THE GAME THEORY OF OPEN-SOURCE SOFTWARE

PAUL REIDY
Senior Sophister

In this paper, Paul Reidy utilises a game theoretical framework to explore the decision of a firm to make its software open-source and to anticipate how programmers are likely to respond and which open source projects are most likely to attract contributions from Programmers. Examining the motivations and decision making processes of all involved parties, he also emphasises the role of the Government in encouraging contributions to projects that are beneficial, or provide value, to society.

Introduction

Open-source software refers to software where the source code is made publicly available and modifications of the software are permitted (e.g., Mozilla Firefox). This contrasts with proprietary software such as the Microsoft Office suite where there are restrictions on how the software may be used and the source code is not made available.

Open-source software may be regarded as a public good because it is both non-rival and non- excludable. However, many open-source projects enjoy frequent contributions such as bug-fixing or patches by a large community of unpaid volunteers across the globe (Lerner, Pathak and Tirole, 2006). This suggests that the classic public goods game is not an adequate description in this case and in this paper we use game theory to present alternative models of some aspects of open-source software. In the first game we examine the decision of a private firm to release its software as open-source. In the second game we consider which open-source projects will attract contributions from programmers.

Game 1: Private Firms and Open-Source Software

Game Setup

In this game, a firm has developed new software and is considering whether to release the source-code publicly as open-source or to keep it private as proprietary software. At the initial node Nature decides whether the Firm is greedy or altruistic. Only the Firm knows the outcome and then it decides between Open-Source (OS) and Proprietary software. If it chooses Proprietary the game ends, while if it chooses OS the Programmer decides whether to Contribute (C) or Do Not Contribute (DNC) to the open-source code. If the Programmer chooses DNC the game ends. If the Programmer chooses C the Firm decides again whether to keep the code Open-Source (OS) or to take it back under its control as Proprietary. The game is represented as a Bayesian game in strategic form in Figure 1.

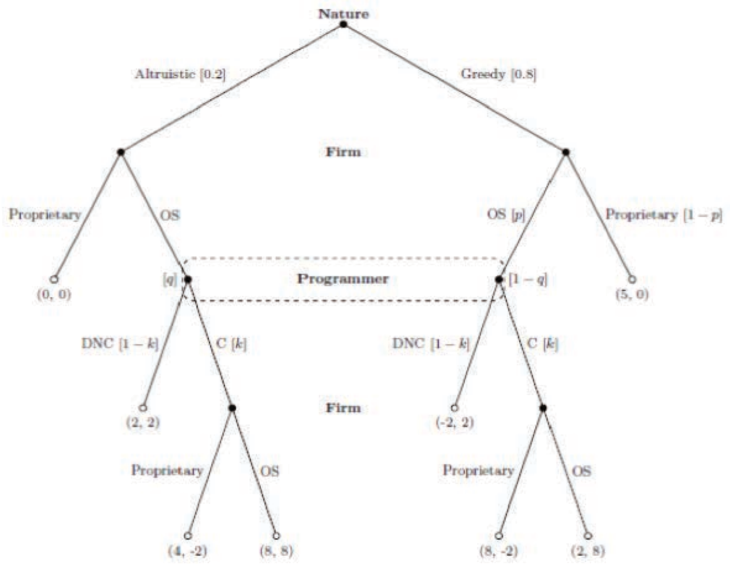

Figure 1: Private Firms and Open-Source Software

If the Firm is greedy, it wants to initially release the source code and then, following an unpaid contribution by the Programmer, make the code proprietary again as it earns higher profits from the improved code, which it can sell only if its proprietary. It thus ranks the outcomes as follows:

U(OS, C, Proprietary)>U(Proprietary)>U(OS, C, OS)>U(OS, DNC)

where U(x, y, z) denotes the utility derived from the case where the Firm chooses action x at its initial stage, the Programmer choose action y and the Firm chooses action z at the final stage. If y or z are missing the game ends before reaching the relevant stage.

