UNIVERSITY OF NOTRE DAME

Public Goods Theory of the Open Source Development Community using Agent-based Simulation

Scott Christley, Jin Xu, Yongqin Gao, Greg Madey

Dept. of Computer Science and Engineering

University of Notre Dame

Supported in part by National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829

Overview

- Public Goods Theory
- Open Source Software Community
- Conceptual Model
  - Characteristics of Open Source Project
  - Characteristics of Individuals
  - Characteristics of the Group
  - Open Source Action Processes
  - Characteristics of Environment
  - Critical Mass/Project Success
- Agent-based Model
- Challenges

Public Goods Theory

- Collective action out of mutual self-interest (Samuelson 1954, Marwell & Oliver 1993)
- Jointness of supply, multiple individuals can use the public good without impinging upon each others use.
- Impossibility of exclusion, no single individual can prevent use of public good by others.
- Free-rider phenomena, individuals can use the public good without contributing.
- Connectivity (Fulk et al. 1996), ability of members to communicate with other members.
- Communality, the shared body of information and knowledge held by the members.

Open Source Software (OSS)

GNU

Linux

- **Free …**
  - to view source
  - to modify
  - to share
  - of cost
- **Examples**
  - Apache
  - Perl
  - GNU
  - Linux
  - Sendmail
  - Python
  - KDE
  - GNOME
  - Mozilla
  - Thousands more

Savannah

The Apache Software Foundation

http://www.apache.org/

Op

Open Source Community

- New, fascinating phenomena
- Motivation of individuals and organizations
- Group dynamics
- Software engineering practices, is this a new way to develop software?
- Factors for project success
- Time series analysis, dynamic networks

OSS Project Characteristics

- Has the properties of collective action, jointness of supply, impossibility of exclusion, and free-rider phenomena.
- Connectivity is provided by email and discussion forums.
- Communality encompasses web pages, documentation, FAQs, wiki's, discussion archives, etc. beyond just the software.
- Complex, socially-constructed, informational good provides heterogeneous benefits to its users and has heterogeneous resource requirements. Bessen (2001) puts forward the notion of complex good for why OSS is provided when commercial software is not.

Characteristics of Individuals

- Heterogeneous interests, resources, benefits
- Fungibility? Maybe not due to resource(skill/time) interdependencies inherent in software development process and software architecture.
- Individual categories; project leaders, core developers, co-developers, active users, and passive users.

Characteristics of Group

- People associated with the project.
- Full communication connectivity through email lists and discussion forums.
- Group norms, heterogeneous
  - Acceptable usage of discussion forums
  - Source code standards
  - Ownership and maintenance responsibility for developed code. Individuals self-selectively acquire new responsibilities.
- Group cognition for purpose of project, dynamic, acts as constraint upon individuals yet individuals also act to shift group cognition goals closer to their own.

OSS Action Processes

- Action undertaken by an individual that translates into collective action.
- Individual actions
  - Write source code and documentation
  - Post messages
  - Report bugs
  - Request features
- Project actions performed by individual on behalf of project.
  - Add/remove developers
  - Commit source code
  - Release file distributions
- Contribution may actually reduce some individual benefits.

Characteristics of Environment

- Environment is not generally considered in public goods theory but plays important role for OSS.
  - Proprietary software companies
  - Political and economic climate
  - Legal and intellectual property rights
  - Technology advances, standardization efforts
- Environment is not static and OSS projects can quickly become marginalized or pushed to the forefront.

Critical Mass/Project Success

- Critical mass is point when public good has received enough resources to be realized.
- Generally a monotonic decision function but not true for OSS.
  - Definition of success is not homogeneous among projects
  - Projects that have achieved a critical mass may lose it later for many different reasons.
- Critical mass for OSS is an ongoing adaptive process.

Agent-based Simulation

- *Insert ontology/CommunicativeModel*
- Individuals and projects modeled as agents
- Social network
- 80,000 projects; 160,000 individuals which is much larger if free-riders are counted.
- Focus on time evolution and dynamics

Prior Work

- Topological analysis (Xu)
  - Power law relationship for projects and developers
  - Scale-free and small world network
- Data Mining (Gao)
  - Five temporal factors are significant for clustering projects into success categories.
  - # of developers, # of file releases, # of help requests, # of tasks opened, # of tasks closed
  - Can categorize failed projects accurately but excellent projects are outliers, few data points

Challenges

- Sourceforge data dump, snapshot of community at single point in time.
- Successful projects are rare! Data farming to produce synthetic data points.
- Many input distributions to be modeled
  - Individual attributes like interest, skill, and time are not directly available but need to be inferred.
  - Rate of action processes
- Dynamic network analysis, can emergence be characterized by local structural changes.

Validation Techniques

- Statistically similar global network properties, but numerous processes can produce such global metrics so this isn't completely satisfying.
- Data mining produces the same significant factors and clustering, but we still have issue of incomplete information.
- Hypothesis generation that can be tested with empirical surveys, interviews, and other data sources (GNU, GNA, Apache, Mozilla).

Stay Tuned!