INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 14, NO. 1, MARCH 2009, 77-86

Consumption and Production of Digital Public Goods

Modeling the Impact of Different Success Metrics in Open Source Software Development

Nicholas P. RADTKE[1,3] and Marco A. JANSSEN[2,3]

***Abstract*—With the Internet has come the phenomenon of people volunteering to work on digital public goods such as open source software and online encyclopedia articles. Presumably, the success of individual public goods has an effect on attracting volunteers. However, the definition of success is ill-defined. This paper explores the impact of different success metrics on a simple public goods model. The findings show that the different success metrics considered do have an impact on the behavior of the model, with the largest differences being between consumer-oriented and producer-oriented metrics. This indicates that many proposed success metrics may be mapped into one of these two categories and within a category, all success metrics measure the same phenomenon. We argue that the characteristics of producer-oriented metrics more closely match real world phenomena, indicating that public goods are driven by producer, and not consumer, interests.**

***Index Terms*—Digital public goods, success metrics, FLOSS, open source software, Wikipedia.**

1. Introduction

In recent years an interesting phenomenon can be observed in the digital media. People are volunteering their time to contribute, for example, to the creation of software [1] or an online encyclopedia [2] at their own costs for the benefit of the wider population. Such products can be called public goods. Traditional non-cooperative game theory argues that people will not invest in public goods since it benefits to free ride on the investments of others.

Research in psychology, economics, and political science shows that people invest in public goods, as observed in case studies and replicated in controlled experiments with human subjects [3]. A variety of factors are put forward as possible explanations for these contributions, such as other-regarding preferences, communication, etc.

Another finding is that the level of cooperation reduces as group size increases [4]. Therefore, it is remarkable to see high levels of contributions to digital public goods like open source software and Wikipedia, where the number of people involved is huge. Looking into more detail of the statistics, it can be seen that the distribution of the public goods products which are successful is skewed, as is as the distribution of producers working on public goods. For example, only 14% of the projects at SourceForge, the largest site hosting open source projects, have been updated during the last year.[4]

In an earlier paper [5] we presented an empirically grounded model that captures several main patterns of the SourceForge repository of open source data. One of the assumptions is that the success of a project affects the attractiveness of a project. However, a key problem in the literature of open source software is the ambiguity of the definition of success for a project. Can it be measured by the number of downloads, the frequency of releases, the number of bug fixes, or any number of other indicators?

This paper presents a more theoretical and simpler model than [5] of the evolution of populations of digital public goods; the model is used to test the consequences of different definitions of success.

First, basic empirical findings from studies on open source software and Wikipedia will be presented in Section 2. In Section 3 the model will be presented, and the analysis of the model is contained in Section 4.

2. Empirical patterns

With Web 2.0 people can contribute and consume goods which are freely available to others. What makes some of these products successful and others not? How is success defined?

2.1. SourceForge

SourceForge is a site that contains a set of online tools facilitating the development of open source software. It was established in November, 1999 and as of November, 2008 hosts 138,674 projects. Using data from this site, the distribution of developers working on a project has been shown to be highly skewed, with 67% of projects having only one developer and 90% of projects having fewer than 4 developers [6]. It has been found that 10% of the developers write 72.3% of the code [7] and the 100 most active developers are involved in 1,886 different projects [8]. Also, the number of people reporting bugs is an order of magnitude greater than the number of developers fixing bugs, which is an order of magnitude greater than the number of core developers for a project [7].

[1]School of Computing and Informatics, Ira A. Fulton School of Engineering, Arizona State University, P.O. Box 878809, Tempe, AZ 85287-8809, U.S.A.

[2]School of Human Evolution and Social Change, Arizona State University, P.O. Box 872402, Tempe, AZ 85287-2402, U.S.A.

[3]Center for the Study of Institutional Diversity, Arizona State University, P.O. Box 872402, Tempe, AZ 85287-2402, U.S.A.

[4]http://www.SourceForge.net year period defined as 11/28/07-11/27/08.

Various extensive surveys have been performed where open source developers are asked why they participate in open source software development (e.g., [9], [10]). The main reasons given are:

- Develop new skills
- Share knowledge with other developers
- Improve existing open source projects
- Engage in a new form of cooperation
- Enjoy the challenge
- Improve job opportunities

2.2. *Wikipedia*

The online encyclopedia Wikipedia started in January, 2001 and now[5] consists of of 11.8 million articles in 264 languages from 597 million edits by 14.6 million users. The distribution of contributions is skewed with 90% of the users contributing fewer than 10% of the edits [11]. The number of people per article and the number of edits per article follows a power law distribution [12].

[2] distinguishes factors that motivate people to participate, namely reputation and commitment to group identity, in Wikipedia. Registered users are assumed to be motivated more than anonymous users to contribute and to have higher quality contributions. [2] shows that anonymous users provide infrequent contributions, but the contributions are of high quality.

In summary, both digital public goods examples show that there is a skewed distribution of contributions.

2.3. *Success Metrics*

In order to understand why the contributions to SourceForge and Wikipedia are so skewed, it is necessary to understand why people contribute or use certain digital public goods. A possible explanation is that contributors prefer to participate in successful projects. However, there is no agreement how to measure the success of digital public goods. For example, success of open source software is not clearly defined. While there are no generally agreed upon standards, the following success indicators have been proposed:

- Completion of the project [13]
- Progression through maturity stages [14]
- Number of developers
- Level of activity (i.e., bug fixes, new feature implementations, mailing list)
- Time between releases
- Project outdegree [15]
- Active developer count change trends [15]

Furthermore, [16] asked eight developers how they defined success and failure of an open source project. Answers varied for success, but all agreed that a project with a lack of users was a failure. Thus having a sufficient user-base may be another metric for success.

For Wikipedia articles, success may relate to the quality of the articles. Quality of the articles is suggested to be related to the number of edits and unique editors to an article [17]. Usual manual evaluations of articles, factual accuracy [18], and credibility [19] are mentioned. Statistics of consumers and consumer experiences with articles would be helpful, but information on how many times articles have been read is not available.