The altruistic Firm does not want to exploit the unpaid contribution of the Programmer and prefers to keep the code open-source because it cares about societal welfare. It ranks the outcomes as:

U(OS, C, OS)>U(OS, C, Proprietary)>U(OS, DNC)>U(Proprietary)

Similarly, the Programmer is passionate about open-source and his most preferred outcome is that the company releases the code as open-source initially and does not renege on this decision later. He strongly dislikes his contributions to be used solely for the firm's benefit in earning profits. He therefore ranks the outcomes as:

U(OS, C, OS)>U(OS, DNC)>U(Proprietary)>U(OS, C, Proprietary)

Equilibrium

The unique semi-separating Perfect Bayes Equilibrium (PBE) is:

1. Firm's Strategy: If altruistic choose OS at the initial stage and if the Programmer chooses C choose OS. If greedy choose OS with probability p at the initial stage and if the Programmer chooses C choose Proprietary

2. Programmer: Choose C with probability k

3. Programmer's Beliefs: If the Firm chooses OS it is altruistic with probability q= 0.20.8p+0.2. If the Firm chooses Proprietary it is greedy with probability 1.

These beliefs are derived used Bayes's rule as shown in Appendix A.1.

We can show that this is a PBE by deriving suitable values of p and k and illustrating that the strategies of both players are sequentially rational. Assuming the Firm chooses OS at the initial stage, the Programmer will only be willing to randomise between C and DNC if:

2= (0.8p0.8p+0.2)(-2)+ (0.20.8p+0.2)(8)

$\therefore$ p=0.375

The Firm's strategy is sequentially rational because if the Firm is altruistic it is optimal to choose OS with probability 1 at its initial stage as the payoff is always higher than choosing Proprietary regardless of the Programmer's action. It is optimal for the altruistic Firm to choose OS at the final stage. On the other hand, if the Firm is greedy it is optimal for it to choose Proprietary at the final stage. The greedy Firm will be willing to randomise between OS and Proprietary at its initial stage only if:

$$5\underbrace{\qquad\qquad}\text{Payoff from Proprietary}=(k)(8)+(1\text{-}k)(\text{-}2)\underbrace{\qquad\qquad}\text{Payoff from OS}$$

$$\therefore k=0.7$$

Appendix A.1 rules out other PBE.

Analysis of the Game

The setup and assumptions of the game seems quite realistic because when firms release their code as open-source, volunteer programmers may be wary that the firm is trying to benefit from their unpaid contributions and will later make its source code private again. Indeed, it does seem puzzling that profit-maximising companies would make their source code publicly available. However, some companies such as Google appear to have genuinely altruistic intentions when they participate in open-source.

The predictions of the game and the semi-separating PBE are also quite interesting and realistic because they show that the possibility of a greedy Firm means that the Programmer will fear being exploited and will only contribute to the open-source project some of the time. This is disappointing for the altruistic Firm which genuinely wants to work with the Programmer to make the open-source software better and will not renege on its commitment to open-source.

However, suitable policies can help to avoid this undesirable outcome. An altruistic Firm may try to distinguish itself from the greedy Firm by making some costly investments in open-source prior to the start of the game such as donating large sums of money to open-source foundations. Alternatively, if the government wishes to ensure that programmers always contribute to open-source projects it could impose fines on greedy firms that try to choose Proprietary in the final stage. This could reduce the probability that greedy firms play the game and yield a pooling equilibrium where Programmers always choose Contribute. Specifically, as shown in Appendix A.2, if the probability of the Firm being altruistic is greater than or equal to 0.4 there will be a pooling PBE of the form:

1. Firm's strategy: If altruistic, choose OS and if the Programmer chooses C then choose OS. If greedy, choose OS and if the Programmer chooses C then choose Proprietary

2. Programmer's Strategy: Choose C if given the opportunity

3. Programmer's Beliefs: Prior Beliefs

A particularly interesting policy solution involves the use of the GNU - General Public Licence (GNU GPL). If the Firm releases its code under this licence it guarantees that any alterations to the code will be made 'freely available ... to whomever the program is distributed' (Lerner and Tirole, 2005). Thus in a game with this licence the greedy Firm could not renege on its decision to make its software open-source and always has to choose OS in the final stage as shown in Figure 2.

