In and Out of Projects: a simple model and an experiment on the SourceForge development community

Paolo Crosetto
paolo.crosetto@unimi.it

December 10, 2008

Abstract

The allocation of time and effort within the communities of Open Source Software developers is an interesting yet relatively unexplored area. How can coordination be achieved, absent monetary rewards? How do developers choose where to direct their efforts amongst the thousands of existing software projects? How come the vast majority of Open Source projects is a failure, i.e. does not go beyond the announcement phase? The paper proposes a simple dynamic stochastic model that addresses these issues. The model is a non-strategic N-player dynamic interaction, in which players are asked to open, join or leave projects that are assigned a random probability of survival, a proxy for the project-launcher skills. The number and nature of open projects varying with time, players face a continuously changing landscape. The model is simulated using a simple agent-based code, and tested in a lab with human subjects. Results show that the model can replicate most of the stylized facts of the SourceForge.net dataset, namely the high number of lurkers hopping from project to project and the highly skewed distribution. The experimental evidence supports the main behavioral hypothesis of the model, interestingly showing that human subject tend to consistently launch more projects than maximizing behavior would imply.

Keywords. Open Source software, Experimental Economics, Agent-Based Models
JEL classification. L17

> *[...] Programmers' incentives give rise to an interesting form of 'network externalities' or 'strategic complementarities', as programmers assess the future as well as the current viewership of their contributions. The study of the resulting dynamics would have implications not only to the study of open source development and working, but also to other areas such as the evolution of scientific research across fields.*
>
> Lerner and Tirole (2001)

1 Introduction[1]

The production of Open Source software, a pure public good, by communities of mainly volunteer developers is a stunning fact that defies explication through traditional economic theory. The allocation of time and effort within the community of developers is another area in which we lack understanding. How can coordination be achieved, absent monetary rewards? How do developers choose where to direct their efforts amongst the thousands of software projects opened at any time, many of which appear to be just duplications of similar projects? How come the vast majority of Open Source projects is a failure, i.e. does not go beyond the announcement-planning phase?

Despite claims of the community being organised as a bazaar (Raymond, 1998), in which software is produced by myriads of little interventions by hundreds of developers in a scattered fashion, data collected from the *SourceForge.net* development website, on which ∼1.7 million developers are registered, taking part into ∼150.000 software projects, support a different picture of how the community works.

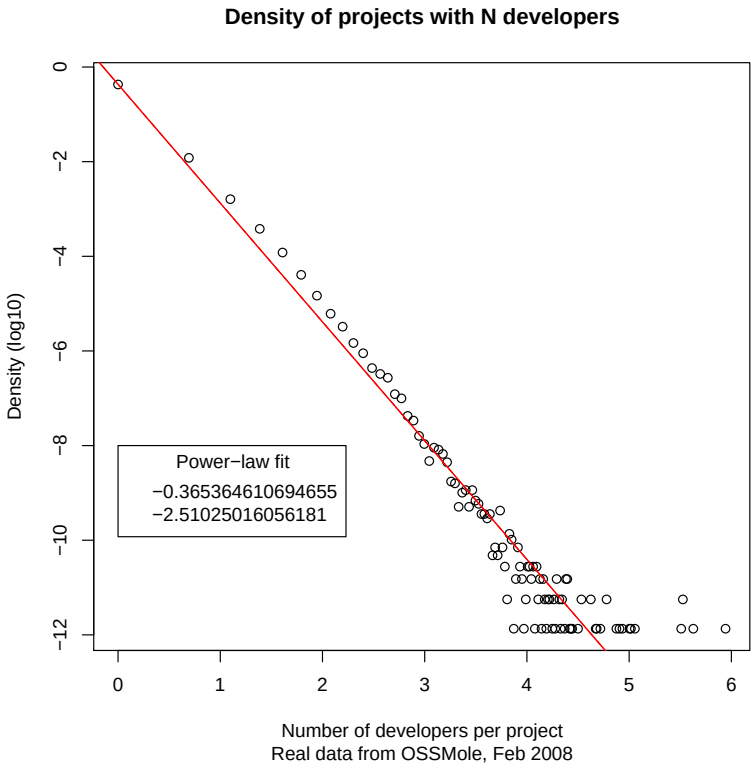

Figure 1: Developers in projects, SQL query on FLOSSmole (2008) data

As many studies and cooperative effort at data-mining (Madey et al., 2004; Weiss, 2005; Howison et al., 2006) show, only a fraction (∼10%) of the projects are active, and only very few of the active projects attract many developers and are real Free/Open Source (FOSS) success stories; moreover, new projects keep being produced, the huge amount of projects in idle state notwithstanding. The distribution of developers across projects is highly skewed, with the vast majority ($\sim 80\%$, that is more than a hundred thousand) of projects featuring

[1]This paper, and all the research it summarizes, have been produced using *only* Open Source software. I had to put this note somewhere. Here it is.

only one developer (i.e. the project owner), while only few projects have a significant number of developers contributing code ($n > 10$ is already a rare occurrence, let alone $n > 50$). The density of projects with N developers in them can be fitted by a power law (see Figure 1): this means that the distribution is highly skewed to the left, and has a very long right tail. Other indicators of project activity (bug opened and closed, feature requests, mailing list activity) show more or less the same skewed distributions; less effort-intensive activities - like asking for a feature to be added to the code - show less skewed distribution than more effort-intensive activities, like filing accurate bug reports, provide translations, and coding (for a detailed, if dated, survey of all data see Weiss (2005)). The distribution of developers across projects will be used as the central stylised fact of the data, and will be the focus of the model. Moreover, the data show a very high mobility of developers across projects, again featuring a highly skewed distribution, with few developers staying in the same project for a long time and the vast majority devoting at best discontinuous effort in many different projects (David and Rullani, 2006).

The picture that emerges is one in which very few big projects attract developers and activity, while the vast majority of projects are contributed to the community but end up being scarcely visible and do not attract any collective effort. Participation is very high, mainly in the form of launching small projects; success rate is extremely low, few big projects appearing; turnover of developers in projects and of success of projects is rather high. A project can gain very fast a big audience, and be *forked* and replaced in the medium term.

The typical success story of an Open Source project involves a brilliant young computer *geek* posting a more-or-less working initial release of a software on the Internet, gathering attention and support from fellow developers, successfully mastering the art of building a community of user-developers around his code, usually made up of a small bunch of core developers and a higher number of "lurkers" or occasional developers, at some stage attracting funding from firms interested in the code, and finally establish himself and/or the project as a standard software used by millions of people[2].

While an interesting and yet not completely answered question is "why would a 'top-notch programmer' decide to initiate a project in the first place by rendering freely available on the web code that might have significant commercial value if taken private?"(Rossi, 2006)[3], another set of interesting questions concerns the organisation of the community of developers, i.e. the incentives faced by the developers that *join* existing projects, working on someone else's small or big idea, or hop from a project to another, or else decide to set up their own project, hoping it would garner community support. More generally, the focus of an interesting set of issues could be the dynamics of contributions within the community at large (the set of active FOSS projects), and its effects on the nature and quality of active projects, and on the returns to developers taking part in the game[4]. Taking for granted the decision of developers to contribute code as a pure public good for free under a *copyleft* license such as the GPL (FSF, 2007), how do these developers coordinate, what projects do they join, how do they enter and leave, and what kind of picture emerges from their choices?

The paper proposes a simple dynamic stochastic model that addresses these issues. The model is a non-strategic N-player dynamic interaction, in which players open, join, leave, re-enter *projects*, their payoff being directly affected by their actions and indirectly linked to the other player's past actions, through the landscape of open projects available at any period.

[2]This is the rough baseline for a story of Linux, Apache, KDE, and many FOSS success stories; a variant of the story is one in which the initial more-or-less working software is provided by a company that used to develop it but finds it unprofitable to continue, and hence "frees" the software to the community: it is the case of Mozilla Firefox, or OpenOffice.org.

[3]The question has been tackled in the economics literature for some time now, see Johnson (2001); Lerner and Tirole (2000) for two different approaches

[4]For an analysis of the dynamics of contributions *within* projects, see Rullani (2006))

The interactions of players contribute in creating a dynamically evolving ecology of projects, a *project landscape*; the ecology of projects is let free to evolve, in an open-ended path-dependent process, within which the players make maximising choices.

