Population Dynamics in Open Source Communities: An Ecological Approach Applied to Github

Pablo Loyola
University of Chile/
University of Tokyo

In-Young Ko
Korea Advanced Institute
of Science and Technology

6th International Workshop on Web Intelligence & Communities
April 7, 2014 in Seoul, Korea

Introduction

- Paradigm shift in Software development:
  - Individual to Social and Distributed
  - Web as the main interface for collaborative work
  - Open Source Communities
    - Transparency
    - Visibility
    - Social layer

Introduction

- In OSS, the group of contributors is the key aspect to estimate quality
  - Critical mass to fix bugs and add new features
  - More people, more eyes (Linus law)
- Without contributors, the project dies
  - Inactivity
  - Lack of interest

Introduction

- We are interested in estimating the population of contributors over time
  - To have a notion of the quality of the product
  - Increasing use of OSS in both public and private sector
- But it is challenging...
  - How to handle motivations, incentives, social and technical issues, roles?

Introduction

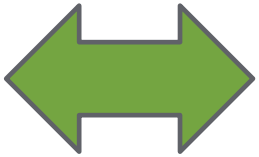
- We hypothesize that OSS development is similar to a mutualistic configuration present in nature :
  - Symbiotic relationship between a host and a parasite

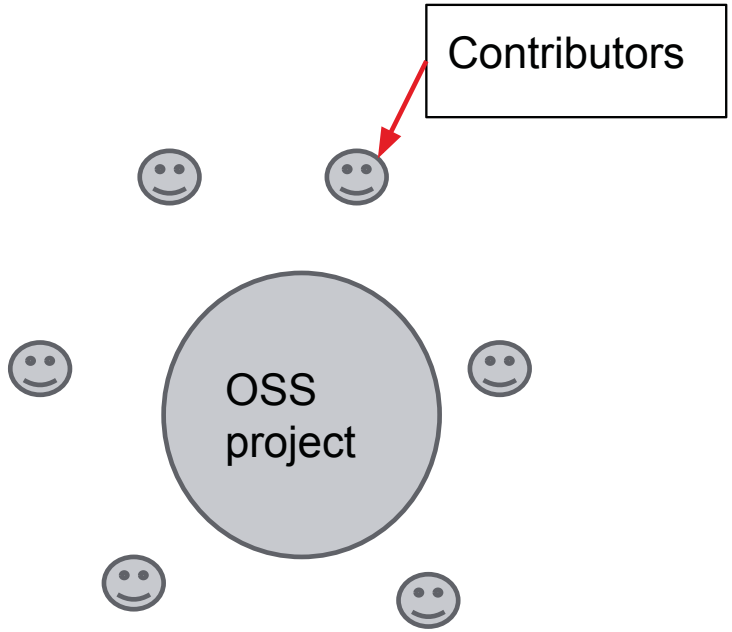
Introduction

Oxpecker

Impala

The oxpecker (parasite) eats the ticks from the impala (host). This prevents diseases for the impala





Contributors (parasites) improve the OSS project and generate a better quality software.

In turn, the OSS project (host) benefits the developers as they can access the code for their own purposes and also gain reputation and knowledge

How to support our idea?

Main characteristics of mutualistic behavior in nature:

1. Mutual benefit between species
2. High degree of specialization
3. Co-evolution between species
4. Constant interaction leads to stable relationship
5. A parasite may evolve to become less harmful to its host
6. Transference of genetic material between species

Mutual benefit between species

- OSS development follows a private-collective innovation model (von Hippel, 2003)
  - Contributors spend their own time and resources to a common project
  - Receive benefits in terms of knowledge and expertise that they can use for their private work

High degree of specialization

- In OSS development contributors arrive with a heterogeneous level of expertise
  - Low code understanding / Lack of synchronization
- Therefore, developers need to learn from the code, sometimes focusing on a small number of modules (ownership)

Co-evolution between species

- Developers begin to contribute and interact
  - Each contributor improves his knowledge, based on continuous feedback on his commits
- OSS project receives contributions which change its structure and behavior.
  - The performance of the resulting software is tested by the community (community feedback allows symbiotic cycle)

Constant interaction leads to stable relationship

- Initially, OSS projects perform in an erratic way
  - High heterogeneity due to lack of a road map (David2008).
- As time passes, some initial collaborators abandon the project
  - Convergence to organizational equilibrium.
  - This leads to a smaller but more cohesive team (Bird2008).