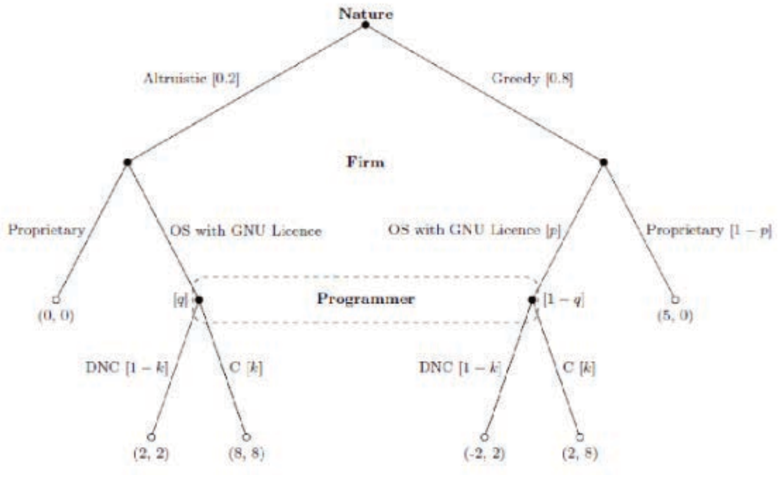

Figure 2: Game with GNU GPL Licence

The separating PBE for the game in Figure 2 is:

1. Firm's Strategy: If altruistic, choose OS with GNU Licence with probability 1. If greedy, choose Proprietary with probability 1

2. Programmer's Strategy: Always choose C if given the opportunity

3. Programmer's Beliefs: If the Firm chooses OS with GNU Licence, it is altruistic with probability 1. If the Firm chooses Proprietary, it is greedy- with probability 1

If the Firm is altruistic, it is optimal to always pick OS with the GNU Licence because it gives a higher payoff than Proprietary. If the Firm is greedy it is optimal to always pick Proprietary because it gives a higher payoff than OS with GNU Licence. The Firm's strategy is thus optimal and Programmer's beliefs are consistent. The Programmer's strategy is optimal because he will only get to play when the Firm is altruistic and always gets a higher payoff from choosing C in this case.

This is very interesting because it shows that a simple and plausible alteration to our original game can yield a separating PBE where it is optimal for the Programmer to always Contribute.

Game 2: Which Open-Source Projects Attract Contributions?

Game Setup

In this game a Founder has an idea for a new open-source project and Nature then determines whether this project will be Useful or Useless for society. Only the Founder knows the outcome of Nature's choice and he then tries to get an unpaid volunteer Programmer to work on his idea. The Founder chooses Promote or Do Not Promote for the project. Following this choice, the Programmer chooses Contribute (C) or Do Not Contribute (DNC).

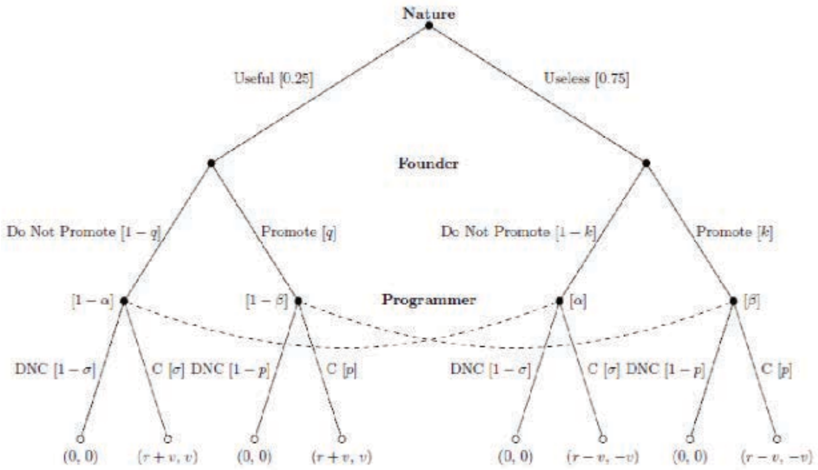

Figure 3:Which Open-Source Projects Attract Contributions?