Newly created projects are assigned as their unique characteristic a random probability, that stands as a proxy for the project-launcher skills and for the project quality. The probability is unknown prior to the choice but becomes public knowledge after the project is created, mimicking the fact that open source software quality is very easily assessed, since the code is open, i.e. readable, by everyone[5]. Players inherit the probability of the projects they are in as their *survival probability*. Every player is free to join any of the projects opened at any time, enjoying full information about the project characteristics; joining a project nonetheless entails a cost, since the joiners will inherit only a fraction of the project's probability, decreasing in their rank in the project. At the end of any period, a random process expels from projects the players that do not have a high enough survival probability; this process generates a flow of exits and re-entries in projects, allowing to analyze the issues related to the emergence of practices in the communities of developers with a lower number of players than the actual hundreds of thousand.

The model is simulated using a very simple agent-based simulation code, and tested in a lab with human subjects. Following methodological hints from Duffy (2006) and Epstein (2005), the paper combines the power of the two different methodologies of agent-based computational economics (ACE) and experimental economics (EE), using each of them for what it can do best[6]. The simulation is used to explore the aggregate outcome of the dynamic model in time, given a set of basic behavioural assumptions, exploiting computing power to explore the model with a high number of agents for many simulated periods. The experiment is instead used to assess the validity of the basic behavioural assumptions employed in the simulation and in the model, focusing on a narrower (but significant) number of subjects; the behavioural pattern that can be identified in the experiment can in turn be used to fine tune the parameters of the simulation.

Simulation results show that the model can replicate some of the stylized facts of the SourceForge.net dataset, namely the highly skewed distribution of players across projects, the emergence of few big projects and the perpetual novelty of new project creation. Moreover, the model operates as an evolutionary selection mechanism, the landscape of existing projects tending to have a higher average quality as the simulation progresses in time. The preliminary experimental evidence (only 3 sessions where held) supports the main behavioral assumptions of the model ($\sim 80\%$ of choices are predicted by the model), interestingly showing that errors can be grouped in categories, different errors being much more likely to occur in some situations rather than others. Human subject tend to consistently launch more projects than maximizing behavior would imply, and to have a somewhat lower preference for being leaders, even when this is suboptimal. The results of the experiment can then be used to fine-tune the behaviour of the simulated artificial agents[7].

The paper is organised as follows. Section 2 outlines the model structure and main assumptions and Section 3 reports the results of the simulation of the model, carried out with a very simple agent-based model written in Python. In Section 4 the case for running a human subject lab experiment alongside the simulated model is made, and the design chosen is detailed; the results of the experiment, both at the aggregate and individual level, as well as

[5] A computer program is a logically structured set of instructions, written in a human-readable formal language (the source code), that can be translated into binary code readable by the computer (the executable code) and used by this computer to perform a series of operations. Executable (binary) code is not easily decrypted, while source code is a set of text files written in standard computer languages.

[6] For an example of a well known paper adopting the same combination of ACE and EE, see Duffy (2001)

[7] This has not been done so far, since a deeper understanding of the behaviour of subjects in the experiment is needed. The experimental evidence is preliminary.

the implications of these results for fine-tuning the simulations and the model are outlined in section 5. Section 6 concludes, suggesting paths of future research.

2 The Model

The aim of this model is to build a simple and highly-stylized mechanism to capture the dynamics of contribtuions in the world of Open Source Software. The focus is on the choices of developers within the FOSS communities, taking for granted the choice of contributing to a public good. In other words, once a developer is committed to FOSS, i.e. has decided to contribute to a public good, which public good will be better for him to support? And what will the overall consequences of the choices be? Why do developers rather launch a new project than join existing ones, even when abandoned projects by the hundreds are strong evidence that success rates are very low?

The choice of the project(s) in which to devote one's effort seems to have aspects related to coordination games, in which the return from one's choice crucially depends on the other players' choices. It would be better to be the leader of a big project than the leader of a project that gets abandoned, the problem being that "success" or "failure" crucially depend on everyone else's choices. Many projects could be viable if the community put efforts into them, but ex-post only few attract such a big interest, success and failure being endogenous. Given these insights, a first attempt at modeling the inner workings of the FOSS communities has been made in a strategic dynamic N-player location setting (see the first chapter of the dissertation in Crosetto (2007)). It proved very difficult, though, to analytically tackle and experimentally explore a dynamic path-dependent N-players scenario with incomplete information and endogenous uncertainty, as that model happened to be; in the literature, such a comprehensive approach, including strategy, dynamics, externalities and feedbacks, plus a location dimension, has been pioneered only recently (Akiyama and Kaneko, 2000). The analysis could not be carried out in closed form, required extensive use of numerical calculations, and led to a situation of multiple equilibria; on the empirical side, the experiments proved to be plagued by coordination problems, the pilots were not successful, and the whole endeavour appeared cumbersome.

The choice made in order to overcome the difficulties was to strip the model of all simultaneous strategic interactions, setting the value of a project as exogenous, and randomly assigned. Though this came at a cost in terms of adherence to the real picture, the model is now dramatically and effectively simplified, while retaining some of its main characteristics.

The resulting model is a stochastic N-players dynamic model, in which agents do not act strategically, as their payoffs do not depend directly on other players' choices, but instead choose the best option taking all the setting as given. Nonetheless, the model is different from a simple choice problem, since it is dynamic and path-dependent, the choices of every player shaping the future available options.

2.1 Model setup

The main idea is that developers act rationally, committing a project in the hope of getting enough attention and fellow-contributors for the project to become a "star" in the FOSS community. The mechanism is quite simple: each developer faces a choice between joining someone else's project (at a cost) or launching one's own. Entry into, exit from or creation of projects is driven by an indicator of "project success", and to a cost of entry, that is increasing in the number of developers already in a project. Project quality rates are exogenously given

with a stochastic process, and once they are drawn they are public knowledge. In detail, the model works as follows:

1. Each of $N, (i = 1..N)$ players must simultaneously choose at every time period $t, (t = 1..T)$ between setting up a project (P) or joining someone else's (J); projects are indexed $j, (j = 1..J)$. Players can also switch from a project they are in into another already open project, leaving the project they are currently in. Players can be in only one project at any given period. The action space for any player is hence given by $A_i = \{P, J_1..J_J\}$
2. The time horizon is finite, of length T, and this is known to players[8].
3. Each time a player plays P, a project j is born with only one characteristic: a probability p_j, summarising the project qualities, drawn from a uniform $[0, 1]$ distribution. This probability stays the same over all the lifetime of a project, independently of how many people join the project, even in the case in which the initial project launcher leaves the project. A project dies, and is eliminated from the *project landscape* only when it has no more players in it. The maximum number of projects open in every period has an upper bound at N, the total number of projects opened in the T periods bounded as a consequence to NT.
4. Players inherit from the project they launch or join their personal probability p_{ij}. The personal probability of the players is a proxy of the player's abilities, and serves as a device to generate exit from and re-entry into projects:
   - when players *launch* projects, they inherit completely the probability of the project they created: $(p_{ij}|A_i = P) = p_j$.
   - when players *join* projects, they inherit the probability of the project they join, discounted by their rank in the project, i.e. by how many people are already in the project. So $(p_{ij}|A_i = J_j) = p_j - \delta\#_j$ in which $\#_j$ is the number of players already in project j at the moment of the choice, and δ is a parameter, $0 < \delta < 1$, that determines the cost of subsequent entries. This cost can be interpreted in terms of ability: if I am not good enough to have a brilliant idea, I can join someone brighter, and learn from him, even if this entails a cost. The more persons already in a project, the more difficult it will be for me to get in and hence the higher the cost of joining.
5. The personal probability p_{ij} acts as a probability of survival until the next period. At the end of each period each players draws a uniformly distributed $[0, 1]$ number d_i. If $d_i > p_{ij}$, the player is sent *out* of the project, while if $d_i \leq p_{ij}$ the player stays in the project he has chosen.
6. Being in a project at the end of the period yields a positive payoff, normalised to 1; being out entails a payoff of 0 for the current period.
7. All projects that still have players in them after the elimination phase will make up the project landscape faced by players in the next period.

The timeline of a period of the model is summarised by Figure 2. Every period is made up of four steps:

1. At the beginning of a period, every agent makes his choice given the *project landscape*. Note that at time $t = 0$ every agents launches a project, since no project exists.

[8] A first version of the model incorporated an indefinite time horizon, with a probability of the game ending after each period set to Π. Since the optimal strategies turned out to be independent of the time horizon, as we will see, for simplicity's sake a fixed length was assumed.