In summary, there is no clear method to define success of digital public goods, especially for the purpose of modeling the development of these goods. Therefore we will use different definitions of success in our model that reflect the accumulation of activities and the number of users involved and explore whether different definitions of success have an impact on the patterns generated by the model.

3. Model Description

The model we present is a very simplistic model of consumers and producers of an ecology of public goods based on the observed processes of open source software development. Given are N_a agents and N_p projects. At each time step an agent may 1) contribute to the development of a project and/or 2) consume (a.k.a. use) a project. Each agent has a probability p_c to contribute to a project and a probability p_u to consume from a project. The probabilities p_c and p_u are drawn from an exponential distribution with parameter value 10. This represents the notion that most agents will have a small probability to be active during a time step. If a value higher than 1 is drawn, this result is ignored and a new value is drawn.

When an agent is active during a time step it will make a decision about which project to contribute to or use based on how close the characteristics of a project match with the preferences of the agent. To define how well agent preferences match characteristics of a project, the matching interests value M_i is calculated for each project, as shown in Equation (1):

$$M_\mathrm{i} = 1 - (n_\mathrm{p} - n_\mathrm{a})^2 \tag{1}$$

where n_p is the characteristics (a.k.a. needs) of the project and n_a the preference value for the agent.[6] If both dimensions match, M_i is equal to 1. Initially values for n_p are assigned randomly from a uniform distribution. However, it is assumed that consuming agents may have certain needs (e.g., an interest in well-documented, easy-to-use projects) while producing agents will have a different set of needs (e.g., an interest in projects for the challenge and to gain experience) [9], [10]. Thus values for n_a are based on an agent's producer number p_c and consumer number p_u. A simple function for mapping from p_c and p_u to n_a is shown in Equation (2):

$$n_\mathrm{a} = f(p_\mathrm{u}, p_\mathrm{c}) = \frac{p_\mathrm{c} - p_\mathrm{u} + 1}{2} \tag{2}$$

A visual depiction of the mapping is shown in Figure 1. Essentially, an agent's needs can be thought of as a continuum between 0.0 and 1.0. Lower values (< 0.5) represent a bias

[5]http://meta.wikimedia.org/wiki/List_of_Wikipedias accessed November 24, 2008.

[6]To keep consistent with our earlier publication [5] covering a more complex model, we refer to n_p and n_a as needs vectors. In this simplified model, these are one-dimensional vectors and can thus be treated as scalars.

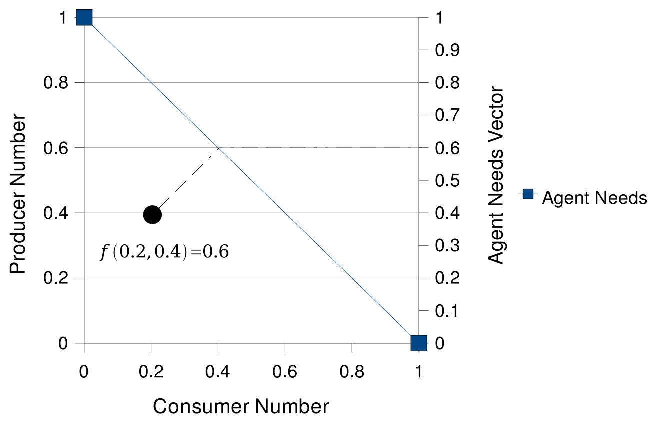


Fig. 1. Example of mapping an agent's producer and consumer numbers to its needs vector. For a given point (x, y), the nearest point on the Agent Needs line is found and then projected onto the right y-axis.

towards consuming and higher values (> 0.5) a bias towards producing. For a given point (x, y), mapping is done by finding the nearest point on the Agent Needs line and then projecting this onto the second y-axis. Thus a full-strength producing agent has a needs vector of 1 ($f(0, 1) = 1$), a full strength consuming agent a needs vector of 0 ($f(1, 0) = 0$), and an equal-part producing, equal-part consuming agent a needs vector of 0.5 ($f(x, x) = 0.5$).

The utility U_u of a project to a consuming agent is proportional to how well the agent's and project's interests match, as shown in Equation (3):

$$U_\text{u} = M_\text{i} \tag{3}$$

Agents who decide to contribute to a project are assumed to take into account both how well it matches the agent's preferences (M_i) and how successful the project is (S). Agents differ in their weights, α, for the two different indicators. Currently, α is drawn from a normal distribution with a mean of $1/2$ and standard deviation of $1/6$. The utility of a project for contributing agents is thus initially defined as

$$U'_\text{c} = \alpha \cdot M_\text{i} + (1 - \alpha) \cdot S \tag{4}$$

where M_i, and S are scaled between 0 and 1. Since there is no agreement on how to measure success S, we explore the consequences of different formulations of success:

- S_1: current number of consumers
- S_2: cumulative number of consumptions
- S_3: current number of producers
- S_4: cumulative number of contributions (a.k.a. work)
- S_5: each project equally successful

S_1 and S_3 represent the current popularity, or recent popularity, of a project with consumers and producers respectively. S_2 and S_4 represent long-term popularity with consumers and producers respectively. S_5 is used as a baseline for comparison purposes. Note that each success metric is normalized. For example, for a given project P, S_1 is the current number of consumers working on P divided by the total number of agents currently consuming any existing projects.

Finally, a switching costs sc is subtracted from the utility of production if the project is different than the last project the agent contributed to. This reflects the transaction costs in switching projects, which may include learning new customs and code. The final utility for contributing agents is defined in Equation (5):

$$U_\text{c} = \alpha \cdot M_\text{i} + (1 - \alpha) \cdot S_i - sc \tag{5}$$

If an agent makes a decision to consume or produce, it calculates the expected utility of all available projects and chooses the project with the highest utility. When a project is not updated (no agent contributes to it) for a number of time steps (using a default of 5 time steps), it is considered a dead or inactive project and is removed from the system. Agents who produce can start a new project with a probability $p_\text{s} \cdot p_\text{c}$, where p_s is a model-level constant controlling the probability of creating a new project. Thus agents who have a higher tendency p_c to contribute to projects are also more eager to start new projects. In the default case p_s is equal to 0.01.

