will see below, some of the mathematics goes across, but the details are different (there is a different normalisation).

This analysis gives a number of interesting results. In addition to explaining the observed reliability growth of k/t, we show that under assumptions that are often reasonable, it is the best possible; that there are fundamental limits on the reliability gains to be had from re-usable software components such as objects or libraries; that the failure time measured by a tester depends only on the initial quality of the program, the scope of the testing, and the number of tests; that each bug's contribution to the overall failure rate is independent of whether the code containing it is executed frequently or rarely (intuitively, code that is executed less is also tested less); and that it is often more economic for different testers to work on a program in parallel rather than in series.

All this leads us to ask to what extent can useful analogies be drawn with biological evolution. To answer this, we develop the basics of population genetics using our techniques. This development is limited, as we cannot take account of beneficial mutations; nonetheless, we can demonstrate useful parallels. For example, it is known that species evolve quickly under a new environmental pressure to the extent that they have genic variance: a faster fox will catch the slower rabbits first, leading to a rapid increase in the rabbit population's average speed. Analogously, much of the improvement of large software systems under testing comes from the fact that some bugs are much more virile than others, and when they are found and removed, the overall quality of the software increases quickly.

However, our model also gives some caveats against stretching the biological analogy too far. We prove a version of 'Murphy's Law': that the number of defects which survive a selection process is maximised. This applies equally to software and to species; software testing removes the minimum possible number of bugs, consistent with the tests applied, while biological evolution enables a species to adapt to a changed environment at a minimum cost in early deaths. However, while this is an advantage to a biological species — it preserves the maximum amount of genetic variability — it is a drawback for the software writer, as it leaves intact the largest possible number of latent bugs, which may be triggered later by a change in the way that the system is used.

In the following section, we solve macroscopically for the entropy, the overall system reliability, and the distribution of defects; the nub of the argument, which will be familiar to students of thermodynamics, is that terms contributing less than $O(1/t)$ can be ignored in the region of interest. The reader who is unfamiliar with thermodynamics might skim this section on a first reading, noting merely the notation and the fact that the model's predictions are in good agreement with measurement. In section 3, we relate our results to previously speculated or apocryphally known properties in practical software engineering. Section 4 gives a microscopic definition of the free energy and entropy, and shows from their properties that the number of defects is a maximum (Murphy's law). In sections 5 and 6, we apply the same thermodynamic method to the defects in the DNA of a living organism. This enables us to develop the analogies that are possible with biology, and show their limitations. In the conclusion, we suggest other physical systems to which our techniques might be applied.

2 The reliability of software

Suppose that we have a piece of software that is large enough for statistical assumptions to hold; that t random tests are used in the de-bugging phase; and that each test failure is traced to the bug or bugs causing it, which are corrected. Let there be $N(t)$ bugs left after t tests (for simplicity, we assume one test per unit time), and let the probability that a test fails be $E(t)$, where a test failure counts double if it is caused by two separate bugs. Then the rate at which defects are eliminated is equal to the rate at which they are discovered. That is:

$$dN = -Edt \tag{1}$$

The thermodynamic entropy is given by $S = \int dE/T$. Thus $S = \int tdE = Et - \int Edt$. This can be solved by substituting equation 1 and neglecting the constant of integration, giving $S = N + Et$.

The entropy S is a decreasing function of t (since $dS/dt = tdE/dt$ and $dE/dt < 0$). Thus both S and N are bounded by their initial value N_0 (the number of bugs initially present) and the quantity $S - N = Et$ is bounded by a constant k (with $k < N_0$), that is:

$$E \leq k/t \tag{2}$$

The quantity $S - N = Et$ vanishes at $t = 0$ and $t = W_0$, where W_0 is the thermodynamically large number of input states the program can process. It has a maximum value $Et = k$. We now wish to show that this is maximum is attained over a wide range of values of t in a large system, and indeed that $Et \approx k$ for $N_0 \ll t \ll W_0$. This will be the region of interest in most real world systems; although the size W_0 of the input state space will be far too large to test, we will certainly do more tests than the initial number of bugs N_0, as this will hopefully be bounded by the number of lines of code.