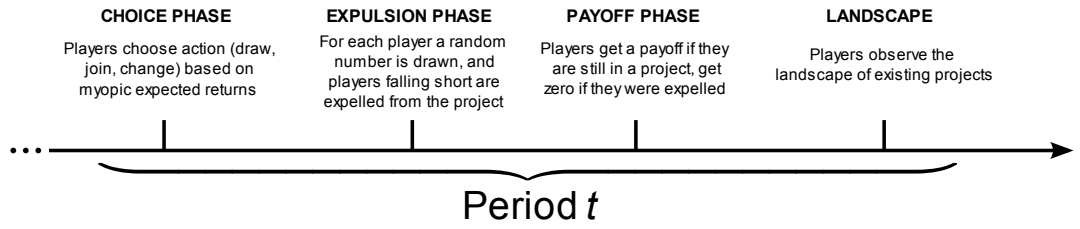

Figure 2: Model and simulation timeline

2. Death strikes: every agent draws a survival threshold, and if this is not met is expelled from the project he's in. Every project left with zero players in it disappears.

3. All agents that have not been eliminated receive a payoff of 1.

4. As a result of the preceding steps, a new *project landscape* appears: there will be up to N projects open, each with its own publicly known p_j, and each with a certain number of players in it; some players will be out.

The model rests on some basic assumptions.

One-dimensional quality: developers in the real world face a landscape of active and abandoned projects, and have to choose if and in which project to devote their effort. Projects do have different characteristics, and can give more or less return (in terms of satisfaction, peer-recognition, learning opportunities, skill-signaling or else, see Lerner and Tirole (2000)) to the potential joiner, can be more or less open to contributions and more or less difficult to integrate within. A first strong assumption is that all these characteristics can be resumed by only one index, $p_j \in [0, 1]$.

Public knowledge: the p_j for all projects is public knowledge. While this mirrors the fact that the source code is open, and hence the strength and promise of a code can be assessed by any prospective co-developer, the assumption of full public knowledge of project quality remains strong.

Elimination: in the real world, developers do leave projects and launch new ones at a fast rate, but the choice of leaving a project is not random and the turnover not as fast as in the model. The device of elimination is needed to generate a high flow of entries and exits from a low number of players, and is devised in a way as to hit only the worse developers, which are more likely, even in the real world, to hop from project to project.

No growth: projects do not grow, nor change, in terms of both p_j and payoff they give to the leader or the joiners. A project is born with a given $\bar{p}_j$ and dies with it. Leaders do not extra benefit from having launched a successful project, apart from having ha high p_{ij}. This assumption was needed in order to preserve the non-strategic nature of the game.

No memory of previous position: in the real FOSS communities, a player having contributed much code to a project is considered its "guru" even after he's left the project, and, should he come back, a place in the Gotha is held for him. In the model, when a player is expelled from a project, can re-enter it only with the rank position now available; the fact that he could have been the leader of the project up to the previous period is not taken into account.

2.2 Solving the model

In a first version of the model, the agents were supposed to maximise their expected utility over the whole length of the game, which was of indefinite length[9]. The model was solved setting up a Bellman equation and proceeding by backward induction. The player was supposed to maximise the lifetime expected payoff, given the assumption that he would behave optimally from the moment he will be eliminated from a project onwards. The Bellman equation turned out to be a simple function of p_{ij} and of $V0$, the value-to-go. This approach faced two problems. First, $V0$ had to be assumed to be independent of current and past choices by the player or by the other players. This was needed in order for the Bellman to be solved, but it was not justified on the ground of the model works. Imagine a player that stays for τ periods in a project, and at the end of period τ is expelled. The choice problem that he will face in period $\tau + 1$ depends on the projects that will be available at that time, and on how many other players will have joined which of these projects: it will be path-dependent. It is not possible, hence, to assume that $V0$ is a constant value, because it *does* depend on the particular history of the play up to that point. Second, even when $V0$ was assumed to be constant, it turned out that the optimal decision was always to act myopically, i.e. to choose the option that would have maximised the expected return from *only one* period.

Hence, the current version of the model was developed, featuring a fixed horizon and assuming myopic behaviour on the part of agents. Agents maximise their utility myopically, i.e. they choose the action that maximises their probability of being still in a project at the end of the period, thus earning the payoff; moreover, players are assumed to be risk-neutral, i.e. they have correct expectations on the stochastic process governing project launch. Since when launching a project (playing P) the project probability p_j is drawn from a uniform $[0, 1]$ distribution,

$$E(p_{ij}|A_i = P) = E(p_j \sim U[0, 1]) = 0.5.$$

The players will then choose to draw if and only if none of the available options give them $p_{ij} > 0.5$, and will otherwise choose to join the best available project. Since the project p_j are public knowledge, only the best project will be joined. The optimal strategy for any player would then be

$$A_i^* = \begin{cases} P & \text{if} \quad p_j < 0.5, \forall j = 1..J \\ \\ J_{j^*} & \text{if} \quad p_j* = max\{p_j, \forall j = 1..J\} > 0.5 \end{cases}$$

Note that joining the project one is currently in is labeled here as "joining", while it actually is a slightly different act, that implies staying. Since it is more intuitive, in the lab the subjects were told that the choice was between launching, joining and staying.

The optimal strategy is very simple, at the limit with triviality. The players just have to spot the highest p_j, and compare it with the expected value of drawing from a uniform distribution. In the real world, this could mean that players just have to choose between trying their luck as project leaders or join the project that is obviously the best. An equilibrium of the game is simply a situation in which everyone plays his optimal strategy. The game not being strategic, this is again a trivial condition to be met.

Nonetheless, the aggregate behaviour of the model and its dynamics show interesting patterns. Even assuming that all players always are in equilibrium, i.e. always choose their maximising strategy, the model shows the emergence of a skewed distribution of projects, the sustained creation of new projects, evolutionary movement towards better projects and lower amount of players *out*. But of course, these features appear only when we examine the aggregate path-dependent behaviour of the model over time; and this is what agent-based simulations are for.

[9]This was done by setting the probability of the game continuing at each time step equal to Π.

3 Simulation results

Agent-based Computational Economics (ACE) is now a well-established methodology within economics. Starting from the simple but powerful segregation model by Schelling (1971), a vast literature is now using computer programs to populate artificial worlds of more or less boundedly-rational, more or less myopic agents and explore the consequences of their more or less local interactions on the emergence of aggregate regularities. A comprehensive methodological and practical survey has been supervised by Tesfatsion and Judd (2006). The simulations performed for the simple model outlined above sit on the lower bound of ACE, being populated with very simple, maximising agents, that are just programmed to behave optimally in any situation. Nonetheless, the simulations share many features with more complex agent-based models: several artificial agents are put into an interactive dynamic situation of interest in order to better understand the aggregate implications of their micromotives on the system's macrobehaviour (Schelling, 1978)

The simulation has been coded using the scripting interpreted language Python, integrated by a whole set of scientific extensions[10].

The simulated model is in any detail identical to the formal model outlined above; agents were myopic maximisers, with no forward-looking attitude, no learning, no other-regarding preferences: they were just programmed to play their optimal strategy at every period. While in theory the simulations could accommodate a very high number of agents, computation needs - Python is very flexible but not very fast - limited the number of players to ~ 1000; results though showed no difference between populations of ~ 100 to ~ 1000 agents, and hence a lower number of agents was chosen.

The simulations run were parameterized with $N = 100$, $T = 500$, $\delta = 0.2$; moreover, the simulation was run for 100 runs and the results were averaged over these runs, to clean the data of any idiosyncratic stochastic effects.

The aggregate results of the simulations roughly match the main stylised facts of the SourceForge.net dataset, namely w.r.t the distribution of players across projects, and give some other hints on the aggregate behaviour of the model.

# of devs	Sim density	Real density
1	0.394	0.691
2	0.0817	0.1467
3	0.0567	0.0612
4	0.0339	0.0327
5	0.0393	0.0198
6	0.0307	0.0123

Table 1: Left tail of the distribution of projects by size, Simulated data

First, the distribution of players across projects was computed for each run of the simulation, and then averaged over the 100 runs performed. The result is a very skewed distribution, with a very high number of one-man project and some big projects. The first entries of the density of projects can be seen in Table 1: projects with one developer are very common, while the density of other project types drops instantly to low values. Note that this mass of one-man projects, that are created and then abandoned, are a result of *optimal play* and not of errors. It is optimal for a player to launch a project when there are no better alternatives

[10]For details, see Appendix A

available. As soon as someone else has opened a better alternative, or the elimination process has freed positions in other projects, it is optimal to abandon the project and let it die.

A log-log plot of the simulated data on the distribution of developers across projects can be found in Figure 3. The log-log distribution can be fitted by a power-law. The fit is performed using `gls` on the log of the variables, and is highly significant (at more than 1%). The actual data from the *SourceForge.net* archive can be seen in Figure 1 on page 2. While the slope and the intercept are not really the same, the overall simulated picture is quite similar to the real data. The simulation was performed with only 100 subjects, and in an admittedly rather trivial setting; nonetheless, the analysis of the data show that optimal behaviour by a number of agents put within the interactive context of a community as described by the model can lead to the creation of many abandoned small projects and just a few big projects.