A parasite may evolve to become less harmful to its host

- Learning process while contributing to a project.
  - The rich interaction with a diversity of contributors
  - Specialization in determined parts of the code,
  - Developer gain adaptiveness which eventually will reduce the number of bugs (Hemetsberger2006).

Transference of genetic material between species

- Social development platforms like Github allow a developer to contribute to several projects simultaneously.
  - Rich source of knowledge
  - Expertise can be reused in several other projects that share the same requirements (Kogut2001).

How to model mutualism?

Lotka-Volterra equations

- Initially developed to model competition between species (May, 1982)
  - Predator - Prey
- Further modifications allowed modelling of mutualistic interactions (Bascompte, 2006)
  - Host - Parasite

How to model mutualism?

n plants and m animal species

P_i represents the population of the plant i
A_j represents the population of animal j

$$\frac{dP_i}{dt} = r_i P_i - S_i P_i^2 + \sum_{j=1}^{m} \alpha_{ij} P_i A_j$$

$$\frac{dA_j}{dt} = q_j A_j - T_j A_j^2 + \sum_{i=1}^{n} \beta_{ji} P_i A_j$$

Table 1: Lokta-Volterra parameters

Parameter	Meaning
r_i	Growth rate of plant i
S_i	Intraspecific competition of plant i
q_j	Growth rate of animal j
T_j	Intraspecific competition of animal j
α_{ij}	Per-capita effect of animal j on plant i
β_{ji}	Per-capita effect of plant i on animal j

How to adapt the model to an OSS scenario?

- Growth rate
  - Simply extract the number of projects generated and contributors per time unit
- Intraspecific competition:
  - Contributor intraspecific competition:
    - Reputation-based metric: Contributors look for recognition from their peers.
    - Number of followers obtained in a certain period of time

How to adapt the model to an OSS scenario?

- Intraspecific competition:
  - OSS project intraspecific competition:
    - Each project “fights” to catch contributors to work on it
    - Metric : Number of contributors per project

How to adapt the model to an OSS scenario?

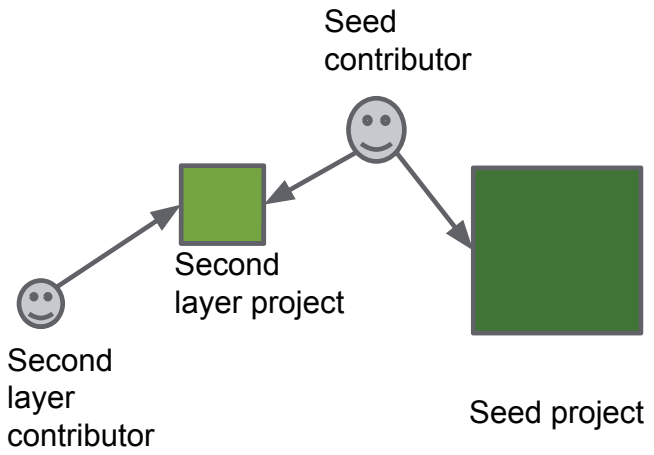
- Per-capita effect
  - Contributor to Project:
    - Number of contributions submitted to the project

      (we assume this is the main way to influence the behaviour of the project).
  - Project to Contributor:
    - Number of followers that a given contributor obtained by participating in a given project.
    - Assumed correlation between followers and

Evaluation

Extract samples (sub-environments) from the Github ecosystem:

1. Seed project extraction
2. Seed contributor extraction
3. Second layer project extraction
4. Second layer contributor extraction



Evaluation

- Data extracted using Github API during January to June, 2013
- It comprised commit and social activity, which were grouped on a daily basis
- Evaluation: Statistical comparison between contributor population calculated with the model and the real contributor population obtained from the data