Agents only contribute to a project when they expect this will lead to a utility greater than or equal to a minimum utility U_min. Agents stay with a project they last worked on if they choose to participate and there is no better alternative. In the default case U_min is equal to 0.2.

Agents who contribute to a project will affect the characteristics of the project n_p. The new value of n_p at time t ($n_{\text{p},t}$) is adjusted by the average values of the preferences of the contributing agents' n_a values and the previous value of n_p ($n_{\text{p},t-1}$). The adjustment rate between a project's old characteristics $n_{\text{p},t-1}$ and the contributing agents' preferences n_a is determined by λ, which is equal to 0.85 in the default case, as shown in Equation (6):

$$n_{\text{p},t} = \lambda \cdot n_{\text{p},t-1} + \frac{1 - \lambda}{totcont} \cdot \sum_{i=1}^{totcont} n_{\text{a},i} \tag{6}$$

where $totcont$ is the current number of agents contributing to a project and $n_{\text{a},i}$ is the preference of the i$^\text{th}$ agent contributing to the project.

Agents may not reconsume the same project within a certain time span, modeling the assumption that consuming an unchanged or minorly changed project will not be worth the effort (e.g., there is effort involved in downloading and installing software or reading a Wikipedia article). The time span limiting reconsumption is set to 10 for the model runs considered.

The model is implemented in NetLogo; multiple runs were executed in parallel on a high performance computing system.

4. Model Analysis

To study the model, two input parameters are varied and several resulting distributions analyzed. The input variables considered are the definition of success and the switching cost. The five possible success metrics were outlined as S_1–S_5 in Section 3. The switching cost, sc, is a penalty subtracted from the utility of projects the agent did not work on during the last time step the agent contributed to a project, as described in Equation (5). This represents the extra effort an agent

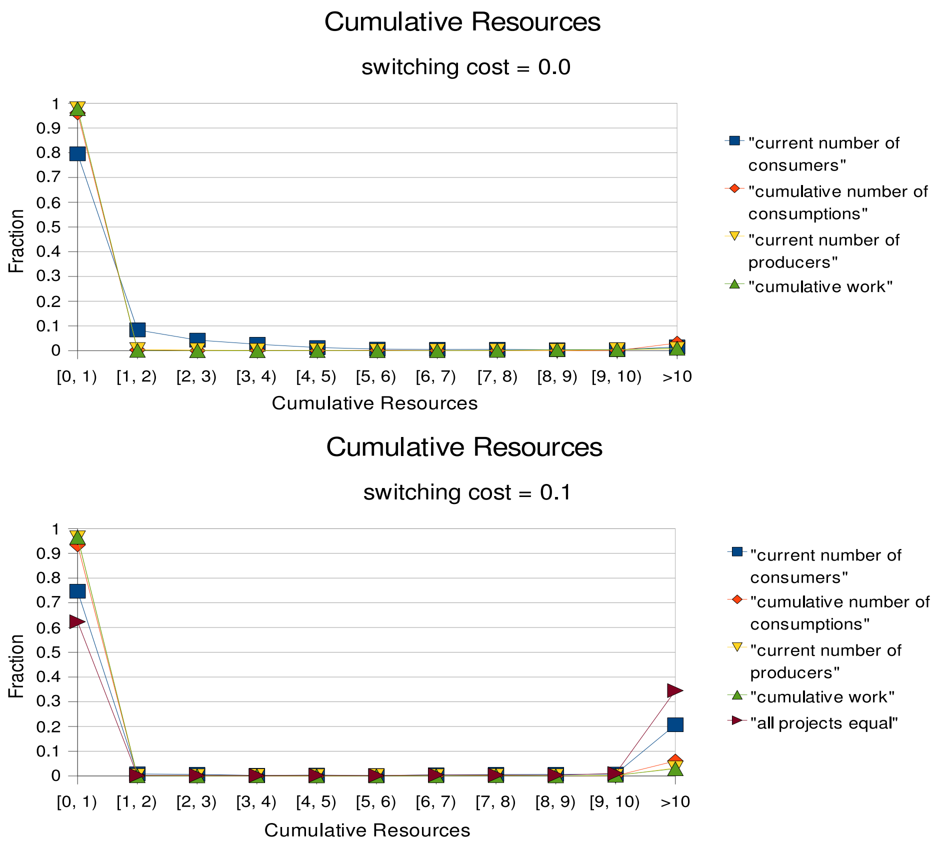


Fig. 2. Fraction of projects as a function of cumulative resources. The upper figure is for switching cost equal to zero and the lower figure is for switching cost equal to 0.1.

must expend to familiarize itself with a new project before it can provide meaningful contributions. Values considered for switching costs are 0.0, 0.1, 0.2, and 0.4. (Note that utility values range between 0.0 and 1.0 so a penalty of 0.4 is significant.)

The model was run for 1000 time steps, thought to be sufficient to allow conditions to stabilize, and populated with 5000 agents. Each parameter combination was run 10 times and the results averaged to account for the stochastic nature of the model.

4.1. Cumulative Resources, Consumer, and Producer Distributions

The following distributions were considered after the 1000$^{\text{th}}$ time step:

- cumulative resources (i.e., amount of work) a project has accumulated
- number of consumers using a project
- number of producers contributing to a project

All results were normalized for comparison of the distributions. The results for switching costs 0.0 and 0.1, and for the five different success formulations for the three different output data are depicted in Figures 2–4.