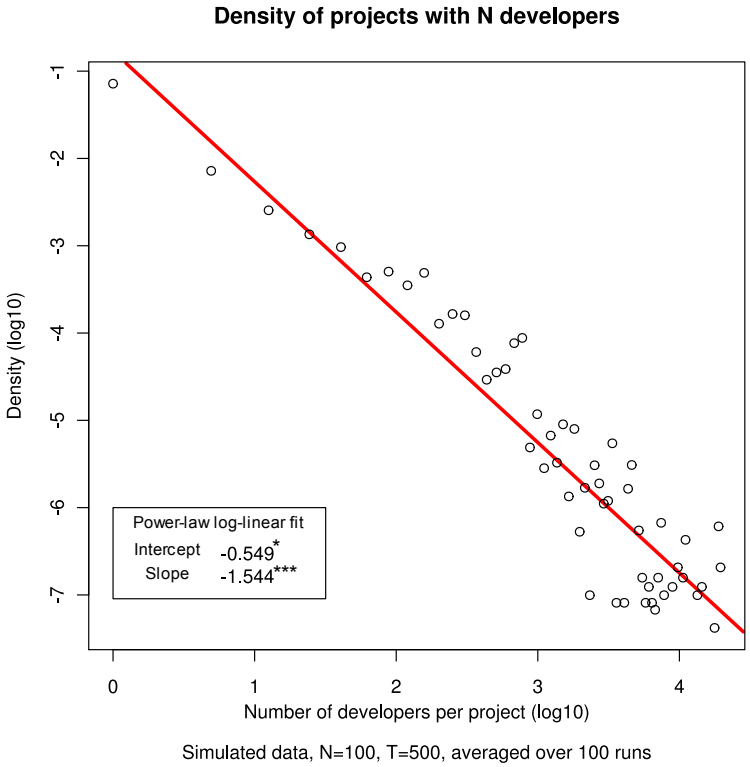

Figure 3: Simulated data, N=100, T=500, $\delta = 0.2$

The model works as an evolutionary selection mechanism, too. Since players are always choosing the best option available, the average quality of projects opened at any period grows. Moreover, players that are eliminated might find it optimal to launch a new project, that will be abandoned if not successful, but that, if successful, could contribute to give more and better options to all the other players. The average p_{ij} for each period has been plotted in Figure 4, right. The data show a highly significant positive relation between the average p_{ij} and time.

The very same mechanism can be seen in action in Figure 4, left, that summarises the number of players that are *out* in each period. This is an inverse relation w.r.t. the average personal probability seen on the right; again, the data show a highly significant negative relation: as periods accumulate, the average number of players *out* drops steadily.

Even if the rules followed by players are trivially simple - yet optimal - the aggregate results emerging from the simulations show at least two interesting features: 1. aggregate

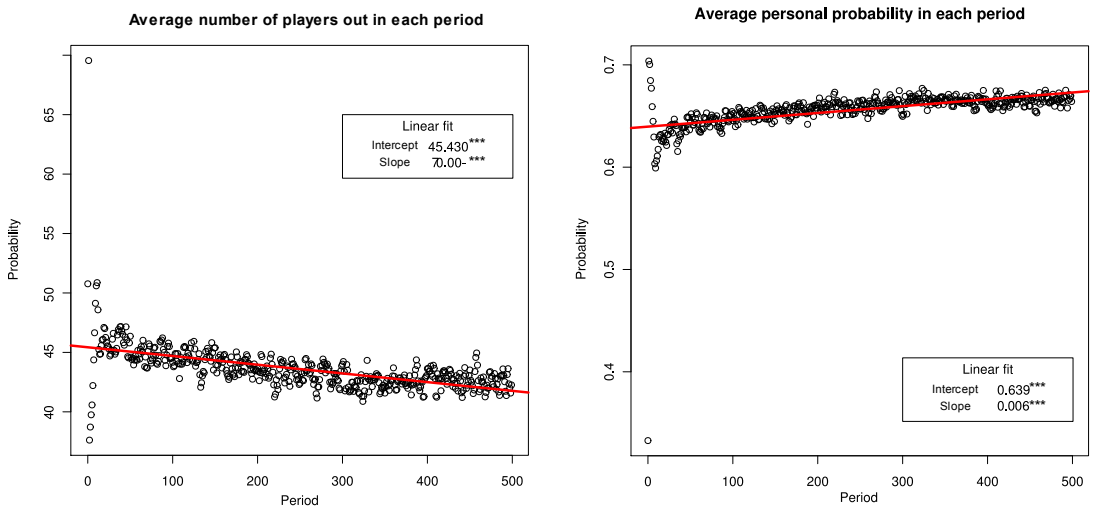

Figure 4: Simulated data, evolutionary effects of the model

distribution of players across projects shows features very similar to the real picture; 2. the model has an evolutionary nature, in which the overall quality of projects increases in time and the overall welfare of the participants increases too. It is now time to question the micro-foundations of the model: do real agents, when put into a situation as the one described in the model, behave as assumed - i.e. always spot and play the optimal strategy? This question can be properly answered by a lab experiment.

4 The experiment

Experiments have been present in economics since the 1940s as teaching and/or clarifying devices; nonetheless, in recent years experiments have gained mainstream status, especially after the award of the Nobel Prize to Vernon Smith in 2002. Experiments have been mainly used to test the assumptions of theories, with a strong focus on behaviour. Experiments can certainly serve other interesting purposes than behavioural assumption testing, and Vernon Smith's first experiments on markets were indeed *not* micro-behavioural experiments (Smith, 1962); nonetheless, experimental techniques have focused on predicting, eliciting, controlling and testing the behaviour of human subjects when dealing with economic choices[11].

In recent years there has been a partial convergence of Experimental Economics and Agent-Based Computational Economics. As argued by Duffy (2006), many studies have used ACE models to understand the macro implications of findings obtained in laboratory settings, while a fewer number of studies has gone in the inverse direction, trying to use lab experiments to refine the results of ACE models. This paper follows the last approach: a very simple agent-based model is used as a benchmark, and a human subject experiment is run in order to fine-tune the very simple behavioural assumptions of the ACE model.

The experiment was aimed at understanding subjects' behaviour when playing by the model rules. The subjects were put into the very same situation described by the model. For reasons of tractability and budget constraints, the number of subjects was set to 12, a number at which simulations showed that some interesting dynamics could emerge; moreover, the

[11] For a very clear and thorough methodological assessment of Experimental Economics, see (Guala, 2005)

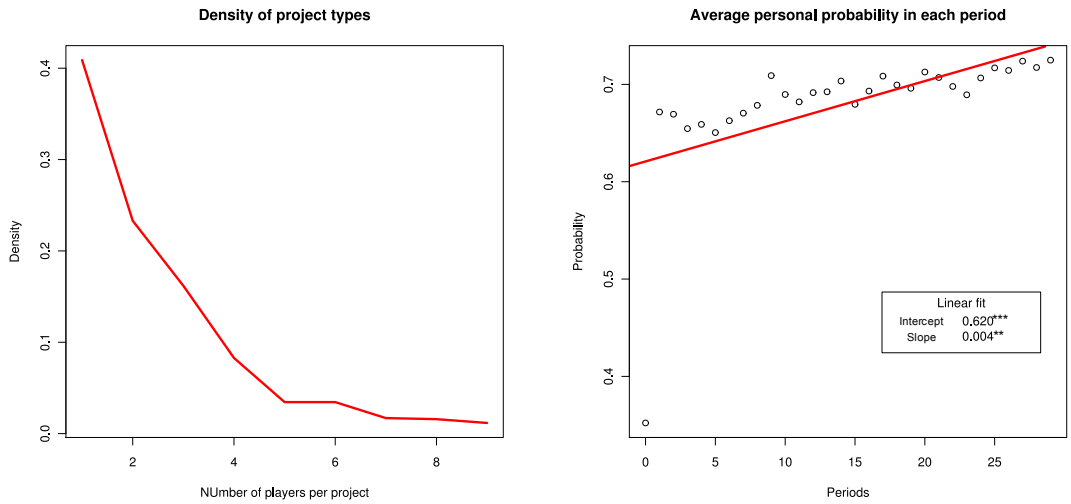

Figure 5: Simulations with $N = 12$. Left: Density of project sizes; Right: Average p_{ij}

game was played for 30 periods, to give the subjects room for learning and to see some action in the project landscape. Extensive simulations were run for the chosen parameterization, $N = 12$, $T = 30$, $\delta = 0.2$, the aggregate results of the simulation for such a low number of players being different from the ones detailed above. The main results for the simulations can be seen in Figure 5: a skewed distribution appears and the average p_{ij} of players goes up in time. Moreover, the total number of projects opened in a simulated session turned out to be 47.