The Founder is quite egotistical and he derives a private benefit of $r > 0$ if the Programmer contributes to his project, regardless of whether it is Useful or Useless. This is because he likes to be popular and to attract attention. If the project is Useful then both the Programmer and the Founder get a payoff of $v > 0$ which is the value created for society. If the project is Useless they both incur a cost of $-v$ which is the opportunity cost of not being involved in projects which actually created value for society. The game and subsequent is based on the 'Defensive Medicine' in Harrington (2009) and is shown in sequential form in Figure 3 above.

Equilibria

For equilibirum 1, a pooling PBE of this game is:

1. Founder's Strategy: Promote the project whether it is Useful or Useless

2. Programmer's Strategy: Ignore the Founder's message and always choose DNC

3. Programmer's Beliefs: Project is Useful with probability 0.25 and Useless with probability 0.75

The Programmer's beliefs are consistent because they are equal to the prior beliefs as the Founder's action is uninformative. The Programmer's strategy is optimal because the expected payoff for DNC is greater than that for C:

$$\underbrace{0}_{\text{Payoff of DNC}} > \underbrace{(0.25)(v)+(0.75)(-v)}_{\text{Payoff of C}}$$

$$0 > -0.5v \qquad \text{(because } v>0\text{)}$$

The Founder's strategy is also optimal because his payoff is 0 whether he chooses Promote or Do Not Promote because the Programmer always chooses DNC.

For equilibrium 2, a separating PBE is given by:

1. Founder's Strategy: Promote the project if it is Useful. Do Not Promote if it is Useless.

2. Programmer's Strategy: Contribute if the Founder promotes the project. Do Not Contribute if the Founder does not promote the project

3. Programmer's Beliefs: If the Founder promotes the project its Useful with probability. If the Founder does not promote the project it is Useless with probability 1

The Programmer's beliefs are consistent because only a Useful project is promoted by the Founder and a Useless project is never promoted. The Programmer's strategy is optimal because he only Contributes when it is Useful. The Founder's strategy for a Useful project is optimal because if he Promotes he gets a payoff of r + v (as the Programmer will Contribute) which is strictly greater than 0, the payoff from choosing Do Not Promote. If the project is Useless, the Founder's strategy of choosing Do Not Promote is optimal as long as:

$$0 \overbrace{\qquad}^{} \text{Payoff of Do Not Promote} > r-v \overbrace{\qquad}^{} \text{Payoff of Promote}$$

$$v \geq r$$

For equilibrium 3, a final semi-separating PBE is given by:

1. Founder's Strategy: Promote the project with probability 1 if it is Useful. Promote the project with probability k if it is Useless

2. Programmer's Strategy: Contribute with probability p if the project is promoted. Do Not Contribute if the project is not promoted

3. Programmer's Beliefs: Using Bayes' Rule the Prob(Useless | Promote)= $\beta = 0.75k0.25 + .75k$ and the Prob(Useless | Do Not Promote)= $\propto = 1$

This equilibrium is not very interesting and will only occur in the special case where r = v and k = 1/3 as shown in Appendix B. In our analysis we will focus on Equilibrium 1 and 2.

Analysis of the Game

The assumptions and setup of the game seem reasonably realistic because it is often very difficult for programmers to predict which open-source projects are actually useful ex-ante. However, it is probably a bit unrealistic to assume that the Founder knows whether the project is useful if the Programmer does not. It might also seem unrealistic that the Founder could be motivated by egotistical concerns. However, as the open-source advocate Eric Raymond (1999) commented 'The 'utility function' [of open-source programmers] is not classically economic, but is the intangible of their own ego satisfaction and reputation.'