We will write equation (2) as $Et = k - g(t)$ where $0 \leq g(t) \leq k$. Since $g(t)$ is bounded, we cannot have $g(t) \sim t^x$ for $x > 0$. On the other hand, if $g(t) = At^{-1}$, then this makes a contribution to N of $-\int g(t)dt/t = A/t$, which is reduced to only one bug after A tests, and this can be ignored as $A < k$. Indeed, we can ignore $g(t) = At^{-x}$ unless x is very small. Finally, if $g(t)$ varies extremely slowly with t, such as $g(t) = At^{-x}$ for small x, then it can be treated as a constant in the region of interest, namely $N_0 \ll t \ll W_0$. In this region, we can subsume the constant and near-constant terms of $g(t)$ into k and disregard the rest, giving:

$$E \approx k/t \tag{3}$$

Thus the mean time to failure is $1/E \approx t/k$ in units where each test takes one unit of time, and $S \approx N \approx N_0 - k \log t$. We shall see that E corresponds to the energy of a thermal system, while the 'energy' of an individual defect is simply its virility E_i.

Large software systems are very difficult to debug [7, 8], which suggests that they contain at least some processes with nontrivial values of k. To make this more precise, we will now describe the distribution of defects. Let there be $\rho(\epsilon)d\epsilon$ bugs initially with failure rates in ϵ to $\epsilon + d\epsilon$. From equation (1), their number will decay exponentially with characteristic time $1/\epsilon$, so that $E = \int \epsilon\rho(\epsilon)e^{-\epsilon t}d\epsilon \approx k/t$. The solution to this equation in the region of interest is

$$\rho(\epsilon) \approx k/\epsilon \tag{4}$$

This solution is valid for $N_0 \ll 1/\epsilon \ll W_0$, and is the distribution that will be measured by experiment. It differs from the ab-initio distribution because some defects will already have been eliminated from a well tested program (those in energy bands with $\rho(\epsilon) \sim \epsilon^x$ for $x > -1$), and other defects are of such low energy that they will almost never come to light in practical situations (those in energy bands with $\rho(\epsilon) \sim \epsilon^x$ for $x < -1$).

Equation (4) is in good agreement with experiment. Extensive measurements of the distribution of bugs in a large number of commercial software products found a near-universal distribution $\rho(\epsilon)$ over several orders of magnitude in ϵ [4]. The experimental data is in the form of tables showing measured values of $\epsilon\rho(\epsilon)$ (and not simply $\rho(\epsilon)$ as asserted by the author in his interpretation). The data there exhibit a cross-over from the behaviour $\rho(\epsilon) \approx \epsilon^{-1.5}$ at higher values of ϵ to $\rho(\epsilon) \approx \epsilon^{-1}$ at lower values of ϵ.

3 The engineering of reliable software

As noted above, it is widely accepted that if we need a mean time to failure of about 10,000 hours for a piece of software, then we need to test it for at least 10,000 hours [5]. This is of importance for public policy, as it impinges on the use of software in applications where very high mean times to failure are required, such as nuclear plant control and aircraft flight systems. We shall now show that this is in fact the best mean time to failure that we can expect given such a level of testing, and that the reliability will usually be less.

So far we have assumed random (also called operational) testing. In practice, however, test sequences are focussed; they may be biased towards the expected use of the programme, or to the areas of knowledge and experience of the tester. A simple example shows how we may quantify the effect of this. Suppose that a debugging team performs t tests on one K-th part of a cleanly divisible and uniform program. The tested part will then be in that same state as if a random test strategy had been adopted using Kt tests on the whole program, and the value of E will be the same in both cases. We conclude that if the quality of the system is initially uniform and correlations can be neglected, then the constant k in equations (3) and (4) is proportional to the size of the phase space tested.

So the failure time measured by a tester depends only on the initial quality of the program, the scope of the testing and the number of tests. The tester's measurements give virtually no further information about the quality of the program or its likely performance in a customer's environment, where the distribution of its input states may be quite different. This is in accordance with observation: students of safety-critical systems recognise that test results are often a poor performance indicator [3, 5, 7, 8]. It also accords with the authors' own experience of software development: a debugging team usually achieves a rapid improvement in failure time and quickly reaches the point of diminishing returns, as would be expected from equation (3). However, when the 'tested' software is passed on to another tester, a number of important bugs are often found quite quickly.

Mature software developers employ several test stages, designed or operated by different people, in order to reduce the correlations between tests. Our work suggests that these test stages will be more efficient if performed in parallel rather than series, as each tester will be finding bugs more frequently. Nonetheless, prolonged field testing is still very important for engineering dependable systems [9].