The experimental design consisted of two treatment, between-subjects, i.e. with subjects performing only one treatment. Both treatment were based on 12 players, playing for 30 periods, with $\delta = 0.2$, and with the payoff accorded to player alive at the end of each period set to 50 eurocents. In the *Standard* treatment, the standard version of the model was played; 2 sessions of the standard treatment were run. In the *Pay-fee* treatment, all the setting remained unchanged, but joiners had to devolve 5 eurocents out of their payoff to the project-leader, if they were not eliminated; if eliminated, they did not have to pay the fee. Conversely, leaders collected fees from all joiners that were not eliminated. The *Pay-Fee* treatment had the same general solution already found for the basic model: even when taking into account the fee to be paid, the optimal strategy is to maximise the probability of surviving the current period, disregarding one's role in a project.

The *Pay-fee* treatment was chosen in order to test if human subjects would tend to have a bias favoring leadership, even if such leadership status should not enter their calculations of the optimal strategy; moreover, it was a first step towards a more complex but realistic model, in which leaders of successful projects can collect extra gains from their leadership status.

5 Experimental results

The experiment consisted of 3 session with 12 players in each; it was performed at University of Milano Bicocca EELAB[12]. Overall, the model could account for $\sim 80\%$ of the choices made, i.e. $\sim 80\%$ of the choices were optimal. The discussion of experimental results will be done as

[12]For details, see Appendix B

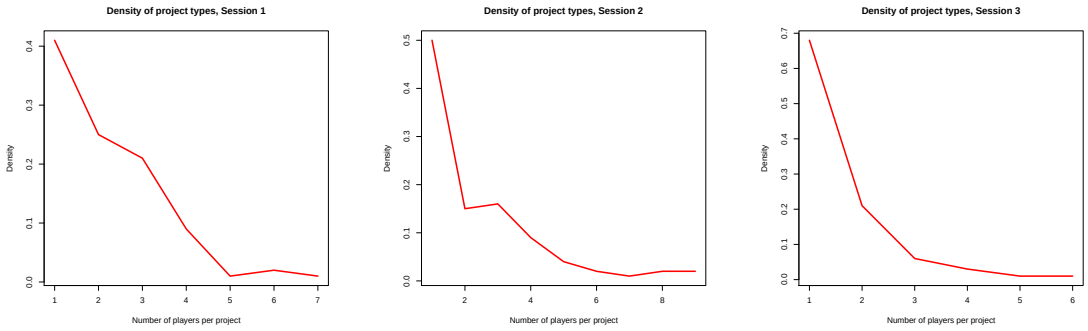

Figure 6: Density of project size, experimental data, sessions 1, 2 (standard) and 3 (pay-fee)

# of devs	Simulation	Session 1	Session 2	Session 3
1	0.408	0.41	0.5	0.68
2	0.233	0.25	0.15	0.21
3	0.161	0.21	0.16	0.06
4	0.082	0.09	0.09	0.03
5	0.0345	0.01	0.04	0.01
6	0.0345	0.02	0.02	0.01

Table 2: Left tail of the distribution of projects by size, Simulated data

follows: we will first outline the aggregate, macro results of the experiment, to appreciate that they are quite similar to the 12-players simulations discussed above; then we will perform some statistical tests on the 3 sessions, to see if there was a treatment effect; finally, we will try and fit a model of behaviour able to explain the mistakes that were made by the subjects.

5.1 Aggregate results

Aggregate results from the three sessions were quite similar to the results obtained by simulating the model with $N = 12$ and $T = 30$. Figure 6 displays the density of projects with different number of players in them for the three sessions. A skewed distribution appears, that is similar to the simulated one; actually, the distribution is *more* skewed in the experimental sessions than in the simulated data, as can be seen from Table 2

Moreover, the experiment replicated the evolutionary nature of the model: the average p_{ij} of the players in all three session showed increases very similar to the simulated ones; data could be fitted by `ols` with significant slopes in all three sessions. The three plots, albeit quite small, can be seen in Figure 7.

Finally, the total number of projects launched turned out to be different from the one predicted. The simulations predicted a value of 47 projects opened in the 30 periods; this datum for the three session turned out to be $S1 : 42$, $S2 : 82$, $S3 : 50$. Overall, since the players did behave as assumed four out of five times, the aggregate results could not look much different.

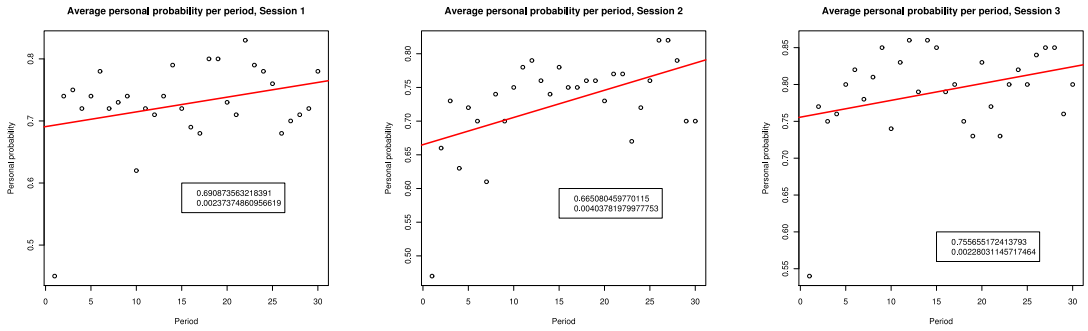

Figure 7: Density of project size, experimental data, sessions 1, 2 (standard) and 3 (pay-fee)

5.2 Are sessions different?

In order to assess whether there was a treatment effect or not, we have to show that the sessions produced indeed different results. The *pay-fee* treatment fo session 3 included an extra bonus for leadership that should not have altered the incentives faced by subjects, since the optimal strategy remained unchanged. If the treatment had effect, we could argue that human players look for leadership more when they get returns from that, even if these returns are not related to the optimal behaviour. If such a treatment effect existed, it could be the first step towards building a more comprehensive moel of developers behaviour, taking into account more linkages between rank in the project (or relative success) and payoff, turning the model into a more strategic and more complete game. The descriptive statistics of the quotas of optimal and suboptimal play in the three sessions are resumed in Table 3.

	S1	**S2**	**S3**	**Total**	%
Total # choices	348	348	348	1044	-
Non-optimal	67	74	69	210	-
%	19.25%	21.26%	19.83%	20.11%	-

Table 3: optimality in the three sessions

When considering aggregate error rates, the three sessions look indeed very similar. Standard non-parametric tests, like the Mann-Whitney U and the Kolmogorov-Smirnov tests, cannot reject the null hypothesis that the data come from the same empirical distribution. The results for these tests are resumed in Table 4.

		Session		
		S1 vs. S2	*S1 vs S3*	*S2 vs S3*
Optimality	KS	1	1	1
	MW	0.784	0.640	0.458
Error Type	KS	0.042	0.001	0.000
	MW	0.834	0.000	0.004

Table 4: Sessions - Kolmogorov-Smirnov and Mann-Withney tests

	Stay L	Stay J	Join Best	Join Other	Draw out	Draw L	Draw J
Stay	-	-	W	W	D	D	D
Join Best	L	C	-	W	D	D	D
Draw	L	C	P	W	-	-	-

Table 5: Types of errors

But not all errors are equal. Out of the $\sim 20\%$ suboptimal choices, we could identify five types of errors, according to the situation a player was facing at the moment of the error, and what the optimal choice at that precise time was (Table 5):

Conservative (C): players prefer to stay in the current project even if other, better projects are available. This case accounts for 13% of all errors.

Excess Draw (D): players should join an existing project, or switch to a better project, but choose to draw instead. This is by far the most common error, accounting for $\sim 57\%$ of all errors.

Leadership (L): players that are leaders prefer to stay in the project they lead instead of switching to the best available alternative. Note that in Standard treatment leadership is just a label; in Pay-fee treatment leadership does indeed generate value, but should not interfere with the decision. 19% of all errors, up to 34% on the Pay-fee treatment.

Prudent (P): It would be best to draw, but players join an existing project that gives them a survival probability < 0.5. Note: this never happened.