ID	Seed Repository	ID	Seed Repository
1	bootstrap	14	CodeIgniter
2	homebrew	15	linux
3	rails	16	phonegap-plugins
4	html5-boilerplate	17	Diaspora
5	hw3rottenpotatos	18	three.js
6	oh-my-zh	19	zf2
7	node	20	jquery-mobile
8	jquery	21	django
9	phonegap-start	22	jekyll
10	impress.js	23	less.js
11	backbone	24	devise
12	d3	25	textmate
13	jquery-ui	26	redis

Table 1: Sub-environments with seed repository

Evaluation

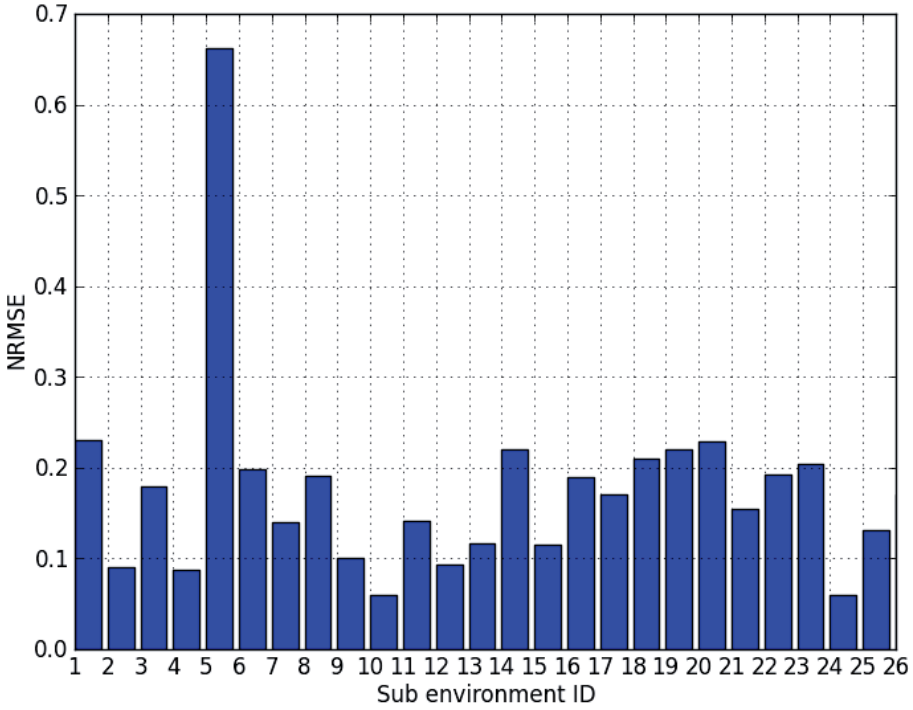
Threats to validity:

- External: We only used Github as OSS ecosystem (SourceForge and Google Code are valid platforms)

- Internal: Time limit for data extraction using the Github API could lead to errors

- Construct: It is possible that this adaptation can be achieved in a different way. However, social component represent an intuitive attempt

Results



$$NRMSE = \frac{RMSE}{\Delta Y}$$

ΔY represents the range of observed values of the variable under study

On average, the mutualistic behavior is present on the overall subset of the Github sub-environment

For low NRMSE sub-environments, the seed repository is one of the commonly called successful OSS projects, such as Rails or Bootstrap

For sub environment 5, the model failed to show convergence

Results

ID	Seed Language	No of Projects	NRMSE
1	Javascript	67	0.11
2	PHP	51	0.31
3	Python	56	0.179
4	Ruby	33	0.101
5	Shell scripting	11	0.521

Table 1: Normalized Root Mean squared error for each language-based sub-environment

- Selecting a seed repository that represents a specific programming language
- The condition for mutualism, a low value of NRMSE, is more accentuated
- Filtering by language, increases the level of specialization scores and subsequently reduces the number of failed contributions
- Ruby and Javascript are languages widely used and the Github environment

Conclusions

- Mutualism presents an initial opportunity for estimating quantitatively the evolution of the population over time
- It requires a better evaluation
  - Better generalization
  - How to merge “sub-environments?”

Future Work

- Explore the idea of real time “observatory”
  - Provide a “health” score for each project
  - Try answer the question: Is it safe to use this software ?
- Incorporate network elements to analyze the internal behaviour of the community over time

Thank you