The *all projects equal* success metric is an outlier when the switching cost is 0.0. This is for the following reason: by causing all projects' success to be equal, the S term in the utility calculation U_c becomes the same for all projects, causing utility (and thus selection) to be based entirely on an agent's and project's matching interests M_i. Essentially, treating all projects as equal successes is the equivalent to selecting a random project for S, the opposite end of the spectrum of using best choice. When paired with a switching cost of 0.0, there is a lack of stability in agents working long-term on certain projects. Instead, agents rapidly flit from one project to another, especially as new projects are created that better match their interests. This results in an explosion in the population of projects, since projects rarely exist for 5 time steps without being worked on by an agent and thus are rarely removed from the simulation. As a result, data from this success metric are not included in the figures when the switching cost is 0.0. For non-zero switching costs the *all projects equal* success metric is used as a baseline to show that agents discriminating among projects using other success metrics does have effect on the resulting distributions.

With a switching cost of 0.0, the cumulative resources distribution is similar for three out of four of the success metrics, with the *current number of consumers* being the exception, as shown in Figure 2. Essentially, most projects accumulate zero or almost no resources. For all success metrics, as the switching cost increases, the cumulative resources morphs towards a bimodal distribution. Projects are either small or large, and the higher the switching cost the greater the number

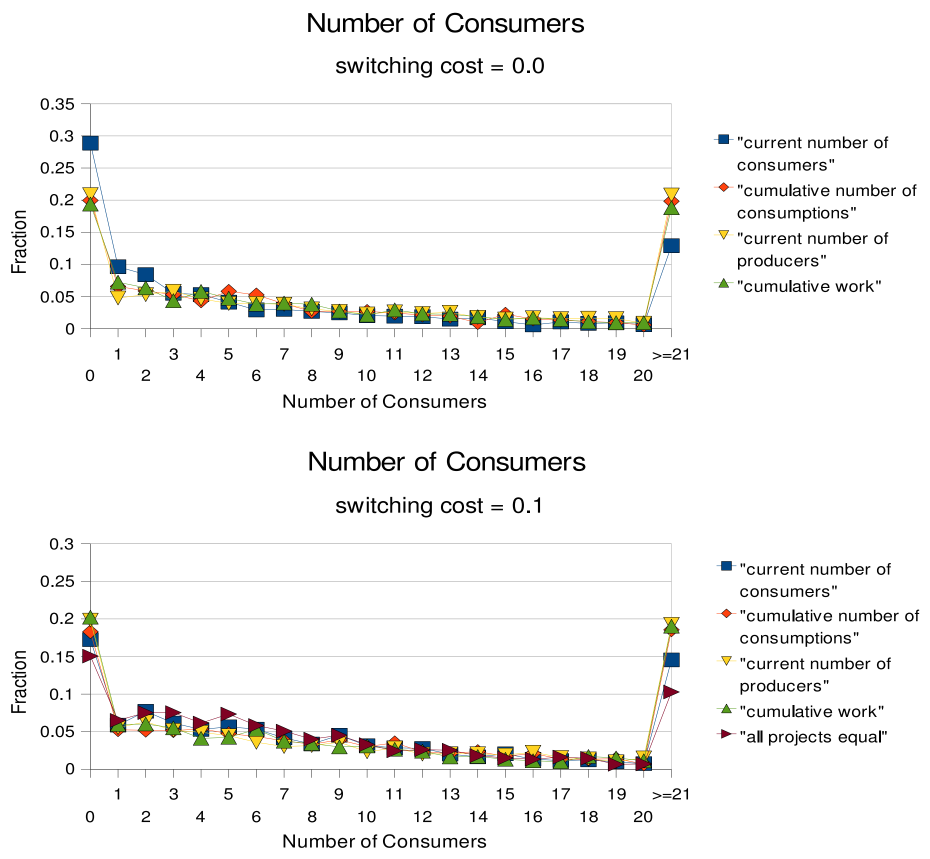


Fig. 3. Fraction of projects as a function of the number of consumers. The upper figure is for switching cost equal to zero and the lower figure is for switching cost equal to 0.1.

of large projects. A non-zero switching cost essentially creates a “stickiness” factor; agents tend to continue, or stick, with the project they were last involved in simply because the penalty of switching to a different project overwhelms the utility of doing so. Thus projects tend to be small, by never attracting agents, or large, by having agents stick with them for extended periods and thus accumulating a substantial amount of work. This is similar to data observed on SourceForge, where most projects never get off the ground, but a few accumulate enough code to become useful software. Note that with a switching cost of 0.1, two groupings emerge: the *cumulative number of consumptions*, *current number of producers*, and *cumulative work* distributions change little with the increased switching cost, while *current number of consumers* becomes more of an outlier. In fact the *current number of consumers* distribution is very similar to the *all projects equal* distribution, indicating this metric is not very discriminating. This is because consumers are restricted from reconsuming the same project twice within 10 time steps. Thus, consuming agents are constantly moving through a subset of favored projects, which causes this metric to perform more like a random selection. Note that producers behave at the other extreme: as the switching cost increases, a producing agent is more and more likely to contribute to the same project in the subsequent time step.

In general, most projects have very few consumers and few projects have many consumers, with a semi-smooth trend connecting the two extremes as shown in Figure 3. Once again, the *current number of consumers* distribution is an outlier when the switching cost is 0.0. All other success metrics, regardless of the switching cost, seem to result in similar distributions, including the baseline of *all projects equal*. This is expected since, unlike contributing agents, consuming agents do not consider a project’s success when calculating its utility U_{u}. Also unlike contributing agents, consuming agents do not include a switching cost in their utility function; therefore, large differences are not expected among the success metrics or by varying the switching cost. Indeed, the minor variations seen must be side effects of other components of the model, such as where producers are contributing.