Plain Wrong (W): players fail to spot the best available option, and join a suboptimal project, knowing they will not be leaders. I don't find any other possible explanation that they were wrong. 10% of all errors

When considering the distribution of errors by type, the three sessions appear indeed different. As can be seen from Table 4, the demanding Kolmogorov-Smirnov test strongly rejects the null hypothesis of equality between samples between sessions 1 and 3 and between sessions 2 and 3, while rejecting, but not as strongly, equality between sessions 1 and 2, which by design were supposed to be equal. The Mann-Whitney test, though, while still rejecting equality of samples between S1 and S3 and between S2 and S3, cannot reject the null in the case of S1 *vs* S2, as we would expect. The evidence is then mixed, but we can say that, at the level of error type, session 3 was indeed different from the two *standard* sessions, even if these two were not so different from one another.

The sample being so small, the differences between sessions 1 and 2 could be traced back to two, very different, outliers in the experimental subjects: in session 1, a subject was "plain wrong" nine out of thirty times, and ended up earning the lowest payoff of all the experiment, while in session 2, a subject adopted the strategy of always launching new projects, making an excess draw error twenty-two out of thirty times, earning below-average payoff.

Session 3 induced different types of errors in the subjects (see Table 6). Leadership errors increased fourfold, fueled by the players' attempts to earn the joiners fees, and Conservative errors decreased to zero, as staying in the "wrong" projects as joiners was clearly now a less attractive alternative. All these changes in error types were due to the framing of the problem, and to a small alteration of the payoffs that should not have changed the way a rational player behaves.

Error Types					
Conservative	22	7	0	29	13.81%
Excess Draw	24	58	38	120	57.14%
Leadership	9	7	24	40	19.05%
Prudent	0	0	0	0	0%
Plain wrong	12	2	7	21	10.00%

Table 6: Distribution of error types across sections

5.3 A model of individual behaviour

A more ambitious approach to analysing the data of the three experimental sessions is to try and fit a model of behaviour that could describe both the optimal choices *and* the errors. Since no subject ever committed Prudent errors, the possible outcomes of the experiment were of five types: the optimal action plus four different error types. The optimal choice is well defined and unique for each period, but not across subjects: subjects take part in the game at each period from different angles, or perspectives, and the optimal action is conditional on their current state and the overall landscape they face. But nonetheless, it is in principle possible to attach a probability to each type of error, and then model these probabilities as functions of the subject's peculiar characteristics, such as the demographic variables collected in the experiment through a questionnaire, or other indicators of the situation the player was facing at the moment.

After having defined probabilities for each possible outcome, and having specified a model of how these probabilities could depend on various subject-specific variables, it is possible to estimate the impact of any of these variables on behaving optimally or committing one of the four error types with maximum likelihood.

While this has not yet been done, a first trial in this direction has been attempted. Let's define the probabilities attached to each type of error as:

p_c**:** probability of committing "conservative" error. Optimality would imply to switch to a new project, but player stays in current project.

p_d**:** probability of committing "excess draw" error. Optimality would imply joining, or staying, but the player draws.

p_l**:** probability of committing "leadership" error. Optimality would imply switching, but player stays in the project in which he is leader.

p_w**:** probability of committing "plain wrong" error. Optimality would imply join project j, but player joins k instead, with $VALUE(j) > VALUE(k)$.

$1 - p_c - p_d - p_l - p_w$**:** residual probability of having chosen the "correct" option.

Each of these probabilities is a function of several subject-specific variables. The main attempt of the exercise is to try and explain excess draws, as they appear to be the most frequent mistake made by human subjects with respect to their electronic counterparts. For simplicity, the three sessions have been pooled to have more observations. The variables that have been identified so far for this exercise are:

demographics: sex, age, economics, hi-school-grade. [demo]. Behaviour could be explained by gender, age, or prior high-school performance, or else the variables could just act as controls.

p-difference: difference in value from current p_{ij} to the probability that could be attained if the player moved to the best available option [pdiff]. The idea is that with a small premium for change, a player might be induced in committing Conservative and Leadership mistakes.

out: a dummy that takes value of 1 if the player was eliminated in period $t-1$, and 0 otherwise [out-1]. A descriptive analysis of the data hints at a relationship between being *out* and excess draw: when a player is *out*, it could be more likely that he will launch a new project even if there is a project available with a p_{ij} available for him that is higher than the rational expectations from launching a new project. In a sense, being *out* could increase the risk-propension of the subject, while a player that comfortably sits in a project and is not eliminated could think twice before leaving his place to risk a new project on his own.

max-available-value: the maximum p_{ij} available to a player [maxavval]. If the available maximum probability is low, even if higher than the expectation from a draw, the player might be tented to draw. When faced with $p_{ij} = 0.7$ for sure or a draw, a player might be joining, while if the cost of risk was reduced, and say $p_{ij} = 0.55$, the same player could be trying his luck and draw.

habit: average of the p_{ij} experienced by a player in the previous τ periods [habit]. Players could become accostumed to high personal probabilities, and, when expelled from a project, could prefer to draw rather than join an existing project with a lower personal probability.

numerosity: number of available options. [num]. If there are many projects available on screen, it is more likely that players will just be plain wrong, out of distraction.

With these variables set, a multinomial logit regression has been run on the pooled data for the three sessions. The dependent variable is a polytomous variable, set to 0 if the player behaved optimally, and getting values from 1 to 4 for the four types of errors (C,D,L,W), representing mutually exclusive outcomes. The data being organised as a panel with dimensions $N = 36$ and $T = 30$, the usual mixed logit model could not be directly applied; as a first approximation, a mixed logit was run, pooling observations by subject.

The results are not very significant, and not very robust to changes in the specification of the model. They are summarised in Table 7 just for reference, since the exercise was in any case a preliminary analysis. Coefficients measure log-odds deviations from the baseline case, assumed to be optimal behaviour.

The results shown in Table 7 only partially support the main claims made while introducing the different regressors. While some signs are in line with intuition coming from a casual look at the data, others are clearly pointing in a direction that is not supported by the data, or not even possible within the model: as an example, being *out* in the previous period can *not* have an effect on the likelihood of committing type C or L errors, simply because these errors can be committed only by a player that was still in the project the previous period. The estimation are at the moment very rough. What needs to be done is to develop a custom log-likelihood function and then estimate a custom model, possibly taking into account the panel dimension of data.

	pdiff	out-1	maxavval	habit	num
C	+	+	-	NS	+
D	+	NS	NS	NS	+
L	-	+	+	NS	+
W	-	+	NS	NS	+

*Signs are shown only for coefficient that pass .05 significance

Table 7: Preliminary results of mixed logit, sign of the coefficient, NS = not significant

Correctly estimating behaviour is important because it could lead us to the next step, the fine-tuning of the simulation model in order to better fit experimental evidence, and then to new simulation runs in order to assess the macro consequences of the micro changes in behaviour that the experiment highlighted.

6 Conclusion

The model deals with the allocation of time and effort within the communities of Open Source Software developers; its main result is that high numbers of abandoned or one-man project can coexist with a handful of big and growing projects, and that launching a project and then abandoning it can indeed be the result of optimal play, as long as the players are all linked together in an N-person, dynamic, path-dependent game. A skewed distribution of players across projects emerges from the interactions of very simple agents, programmed to maximise their current expected utility without looking into the future, or taking into account the other player's actions. In this sense, the model can account for the high rate of failures that can be seen in online repositories: developers can find it optimal to launch their own project even in the presence of valuable alternatives, because entering someone else's project entails a cost, and does not give enough benefits if the project is already too *crowded*; ex-post, though, the project launched yields lower-than-expected results, and is then abandoned.

A crucial assumption of the model is the complete information enjoyed by the players about the quality of existing projects. This assumption seems grounded, since Open Source software is indeed open, and hence every developer can use the software for free, and also have a look inside the black box to see how it works. In this respect, the model could also partly account for the success of Open Source software: the source code being open, developers find it much more easy to make optimal choices; moreover, the model shows evolutionary characteristics, the simulations showing that as a result of the interplay between players, in the long run only the better projects survive and the developers are all better off. To assess how much these results depend on perfect information, a new treatment with less availale, costly, hidden or private information could be developed.

The model is non-strategic, in order to keep things very simple, and allow us to perform thorough simulations and a behavioural experiment. The experimental results support the main behavioural assumption, i.e. that in case of complete informations the players are able to spot the best strategy, but interestingly highlights some consistent patterns of human behaviour, namely the excess confidence, resulting in an excess number of projects being launched, and the leadership bias, resulting in players sticking to their leadership status in a project, even when it would not be optimal to do so.