A trend in software engineering is to employ re-usable software components (or 'objects'). If a program is constructed from objects j that have each been subjected to a large number t_j of tests and are invoked on average once every n_j tests, the total failure rate is $E \approx \sum k_j / t_j n_j$. This shows that one can improve the reliability of software by using objects, provided they have been tested more thoroughly than the rest of the software in which they are deployed; this might be the case if they have already been extensively field tested in other products, or obtained from a company with a more thorough test department than one's own. However, the software's overall failure rate will then be dominated by terms that correspond to new code, and this will limit the achievable reliability gain.

The error correction process is imperfect in practice, so some authors have suggested replacing equation 1 with alternatives, such as $dN = -0.85Edt$ [4]. This leads to a similar analysis, using the definition $S = N + 0.85Et$. However, many professional software developers now use automated tests that can be repeated at will [10], so that any new bugs that affect the already-tested phase space can be quickly removed and equation (1) remains a good approximation.

One must often take account of the severity of the bugs as well as their numbers. For example, we might only be interested in those bugs which could conceivably cause loss of life or the leakage of classified information. However the above equations can be applied to the bugs of any given type, so long as the definition is consistent and a non-trivial number of bugs fall within it.

But however we define reliability, the difficulty of writing reliable software can be described in the language of thermodynamics as an equipartition property: each bug's contribution to the overall failure rate is independent of whether the component containing it is executed often or rarely. Intuitively, code that is executed less is also tested less; mathematically, a component that is executed on average once every n tests has $N \approx N_0 - k \log(t/n)$ and so its failure rate $-\partial N/\partial t$ is independent of n. Such analogies with statistical thermodynamics bring us to ask whether we can borrow any deeper insights from that field.

4 Free energy and Murphy's law

Let the probability that the i-th bug survives t tests be $p_i(t)$, and let the proportion of tests that it affects, if it is still present, be E_i. Then $dp_i = -E_i p_i dt$ and the bugs obey Poisson survival statistics $p_i = e^{-E_i t}$ [3]. This distribution is quite general provided that the statistics of the tests are static. For example, with focussed tests, the probability E_i of triggering a particular bug depends on whether it is in the area of focus, but any static and non-trivial statistics will yield stable E_i values and give Poisson survival statistics.

Consider the quantity $F = \sum p_i(\log p_i + E_i t - 1)/t$. This is unaltered if the distribution is varied by a small amount, since substituting $p_i = e^{-E_i t} + \lambda v_i$ (for some arbitrary v_i) yields $\partial F/\partial\lambda = 0$ and $\partial^2 F/\partial\lambda^2 > 0$ at $\lambda = 0$. This means that F is minimised for fixed t. Conversely, the basic equation (1) and the exponential distribution can be derived if we take as an axiom that F is minimised. In this sense, the properties of F mirror those of the free energy in a thermal system.

Macroscopically, $E = \sum E_i p_i$, $N = \sum p_i$ and $F = -N/t$. So the minimisation of the free energy F means that the number of defects N is maximised; that is, debugging removes the minimum possible number of bugs that must be removed in order to pass the test sequence. For example, bugs outside the area of test focus are not removed. This property appears to correspond with the informal principle called 'Murphy's law'.

Consider the quantity $S = \sum p_i(1 - \log p_i)$. We have $F = E - S/t$, so S is analogous to the entropy of a thermal system. We would expect it to be maximised at fixed E, and indeed we find that substituting $p_i = e^{-E_i t} + \lambda v_i$ yields $\partial S/\partial\lambda = t\partial E/\partial\lambda$ and $\partial^2 S/\partial\lambda^2 < 0$ at $\lambda = 0$. We also have $1/T = t = \partial S/\partial E$, which is the formal definition of temperature. This justifies our earlier definition of the temperature T of software as $1/t$; we have a distribution of defects that behaves statistically as if they were in thermal equilibrium at this temperature [11]. Note that the distribution is normalised differently from the classical distribution of excitations, which has a denominator $(1 + e^{-E_i/T})$, which is why the formulae in this section have some minor differences from their standard thermal counterparts.

We will now show that S is approximately proportional to the logarithm of the number of ways a large system can fail. If $W_F - 1$ of the W_0 distinct input states are processed incorrectly, then to a good approximation $E = W_F/W_0$ and so, from equations (1) and (3), $W_F \approx kW_0/t$ and $S \approx k \log W_F - C$ for some constant C. When there are no bugs left, $S = 0$ and $W_F = 1$, so we can take $C = 0$.