The number of producers distributions are shown in Figure 4. As was seen in the cumulative resources distributions, once again the *current number of consumers* metric is less similar to the other success metrics and more similar to the baseline of *all projects equal*. This supports the notion that the *current number of consumers* metric is non-discriminating. As the switching cost increases, more producers are associated with projects. This is similar to the results of the cumulative resources distribution and again can be explained by the increased stickiness factor. As the switching cost increases, ignored projects are removed, but those that do attract the attention of producers tend to keep those producers and gain additional producers, which also stay with the project long-

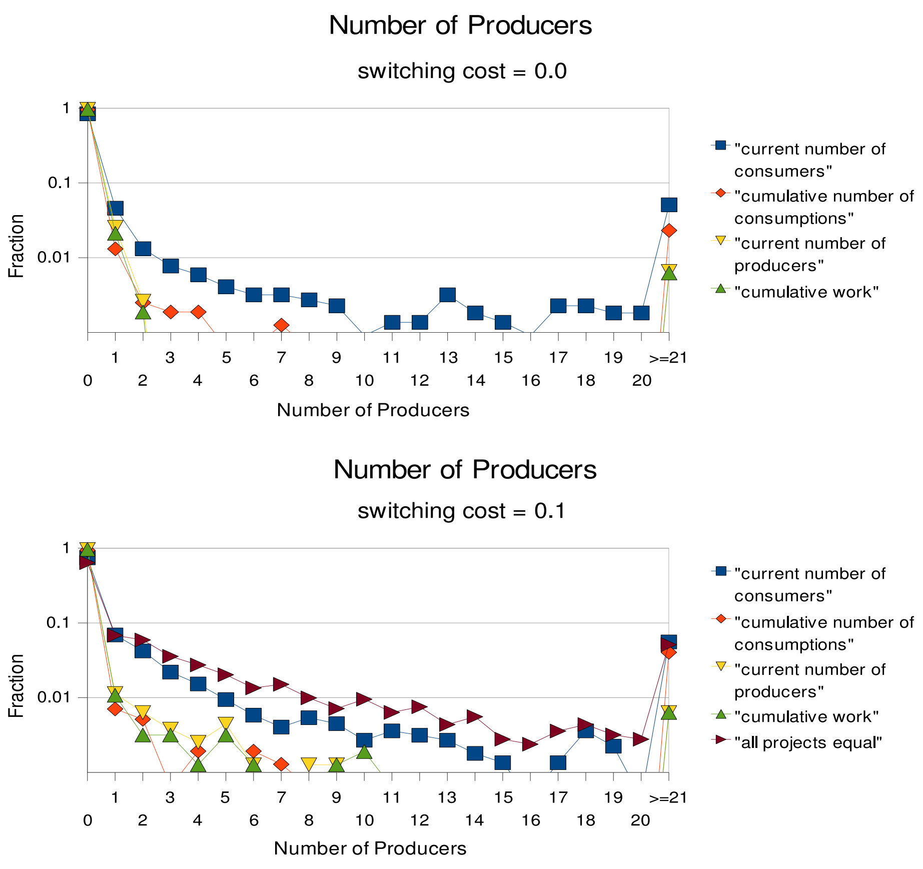


Fig. 4. Fraction of projects as a function of the number of producers. The upper figure is for switching cost equal to zero and the lower figure is for switching costs equal to 0.1.

term, over time. Note the highly skewed distribution from the model mimics data from studies of SourceForge. Averaged over all success metrics and with a switching cost of 0.1, 73% of the model's projects have zero or one developers and 85% have fewer than four developers, as compared to 67% and 90% respectively for SourceForge data [6].

At this level, the evaluated success metrics show only minor variations for the observed distributions. The one exception is the *current number of consumers* metric, which tracks closely with the baseline *all projects equal*, meaning it is similar to random choice. Since the success metrics only impact the producers, it is not surprising that more variation is seen when looking at cumulative resources and the number of producers distributions. Finally, switching cost also has an effect on the model, with projects having more producers and accumulating more resources when switching costs are high.

4.2. Projects' Needs Vector Distributions

Part of the goal of this research is to better understand what types of projects are successful. Recall that new projects are continually added during a simulation run while projects that do not receive contributions for 5 time steps are removed. Thus, at any given time the population of projects consist of active projects, as inactive projects are continually being eliminated. By examining the projects' needs vector distribution, an understanding can be gained about what types of projects survive and are therefore arguably successful. Figure 5 shows the distribution of projects' needs vectors for each of the 5 success metrics. There is not significant variation with different switching costs and therefore only the switching cost of 0.1 is shown. Note that the peaks of the distributions are slightly off center. Recall that needs vectors less than 0.5 are biased towards consumers and greater than 0.5 towards producers. The distributions are skewed to the right, with peaks occurring in the range [0.5, 0.6), supporting the notion that surviving projects tend to be biased to producer needs. This is in alignment with literature that argues open source development is a producer-driven process (e.g., [14], [20], [21]).

Finally, some additional values of surviving projects are examined. Figure 6 contains scatterplots of project needs vectors versus cumulative resources. Each plot contains the results for 10 runs of the model with a switching cost of 0.1. There are separate plots for each of the non-baseline success metrics. Consumer-oriented success metrics cause projects to accumulate a wide range of resources over a wide range of needs vectors, as shown in Figures 6(a) and 6(b). On the other hand, only a very narrow range of needs vectors accumulate resources for producer-oriented success metrics, as shown in Figures 6(c) and 6(d). The behavior of the producer-oriented metrics is closer to that observed at SourceForge, where only a small percentage of projects develop beyond a few lines