The predictions of the game are quite interesting. In Equilibrium 1, the Founder always promotes the project and the Programmer always ignores this uninformative message and chooses not to contribute. This is undesirable for society because the Programmer never contributes even though in some cases it is a Useful project.

Equilibrium 2 is desirable for society because the Founder only promotes the project if it is Useful and the Programmer only contributes if it is promoted (and thus Useful). However, we showed above that this equilibrium can only exist if $v \geq r$. The intuition is that when $r = 0$ the Founder has no ego and has the exact same payoffs as the Programmer. However, when $r > 0$ the interests of the two players start to diverge and as r increases the Founder gets relatively more and more concerned about this own interests and less concerned about, v, the value for society. As long as $v \geq r$ the Founder is not 'too' ego-driven and still cares about the value created by the open-source project and the separating PBE can exist. However, when $v<r$ the ego of the Founder is too large and this separating PBE cannot exist.

A policy implication of this is that to ensure the desirable separating PBE exists the government could try to boost v, the value created for society. If v depends on the number of users of the software, for example, the government may provide subsidies encouraging people to switch to the open-source software. Alternatively, the open-source community itself may discourage ego-driven behaviour to lower the value of r. We could also extend the game by adding more stages such as a stage where the Programmer can punish the Founder if he promoted a Useless project to boost his own ego.

References

Harrington, J.E. Jr. 2009. Games, Strategies, and Decision Making. New York: Worth Publishers.

Lerner, J. and Tirole, J. 2005. 'The Economics of Technology Sharing: Open-Source and Beyond'. Journal of Economic Perspectives, 19:2:99-120.

Lerner, J., Pathak, P.A. and Tirole, J. 2006. 'The Dynamics of Open-Source Contributors.' American Economic Review, 96:2:114-18.

Raymond, E.S. 1999. The Cathedral and the Bazaar: Musings on Linux and Open Source by An Accidental Revolutionary. California: O'Reilly Media.

Appendix A

A.1 Game 1 Beliefs and Ruling Out Other Equilibria

Let q equal the Programmer's belief that the Firm is Altruistic given that is has chosen OS. Then we derive the Programmer's expected utilities of choosing C and DNC given q:

$$EU\_p\ (C \mid q)=8(q)+(-2)(1-q)=10q-2$$

$$EU\_p\ (DNC \mid 1-q)=2$$

Therefore:

If $q > 0.4$ the Programmer will choose Contribute

If $q < 0.4$ the Programmer will choose Do Not Contribute

If $q = 0.4$ the Programmer is indifferent between Contribute and Do Not Contribute

The Programmer will prefer to choose Contribute if:

$$10q-2>2$$

$$q>0.4$$

If the Programmer observes OS what should it believe about the type of the firm? Using Bayes' rule:

Prob(Greedy | OS)= (Prob(Greedy)Prob(OS | Greedy))/(Prob(Greedy)Prob(OS |Greedy)+Prob(Altruistic)Prob(OS | Altruistic))

$= 0.8p/(0.8p+0.2)$

Therefore $1-q=08p/(0.8p+0.2)$ and $q=0.2/(0.8p+0.2)$

If the Programmer observes Proprietary what should it believe about the type of firm?

Using Bayes's rule:

$$\text{Prob(Greedy | Prop)} = (\text{Prob(Greedy)Prob(Prop | Greedy)})/(\text{Prob(Greedy)Prob(Prop | Greedy)}+\text{Prob(Altruistic)Prob(Prop | Altruistic)})$$

$= 0.8(1\text{-}p)/(0.8(1\text{-}p)+(0.2)(0))=1$

Case 1: q > 0.4

When $q > 0.4$ we can derive the value of p from:

$$q>0.4$$

$$0.2/(0.2+0.8p)>0.4$$

$$0.12>0.32p$$

$$p<0.375$$

Since $q > 0.4$ the Programmer always plays Contribute. But if the Programmer always plays Contribute then the greedy Firm will to always play OS at the initial stage. Thus $p = 1$ which violates the condition $p < 0.375$ and this cannot be an equilibrium.