While a full-fledged model to account for the behaviour of subjects in the experiments is still to be developed, these insights will be very important in changing the basic assumptions

of the model; after having established a new model, or a new set of possible behaviours within the model, new simulations could be run and possibly result in a better fit with the real data.

Finally, the model, and the methodology adopted to tackle the issue, could form the core of a more complete exploration of the issues raised by the existence, and sustained growth, of Open Source software and of other Open contents on and off the Internet, including the study of all the new forms of decentralised cooperation among strangers, that seem to replace the markets in a growing part of our daily experiences.

References

Akiyama, Eizo and Kunihiko Kaneko, "Dynamical systems game theory and dynamics of games," *Physica D*, 2000, *147*, 221–258.

Crosetto, Paolo, "The role of interactions within the community in the development of FOS software: an experiment." Chapter 1, DEAS, University of Milano 2007.

David, Paul A. and Francesco Rullani, "Micro-dynamics of Free and Open Source Software Development. Lurking, laboring and launching new projects on SourceForge," LEM Papers Series 2006/26, Laboratory of Economics and Management (LEM), Sant'Anna School of Advanced Studies, Pisa, Italy October 2006. available at http://ideas.repec.org/p/ssa/lemwps/2006-26.html.

Duffy, John, "Learning to speculate: Experiments with artificial and real agents," *Journal of Economic Dynamics and Control*, 2001, *25*, 259–319.

_ , *Agent-based models and human subject experiments*, Vol. Handbook of Computational Economics, Volume 2., Elsevier,

Epstein, Joshua M., "Remarks on the Foundations of Agent-Based Generative Social Science," Working Paper 41, The Brookings Institution, Washington, DC Jul. 2005.

Fischbacher, Urs, "z-Tree: Zurich Toolbox for Ready-made Economic Experiments," *Experimental Economics*, June 2007, *10* (2), 171–178.

FLOSSmole, "Collaborative collection and analysis of free/libre/open source project data," http://ossmole.sourceforge.net/ 2008.

FSF, Free Software Foundation, "GNU General Public Licence, version 3," http://www.gnu.org/copyleft/gpl.html June 2007.

Guala, Francesco, *The Methodology of Experimental Economics*, Cambridge University Press, 2005.

Howison, J., M. Conklin, and K. Crowston, "FLOSSmole: A collaborative repository for FLOSS research data and analyses.," *International Journal of Information Technology and Web Engineering*, 2006, *1(3)*, 17–26.

Johnson, Justin Pappas, "Economics of Open Source Software." PhD dissertation, M.I.T. 2001.

Lerner, Josh and Jean Tirole, "The Simple Economics of Open Source," NBER Working Papers 7600, National Bureau of Economic Research, Inc March 2000. available at http://ideas.repec.org/p/nbr/nberwo/7600.html.

_ **and** _ , "The open source movement: Key research questions," *European Economic Review*, 2001, *45*, 819–826.

Madey, Gregory, Vincent Freeh, and Renee Tynan, *Modeling the Free/Open Source Software Community: A Quantitative Investigation*, Idea Publishing,

Raymond, Eric S., "The Cathedral and the Bazaar," http://www.catb.org/ esr/writings/cathedral-bazaar/cathedral-bazaar/ 1998.

Rossi, Maria Alessandra, *The economics of Open Source Software development*, Elsevier, Amsterdam, 2006.

Rullani, Francesco, "Dragging developers towards the core," CESPRI Working Papers 190, CESPRI, Centre for Research on Innovation and Internationalisation, Universita' Bocconi, Milano, Italy November 2006. available at http://ideas.repec.org/p/cri/cespri/wp190.html.

Schelling, Thomas C., "Dynamic Models of Segregation," *Journal of Mathematical Sociology*, 1971, *1*, 143–186.

_ , *Micromotives and Macrobehaviour*, Norton, New York, 1978.

Smith, Vernon L., "An Experimental Study of Competitive Market Behavior," *Journal of Political Economy*, 1962, *70*, 111.

Tesfatsion, Leigh and Kenneth L. Judd, *Handbook of Computational Economics, Volume 2, First Edition : Agent-Based Computational Economics (Handbook of Computational Economics)*, North Holland, June 2006.

van Rossum, Guido, "Python Reference Manual," CWI Report May 1995.

Weiss, David, "A Large Crawl and Quantitative Analysis of Open Source Projects Hosted on SourceForge," Technical Report, Institute of Computing Science, Pozna University of Technology, Poland 2005.

Appendices

A Details on the simulation of the model

The simulation was written in Python scripting language (van Rossum, 1995), using the scientific extensions in Pylab (NumPy, SciPy, MatPlotLib) and the R-Python plugin (RPy). Python is definitely not a fast computing language, but is very easy, flexible, and thanks to the extensions can be used to easily build a simulation from the ground up, automating the collection of statistics and the making of graphs directly within the simulation program. NumPy provides python with fast array manipulation and calculation; SciPy integrates maximisation routines and other useful mathematical computation abilities in Python; MatPlotLib provides extensions to generate .pdf, .svg, .png and other types of plots directly from the simulation code; finally, RPy allows to use R commands directly into the Python code. This last feature was not used in the code, but turned out to be useful in building the graphics. The code is available on demand.

B Experimental details and instructions

The experimental sessions and the pilot were conducted in the month of November, 2008, in Milano, Italy.

The pilot was held with my PhD colleagues in Università di Milano Lab, November 14th, 2008. It featured 6 subjects, and had the sole intention of testing the experimental software. The data collected were not analysed and are not included in the results shown in this paper.

The three experimental sessions were held in the Università di Milano-Bicocca Experimental Economics Lab (EELAB), November 20th, 2008. Three treatments, with 12 subjects each, were conducted. 24 subjects were tested on the *Standard* treatment, and 12 on the *Entry Fee* treatment. Subjects were recruited amongst undergraduate students, using standard recruiting procedures. The experiment was programmed and conducted with the software z-Tree (Fischbacher, 2007). The 3 sessions had an average payment of slightly more than 11 €, for an overall cost of $397,2$ €

The instructions (in Italian) for the two treatments follow. The z-Tree binary treatment file (or its exported text counterpart) are available on demand.

B.1 *Standard* Treatment

Benvenuto!

Stai per prendere parte a un esperimento economico. Ti sarà chiesto di prendere alcune semplici decisioni, e in base ai risultati delle tue decisioni potrai guadagnare fino a un massimo di 15 euro. Le scelte che farai e i dati che fornirai nel corso dell'esperimento sono anonimi. Sarai identificato unicamente dal numero del PC a cui sei seduto. All'entrata ti è stato un foglietto con il numero del PC a cui ti sei seduto: conservalo. L'esperimento durerà circa un'ora. alla fine dell'esperimento ti verrà immediatamente corrisposto quanto da te guadagnato. Dovrai compilare la semplice ricevuta che hai trovato nella tua postazione, e consegnarla unitamente al foglietto con il numero del PC. L'esperimento dura 30 periodi. In ogni periodo dovrai fare una semplice scelta: dovrai decidere se avviare, oppure entrare in un progetto.

Ad ogni periodo potrai scegliere se:

1. Avviare un progetto: un progetto viene creato. Il progetto ha un'unica caratteristica: la probabilità di sopravvivere fino al periodo successivo. Quando crei un progetto, viene assegnata al progetto una probabilità di sopravvivenza; questa probabilità è un numero casuale tra zero e uno, in cui ogni valore ha la stessa probabilità di essere estratto. La probabilità che verrà assegnata al progetto ti è ignota prima che tu scelga di avviare un progetto. Se sei l'iniziatore del progetto, ne sei il leader e ne erediti interamente la probabilità di sopravvivenza: la tua probabilità di sopravvivenza è esattamente quella del progetto che hai appena creato.

2. Entrare in un progetto esistente: se esistono progetti aperti da altri giocatori, puoi scegliere di entrare in uno dei loro progetti. In questo caso erediterai la probabilità di sopravvivenza di quel progetto, diminuita secondo il tuo grado nel progetto. Essere il secondo a entrare in un progetto implica una probabilità di sopravvivere minore del leader, ma maggiore rispetto a essere il terzo, o il quarto... e cos via.