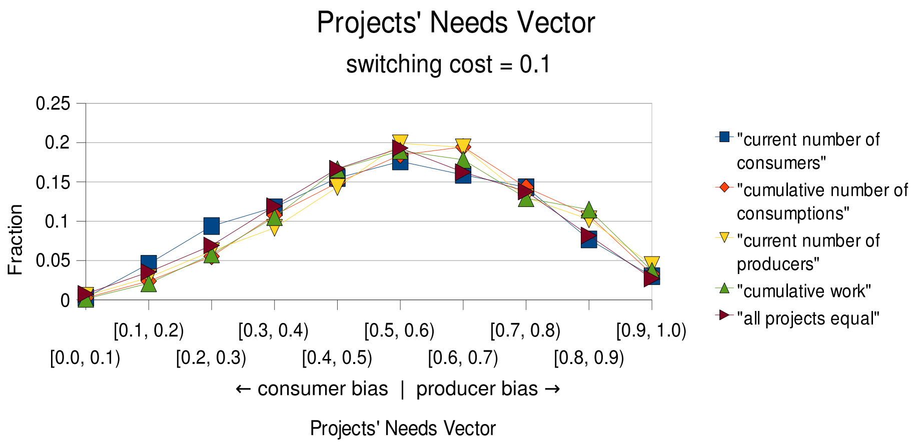
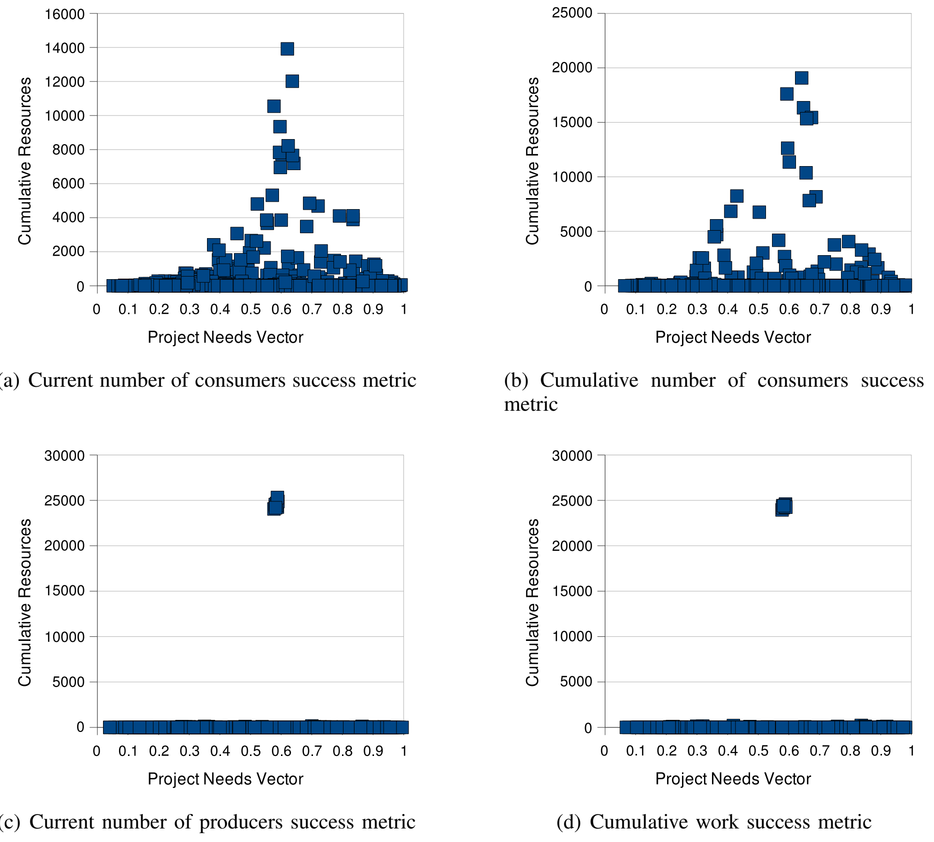


Fig. 5. Fraction of surviving projects as a function of projects' needs vectors. Values less than 0.5 indicate a consumer bias and greater than 0.5 indicate a producer bias.



(a) Current number of consumers success metric

(b) Cumulative number of consumers success metric

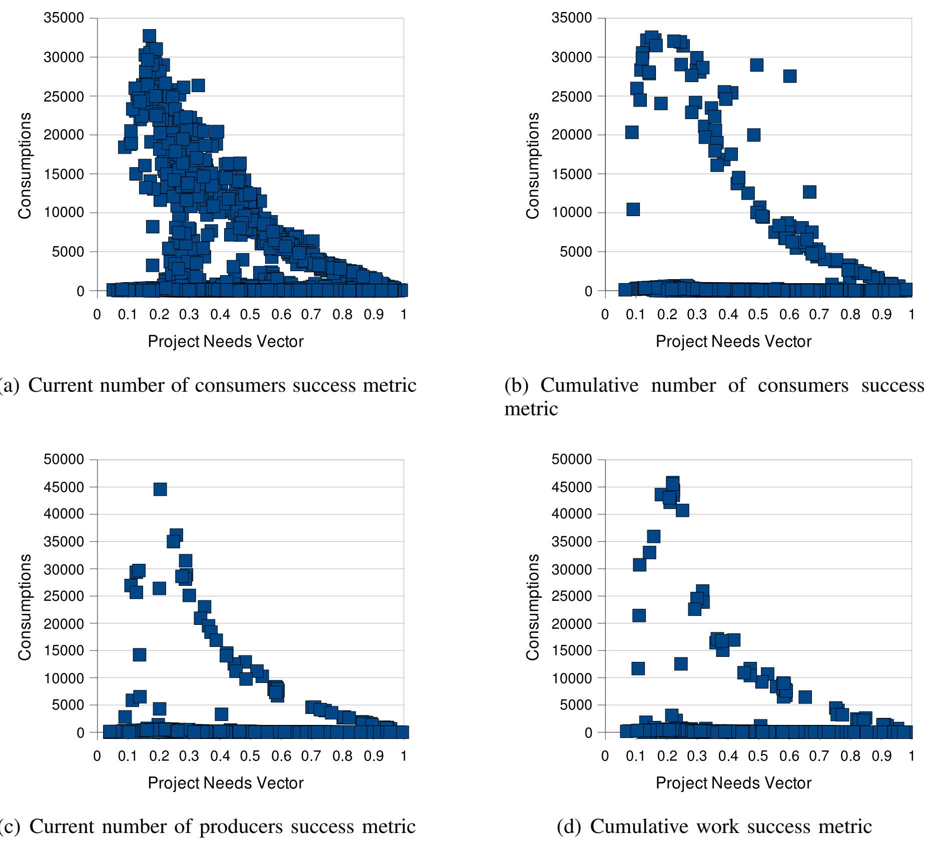
(c) Current number of producers success metric

(d) Cumulative work success metric

Fig. 6. Project needs vectors versus cumulative resources. (a) and (b) are consumer-oriented metrics and (c) and (d) are producer-oriented metrics.

of code. Notice that for all success metrics the maximum resources accumulate for projects with needs vectors around 0.6, or those projects slightly biased towards producers.