5 Thermodynamic properties of biological adaptation

Previous authors have considered the relationship between entropy and evolution at the level of flows of free energy in organisms and ecosystems [12]. However, we may also apply thermodynamic concepts to flows of information, such as the information in the genetic material itself. In this section, we will show that this approach can be used to derive a number of known results in population genetics; in the next section, we will use it to explain a puzzling phenomenon in the dynamics of viral evolution.

We can think of the defects in the DNA of a living organism as being introduced by spontaneous mutation and removed by evolutionary selection [13, 14]. We assume that there are a large number of defects of low severity in the genome.

Let E_i be the probability that a defect in the i'th base-pair will cause an individual to fail to reproduce, let p_i be the proportion of the population that carries the defect, and let μ be the probability that a spontaneous mutation arises in any given base-pair at each conception. Following an asexual genetic line through t generations, the microscopic equation of motion is

$$dp_i = (\mu - E_i p_i)dt \tag{5}$$

The properties we obtain below will be approximate because equation 5 is approximate. In particular, it does not account adequately for rare favourable mutations, and it neglects base-pairs that have no immediate effect on survival. In fact, the standard equation used by biologists is $dp_i = (\mu(1 - p_i) - E_i p_i)/(1 - E_i p_i)$, and S is usually used instead of E for selection; but equation (5) is accurate where p_i is small, as we assume, and will help develop the analogy with our software engineering model.

Multiplying equation 5 by E_i, and summing over the A active base-pairs produces $\partial E/\partial t = C - A < E_i^2 >$ where C is a constant. In a period of rapid evolutionary change, when C can be neglected, this is Fisher's fundamental theorem of natural selection, published in 1930, that 'the rate of increase in the fitness of any organism at any time is equal to its genetic variance in fitness at that time' [6, 15]. During periods of stability, $\partial E/\partial t = 0$ and $A < E_i^2 >= C$, so a better description (also noted by Fisher but not incorporated into his fundamental theorem) would be, 'the rate of increase in the fitness of any organism is proportional to its genic variance in fitness at that time, minus the steady-state variance in fitness'.

Equation 5 lets us establish general maximisation properties. We shall sketch the analysis, which is similar to that given above. The macroscopic equation of motion is $dN = (A\mu - E)dt$; the entropy is $S = \int t dE = Et - \int E dt = N + (E - A\mu)t - C'$ for some constant C'; and the free energy is $F = E - S/t = A\mu + (C' - N)/t$. Since the free energy is minimised at constant t, it follows that N is maximised.

In other words, a species adapts to its environment at minimal cost by removing as few defects as necessary and with as few early deaths as necessary. In this sense, natural selection is an optimal process. It is also a consequence of the maximisation property that any subsequent change in the environment will trigger defects that have not been selected out. This is in accordance with the observation that most environmental changes have a negative effect on fitness [6]. Finally, the maximisation of N means that there is an optimally large pool of genetic diversity to form the basis of future evolution — subject to limits set by population size via genetic drift.

Introducing sex makes the situation slightly more complicated, but introduces no fundamental problems; Fisher's theorem can again be shown to hold in appropriate circumstances [15].

6 When biological and software evolution are similar

During a period of rapid evolutionary change, such as after an environmental catastrophe, we can neglect the mutation rate, so μ in equation 5 becomes zero and the dynamics of biological evolution will mirror those of software development, giving $E = k/t$ at t generations after the start of the period. On the other hand, during periods of stability, $\partial p_i/\partial t = 0$. Solving equation 5 yields an equipartition property: $p_i E_i \mid_{steady-state} = \mu$ for all i, and summing over the A active base-pairs this becomes $E = A\mu$. Therefore

$$E \approx k/t + A\mu \tag{6}$$

This is in good agreement with experiment. Recently, the fitness of a large population of the bacterium E. Coli was measured for 10,000 generations after an environmental change [16]. The mean fitness was reported to follow $y = x_0 + at/(b+t)$, where x_0, a and b are constants. Writing $E - A\mu = x_0 + a - y$ and $k = ab$, this becomes $E - A\mu = k/t(1 + b/t)$. Each population in the experiment was founded from a single cell of an asexual clone, rather than from a diverse population, and so we might expect the denominator $(1 + b/t)$ to arise if it took an initial b generations to build up genetic diversity.

In a small system, with only a few defects, we would expect the failure rate to follow the underlying Poisson survival statistics of the defect with the largest E_i (the so-called strong law of rare events) and decrease exponentially. In very recent measurements [17] the effect of defects in the genome of an RNA virus were found to decrease exponentially with time. The agreement with experiment may be fortuitous because our simplified analysis does not include the effects of rare favourable mutations.