E' sempre possibile abbandonare il progetto in cui si è per aderire a un altro; ma questo implica lasciare il progetto corrente, in quanto si può essere solo in un progetto alla volta. Questo significa che in ogni periodo potrete fare una e una sola scelta, cioè barrare una e una sola casella. Sullo schermo appariranno tutte le opzioni aperte in ogni periodo, con la probabilità di sopravvivenza (già scontata per il grado che andrai a ricoprire) e il grado che otterrai se entri nel progetto. Alla fine di ogni periodo, un secondo numero casuale compreso tra zero e uno viene estratto: è la soglia di passaggio del turno. Se questo numero è più alto della tua probabilità di sopravvivenza, sarai eliminato e fuori dai giochi fino alla fine del turno; se è uguale, o più basso, sei ancora in gioco. Tutti i giocatori ancora in gioco alla fine del turno ricevono 50 centesimi di euro; i giocatori eliminati non ricevono nulla. I centesimi guadagnati vengono accumulati nel corso dell'esperimento. Alla fine di ogni periodo sullo schermo appaiono il tuo guadagno nel periodo corrente e il guadagno accumulato fino a quel punto. Se sarai eliminato, inizierai il periodo successivo fuori dai giochi; potrai nuovamente scegliere se entrare in uno dei progetti esistenti, con la probabilità e nel ruolo previsti da quel progetto, oppure se lanciare un tuo progetto, estraendo una nuova probabilità di sopravvivenza casuale compresa tra zero e uno. Se sarai in gioco, inizierai il periodo successivo all'interno del progetto in cui sei entrato nel periodo precedente. Avrai comunque a disposizione 3 scelte: restare nel progetto in cui sei, unirti ad un altro progetto lasciando quello in cui sei, oppure avviare un progetto nuovo, estraendo una nuova probabilità di sopravvivenza. Nota che l'eliminazione è sempre individuale: vengono eliminati i singoli giocatori, NON i progetti. Un progetto resta attivo finch'e ha un giocatore al suo interno, anche quando il giocatore che per primo l'aveva iniziato dovesse essere eliminato.

Prima di iniziare i periodi dell'esperimento vero e proprio, verranno svolti 5 periodi di prova, in cui potrai impratichirti con il funzionamento dell'esperimento. I centesimi guadagnati in questi periodi non danno diritto ad alcun pagamento, e saranno quindi azzerati prima di iniziare l'esperimento vero e proprio. Lo sperimentatore ti dirà con chiarezza quando finiranno i periodi di prova e inizierà l'esperimento vero e proprio, in cui i centesimi guadagnati saranno effettivamente pagati. Alla fine del trentesimo periodo, apparirà sul tuo schermo un breve questionario. Ti ricordiamo ancora che i dati che immetterai saranno anonimi. Dopodich'e, verrà visualizzato il tuo guadagno finale, in euro. Subito dopo l'esperimento, lo sperimentatore procederà al pagamento di quanto guadagnato. Una volta che l'esperimento sarà finito, compila la ricevuta e presentati al banco dello sperimentatore con la ricevuta compilata e con il foglietto su cui è indicato il numero del tuo PC.

Domande?

B.2 *Entry Fee* Treatment

Benvenuto!

Stai per prendere parte a un esperimento economico. Ti sarà chiesto di prendere alcune semplici decisioni, e in base ai risultati delle tue decisioni potrai guadagnare fino a un massimo di 15 euro. Le scelte che farai e i dati che fornirai nel corso dell'esperimento sono anonimi. Sarai identificato unicamente dal numero del PC a cui sei seduto. All'entrata ti è stato un foglietto con il numero del PC a cui ti sei seduto: conservalo. L'esperimento durerà circa un'ora. alla fine dell'esperimento ti verrà immediatamente corrisposto quanto da te guadagnato. Dovrai compilare la semplice ricevuta che hai trovato nella tua postazione, e consegnarla unitamente al foglietto con il numero del PC. L'esperimento dura 30 periodi. In ogni periodo dovrai fare una semplice scelta: dovrai decidere se avviare, oppure entrare in un progetto.

Ad ogni periodo potrai scegliere se:

1. Avviare un progetto: un progetto viene creato. Il progetto ha un'unica caratteristica: la probabilità di sopravvivere fino al periodo successivo. Quando crei un progetto, viene assegnata al progetto una probabilità di sopravvivenza; questa probabilità è un numero casuale tra zero e uno, in cui ogni valore ha la stessa probabilità di essere estratto. La probabilità che verrà assegnata al progetto ti è ignota prima che tu scelga di avviare un progetto. Se sei l'iniziatore del progetto, ne sei il leader e ne erediti interamente la probabilità di sopravvivenza: la tua probabilità di sopravvivenza è esattamente quella del progetto che hai appena creato. Avviare un progetto comporta un beneficio: in qualità di leader del progetto incasserai i pedaggi di entrata che verranno pagati da tutti coloro che saranno entrati nel tuo progetto, se saranno in gioco alla fine del periodo.

2. Entrare in un progetto esistente: se esistono progetti aperti da altri giocatori, puoi scegliere di entrare in uno dei loro progetti. In questo caso erediterai la probabilità di sopravvivenza di quel progetto, diminuita secondo il tuo grado nel progetto. Essere il secondo a entrare in un progetto implica una probabilità di sopravvivere minore del leader, ma maggiore rispetto a essere il terzo, o il quarto... e cos via. Entrare in un progetto ha un costo: qualora tu sia in gioco alla fine del turno, dovrai pagare 5 centesimi di euro al leader del progetto in cui sei entrato; qualora tu venga eliminato alla fine del turno, non pagherai nulla.

E' sempre possibile abbandonare il progetto in cui si è per aderire a un altro; ma questo implica lasciare il progetto corrente, in quanto si può essere solo in un progetto alla volta. Questo significa che in ogni periodo potrete fare una e una sola scelta, cioè barrare una e una sola casella. Sullo schermo appariranno tutte le opzioni aperte in ogni periodo, con la probabilità di sopravvivenza (già scontata per il grado che andrai a ricoprire) e il grado che otterrai se entri nel progetto. Alla fine di ogni periodo, un secondo numero casuale compreso tra zero e uno viene estratto: è la soglia di passaggio del turno. Se questo numero è più alto della tua probabilità di sopravvivenza, sarai eliminato e fuori dai giochi fino alla fine del turno; se è uguale, o più basso, sei ancora in gioco. Tutti i giocatori ancora in gioco alla fine del turno ricevono 50 centesimi di euro; i giocatori eliminati non ricevono nulla. I centesimi guadagnati vengono accumulati nel corso dell'esperimento. Alla fine di ogni periodo sullo schermo appaiono il tuo guadagno nel periodo corrente e il guadagno accumulato fino a quel punto. Inoltre, tutti coloro che hanno scelto di entrare in un progetto, se ancora in gioco, verseranno 5 centesimi (dei 50 guadagnati) al leader del progetto in cui sono; i leader, invece, riceveranno 5 centesimi (aggiuntivi rispetto ai 50 guadagnati) da ogni giocatore ancora in gioco che abbia aderito al loro progetto. Se sarai eliminato, inizierai il periodo successivo fuori dai giochi;

potrai nuovamente scegliere se entrare in uno dei progetti esistenti, con la probabilità e nel ruolo previsti da quel progetto, oppure se lanciare un tuo progetto, estraendo una nuova probabilità di sopravvivenza casuale compresa tra zero e uno. Se sarai in gioco, inizierai il periodo successivo all'interno del progetto in cui sei entrato nel periodo precedente. Avrai comunque a disposizione 3 scelte: restare nel progetto in cui sei, unirti ad un altro progetto lasciando quello in cui sei, oppure avviare un progetto nuovo, estraendo una nuova probabilità di sopravvivenza. Nota che l'eliminazione è sempre individuale: vengono eliminati i singoli giocatori, NON i progetti. Un progetto resta attivo finch'e ha un giocatore al suo interno, anche quando il giocatore che per primo l'aveva iniziato dovesse essere eliminato.

Prima di iniziare i periodi dell'esperimento vero e proprio, verranno svolti 5 periodi di prova, in cui potrai impratichirti con il funzionamento dell'esperimento. I centesimi guadagnati in questi periodi non danno diritto ad alcun pagamento, e saranno quindi azzerati prima di iniziare l'esperimento vero e proprio. Lo sperimentatore ti dirà con chiarezza quando finiranno i periodi di prova e inizierà l'esperimento vero e proprio, in cui i centesimi guadagnati saranno effettivamente pagati. Alla fine del trentesimo periodo, apparirà sul tuo schermo un breve questionario. Ti ricordiamo ancora che i dati che immetterai saranno anonimi. Dopodich'e, verrà visualizzato il tuo guadagno finale, in euro. Subito dopo l'esperimento, lo sperimentatore procederà al pagamento di quanto guadagnato. Una volta che l'esperimento sarà finito, compila la ricevuta e presentati al banco dello sperimentatore con la ricevuta compilata e con il foglietto su cui è indicato il numero del tuo PC.

Domande?