Figure 7 shows scatterplots of project needs vectors versus cumulative consumptions. Notice that the peak consumptions occur at project needs vectors less than 0.5, showing a consumer bias. This is logical, as consumers, and not producers, affect the number of times a project is downloaded. Consumer-oriented metrics result in more consumed projects with low needs vectors. To understand why, recall that only producers affect which projects survive. When calculating the utility U_c of a project, producers consider two factors: the matching interests M_i and the success S. The matching interests is always biased towards producer-oriented projects. Thus when using producer-oriented success metrics, producing agents mostly select projects with needs vectors greater than 0.5. On the other hand, using consumer-oriented success metrics causes the second term in the utility function to favor consumer-oriented projects and thus more projects with needs vectors less than 0.5 survive. Consumers then select projects for download based solely on matching interests and thus will gravitate towards consumer-oriented projects if any exist. The reason the peak



(a) Current number of consumers success metric

(b) Cumulative number of consumers success metric

(c) Current number of producers success metric

(d) Cumulative work success metric

Fig. 7. Project needs vectors versus cumulative consumptions. (a) and (b) are consumer-oriented metrics and (c) and (d) are producer-oriented metrics.

consumptions are not more off center is because most agents have only a slight consumer or producer bias, as a result of how p_c and p_u are initially assigned and mapped to agent needs vectors. Thus consumer-oriented projects with very low needs vectors do not receive as many downloads as values just slightly less than 0.5 simply because there are more agents with mid-value needs vectors. A sample distribution of agents' needs vectors is shown in Figure 8.

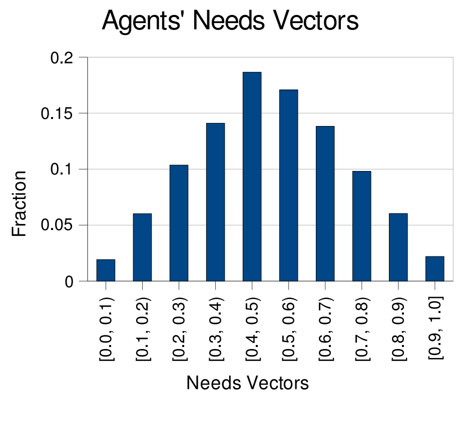
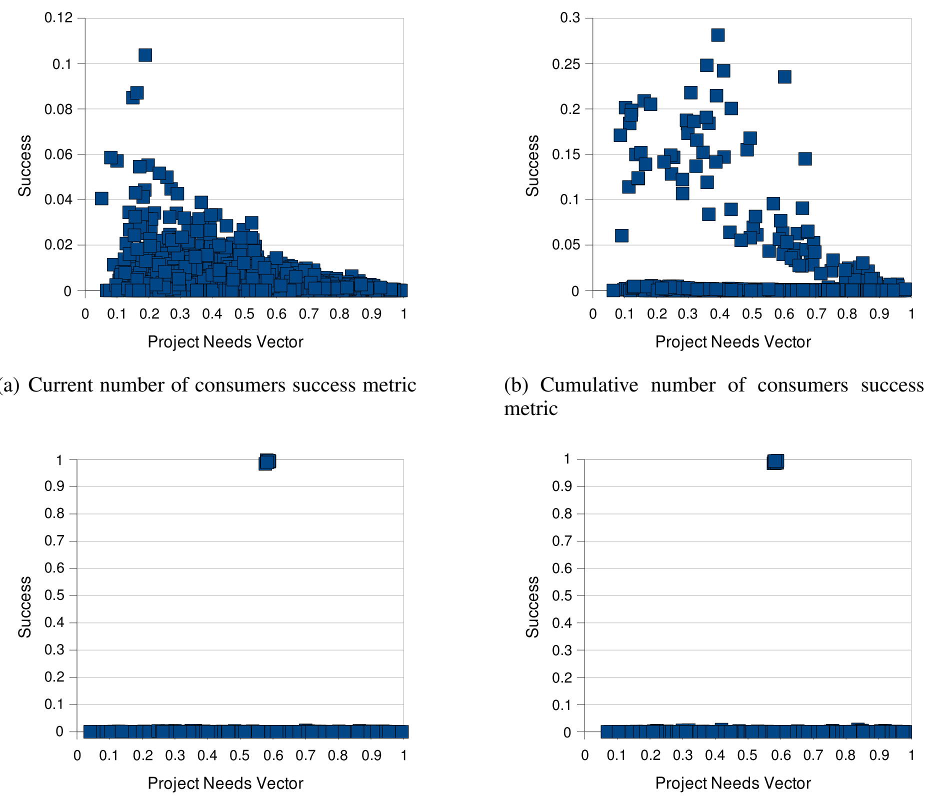


Fig. 8. Histogram of agents' needs vectors. Most agents have only a slight consumer (< 0.5) or producer (> 0.5) bias.