7 Conclusion

We have shown how to apply the concepts of entropy and free energy to the information content of complex adaptive systems. Our method applies generally to large systems that are governed or bounded by a superposition of Poisson survival statistics. We have shown in detail how it applies to two specific types of evolutionary system: software under testing, and a biological species under evolutionary pressure. Our mathematical model is in good agreement with the observed behaviour of both such systems, and explains a previously puzzling inconsistency between the e^{-kt} reliability growth of small systems and the k/t reliability growth of large ones. We have also proved a version of Murphy's law — that such evolutionary processes are asymptotically efficient, in that software testing removes the smallest possible number of bugs consistent with the regime of testing, while biological evolution removes the smallest possible number of unfit individuals consistent with the applied selective pressure. However, while Murphy's law is good news for an evolving species, it is rather bad news for the software developer.

We believe that our methods may be applicable to many other complex adaptive systems. Candidates include the extinction of species in an ecosystem under pressure, the response of the immune system to a new antigen, the evolution of medical practice in the light of studies of clinical outcomes, the history of processes within a company or an economy, learning in the brain, the defects in the corpus of scientific laws and mathematical proofs, and, presumably, any defects remaining in this article.

Acknowledgements: We wish to thank J Bennett, J Brookfield, D Coxell, AWF Edwards, NG Leveson, B Littlewood, J Knightley, J McMeekin, JR Partington and W Atmar for helpful discussions, and the Isaac Newton Institute for hospitality to the second author while the first draft of this paper was being written.

References

[1] Needham R. M., *personal communication*

[2] Maguire S., **Debugging the Development Process**, Microsoft Press, ISBN 1-55615-650-2 p 50 (1994)

[3] Littlewood B., *Predicting software reliability*, **Philosophical Transactions of the Royal Society of London** A327, pp 513–527 (1989)

[4] Adams E. N., *Optimising preventive maintenance of software products*, **IBM Journal of Research & Development**, Vol. 28, issue 1 pp 2–14 (1984)

[5] Butler R. W., Finelli G. B. *The infeasibility of experimental quantification of life-critical software reliability*, **ACM Symposium on Software for Critical Systems**, New Orleans ISBN 0-89791-455-4 pp 66–76 (Dec 1991)

[6] Fisher R. A., **The Genetical Theory of Natural Selection**, Clarendon Press, Oxford (1930); 2nd ed. Dover Publications, NY (1958)

[7] Leveson N. G., **Safeware: System Safety and Computers**, Addison Wesley. ISBN 0-201-11972-2. pp 26–38 (1995)

[8] Littlewood B., Strigini L., *The risks of software*, **Scientific American** special edition 'The Computer in the 21st Century' pp 180–185 (1995)

[9] Anderson R. J., Bezuidenhout S. J., *Cryptographic Credit Control in Pre-Payment Metering Systems*, **IEEE Symposium on Security and Privacy**, Oakland ISBN 0-8186-7015-0 (1995) pp 15–23

[10] **Automator QA**, software published by Direct Technology, Grove House, 551 London Road, lsleworth, Middx. TW7 4D5, UK.

[11] Feynman R. P., Leighton R. B., Sands M. **The Feynman Lectures on Physics**, ISBN 0-201-02116-1 Chapters 39–46 (1963)

[12] Schneider E. D., *Thermodynamics, Ecological Succession, and Natural Selection: A Common Thread*, **Entropy, Information and Evolution**, Weber B. H., Depew D. J., Smith J. D. (editors), MIT Press ISBN 0-262-23132-8 pp 107–137 (1988)

[13] Brookfield J. F., *Evolving Darwinism*, **Nature** vol. 376 pp 551–552 (17th August 1995)

[14] Burt A., *The evolution of fitness*, **Evolution** vol. 49 pp 1–8 (1995)

[15] Edwards A. W. F., *The fundamental theorem of natural selection*, **Biological Reviews** vol. 60 pp 443–474 (1994)

[16] Lenski R. E., Travisano M., *Dynamics of adaptation and diversification: a 10,000-generation experiment with bacterial populations*, **Proceedings of the National Academy of Sciences of the USA** vol. 91 pp 6806–6814 (July 1994)

[17] Novella I. S., Duarte E. A., Elena S. F., Moya A., Domingo E., Holland J., *Exponential increase in virus fitness during large population transmissions*, **Proceedings of the National Academy of Sciences of the USA** vol. 92 pp 5841–5844 (June 1995)