Case 2: q < 0.4

When $q < 0.4$ then $p > 0.375$. Since $q < 0.4$ the Programmer always plays Do Not Contribute. But if the Programmer always plays Do Not Contribute the greedy Firm always wants to play Proprietary at the initial stage. Thus $p=0$ which violates the condition $p > 0.375$ and this cannot be an equilibrium either.

Case 3: q=0.4

When $q=0.4$ then $p=0.375$. Since $q=0.4$ the Programmer is indifferent between Contribute and Do Not Contribute. If $p=0.375$ the greedy firm is mixing between Proprietary and OS. The greedy Firm will only do this if it is indifferent between which can only be the case if the Programmer is mixing. If the Programmer chooses Contribute with probability k and Do Not Contribute with probability $1 – k$ then the greedy Firm will mix if:

$$\underbrace{5}_{\text{(Payoff from Proprietary)}} = \underbrace{(k)(8)+(1-k)(-2)}_{\text{(Payoff from OS)}}$$

$$k=0.7$$

A.2 Pooling PBE for Game 1

Consider a pooling PBE for the game in Figure 1:

> **1.** Firm’s strategy: If altruistic then choose OS and if the Programmer chooses C then choose OS. If greedy choose OS and if the Programmer chooses C then choose Proprietary
>
> **2.** Programmer’s Strategy: Choose C if given the opportunity
>
> **3.** Programmer’s Beliefs: Prior Beliefs

The Firm's strategy if it is altruistic is optimal because it always prefers to choose OS no matter what the Programmer does. The Firm's strategy if it is greedy is also optimal because the Programmer always chooses C. The Programmer's beliefs are consistent because the action of the Firm provides no information so prior beliefs are used. Letting γ denote the probability of an altruistic firm, the Programmer’s strategy is only optimal if:

$$2 \leq (\gamma)(8)+(1-\gamma)(-2)$$

$$\gamma \geq 0.4$$

Thus we will only have a pooling PBE where the Programmer always chooses C if the probability of the firm being altruistic is greater than or equal to 0.4.

Appendix B

Consider the semi-separating equilibrium in Game 2:

1. Founder's Strategy: Promote the project with probability 1 if it is Useful. Promote the project with probability k if it is Useless

2. Programmer's Strategy: Contribute with probability p if the project is promoted. Do Not Contribute if the project is not promoted

3. Programmer's Beliefs: Using Bayes' Rule the Prob(Useless | Promote)= $\beta = 0.75k/(0.25+0.75k)$ and the Prob(Useless | Do Not Promote)= $\propto$ =1

If the project is Useful the Founder's strategy will be optimal as long as:

$$p(r+v)+(1-p)(0)\geq 0$$

$$p(r+v)\geq 0$$

This is true because $r > 0$ and $v > 0$.

If the project is Useless the Founder's strategy dictates that he randomises between promoting and not promoting and the Founder will only be willing to do this when:

$$0=(0)(1-p)+(r-v)(p)$$

$$0=(r-v)p$$

This will only hold when $r = v$.

If the project is promoted the Programmer will only be happy to randomise between contributing and not contributing when:

$$0.25/(0.25+0.75k)\ (v)+\ 0.75k/(0.25+0.75k)\ (-v)=0$$

$$k=\ 1/3$$

If the project is not promoted then the Programmer will not contribute which is optimal because:

$$0 \geq \text{Prob}(\text{Useful} \mid \text{Do Not Promote})(v)+\text{Prob}(\text{Useless} \mid \text{Do Not Promote})(-v)$$

$$0 \geq (0)(v)+(1)(-v)$$

$$0 \geq -v$$