Figure 9 shows scatterplots of project needs vectors versus success. With consumer-oriented metrics, success values, even for the best projects, are low, and the project needs vector for the most successful projects is ill-defined. The producer-oriented metrics are just the opposite. All successful projects cluster tightly around a needs vector value of 0.59, again indicating a producer bias. As mentioned earlier, this value is not more extreme because the majority of agents have needs vectors near the center. The fact that all projects have extremely high success values shows the tendency of one project to dominate over all others. Essentially, it is almost impossible for a new project to become successful once there is an established successful project. This is partially due to agents always selecting the best project and is akin to the notion that popularity begets popularity. This explains why there are more non-zero values in Figures 6, 7, and 9 when using consumer-oriented metrics. Consumer-oriented metrics cause a diversity in the population of surviving projects through a lack of feedback loop: producers are attracted to projects popular with consumers, while consumers are attracted to projects that match their interests. When using producer-oriented metrics, producers flock to projects that are popular with producers, which in turns makes those projects even more attractive to other producers. The positive feedback loop means only a small number of projects survive, and once a successful project is established, it becomes almost impossible for new projects to gain developers and thus increase their success.

In summary, some differences are observed in surviving projects needs vectors when using different success metrics. For example, there is more variability and a larger number of non-zero values for cumulative resources, cumulative con-



(a) Current number of consumers success metric

(b) Cumulative number of consumers success metric

(c) Current number of producers success metric

(d) Cumulative work success metric

Fig. 9. Project needs vectors versus success. (a) and (b) are consumer-oriented metrics and (c) and (d) are producer-oriented metrics.

sumptions, and success when using consumer-oriented metrics. There is also some minor variation in peak values, although all metrics for cumulative resources and success show a producer bias while all metrics for cumulative consumptions show a consumer bias. Thus, when choosing a success metric, the exact success metric may be less important than whether it belongs to a consumer or producer category. Within one of these two categories, the differences are minor.

5. Conclusion

Digital public goods, such as open source software, are produced by volunteers who contribute content for free. Empirical analysis shows that digital public goods experience unequal distributions of various attributes, such as the number of downloads/views of a project or the number of developers associated with a project. What causes these skewed distributions? Survey research shows that one of the hypotheses is that users are attracted to successful projects. However, there is no generally agreed upon definition of a successful project.

We create a simple public goods model to study the impacts of using different success metrics. Our analysis with different definitions of success shows that using different forms of measuring success has an impact on the model's output. Success may be viewed differently by consumers versus producers. We therefore categorize our success metrics into consumer-oriented and producer-oriented groups. In general, we find differences between these two groups. Consumer-oriented metrics result in larger and more diverse populations of projects. Within the consumer-oriented category, we find more variation, including some cases where the *current number of consumers* metric behaves like random selection. The variability for producer-oriented metrics is much smaller. Finally, we demonstrate that our model is producer-driven and argue that the data generated by the model when using producer-oriented success metrics has characteristics that more closely match real world data, supporting the notion that public goods are driven by the interests of producers, not consumers.

References

[1] S. C. Smith and A. Sidorova, "Survival of open-source projects: A population ecology perspective," in *ICIS 2003. Proceedings of International Conference on Information Systems 2003*, Seattle, WA, 2003.

[2] D. Anthony, S. W. Smith, and T. Williamson, "Explaining quality in Internet collective goods: Zealots and good samaritans in the case of Wikipedia," Online, November 2005, retrieved November 24, 2008 from http://web.mit.edu/iandeseminar/Papers/Fall2005/anthony.pdf.

[3] J. O. Ledyard, "Public goods: A survey of experimental research," in *The Handbook of Experimental Economics*, J. H. Kagel and A. E. Roth, Eds. Princeton, New Jersey, USA: Princeton University Press, 1995, pp. 111–194.

[4] R. M. Isaac, J. M. Walker, and A. W. Williams, "Group size and the voluntary provision of public goods: Experimental evidence utilizing large groups," *Journal of Public Economics*, vol. 54, no. 1, pp. 1–36, May 1994. [Online]. Available: http://ideas.repec.org/a/eee/pubeco/v54y1994i1p1-36.html

[5] N. P. Radtke, M. A. Janssen, and J. S. Collofello, "What makes Free/Libre Open Source Software (FLOSS) projects successful? An agent-based model of FLOSS projects," *International Journal of Open Source Software and Processes*, in press.

[6] D. Weiss, "Quantitative analysis of open source projects on SourceForge," in *Proceedings of The First International Conference on Open Source Systems (OSS 2005)*, M. Scotto and G. Succi, Eds., Genova, Italy, 2005, pp. 140–147.

[7] M. A. Rossi, "Decoding the "Free/Open Source(F/OSS) Software puzzle" a survey of theoretical and empirical contributions," Dipartimento di Economia Politica, Università degli Studi di Siena, Quaderni 424, Apr. 2004.

[8] S. Krishnamurthy, "Cave or community?: An empirical examination of 100 mature open source projects," *First Monday*, vol. 7, no. 6, June 2002. [Online]. Available: http://www.firstmonday.org/issues/issue7_6/krishnamurthy/index.html

[9] R. A. Ghosh, B. Krieger, R. Glott, and G. Robles, "Part 4: Survey of developers," in *Free/Libre and Open Source Software: Survey and Study*. Maastricht, The Netherlands: University of Maastricht, The Netherlands, June 2002.

[10] K. R. Lakhani, B. Wolf, J. Bates, and C. DiBona, "The Boston Consulting Group hacker survey," Online, 2002, http://www.osdn.com/bcg/.

[11] F. Ortega, J. M. Gonzalez-Barahona, and G. Robles, "On the inequality of contributions to Wikipedia," in *HICSS '08: Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 304–310. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2008.333

[12] J. Voss, "Measuring Wikipedia," in *Proceedings of the 10th International Conference of the International Society for Scientometrics and Informetrics*. ISSI, July 2005, pp. 221–231.

[13] K. Crowston, J. Howison, and H. Annabi, "Information systems success in free and open source software development: Theory and measures," *Software Process: Improvement and Practice*, vol. 11, no. 2, pp. 123–148, March/April 2006.

[14] K. Crowston and B. Scozzi, "Open source software projects as virtual organizations: Competency rallying for software development," in *IEE Proceedings Software*, vol. 149, no. 1, 2002, pp. 3–17.

[15] Y. Wang, "Prediction of success in open source software development," Master of science dissertation, University of California, Davis, Davis, CA, Spring 2007.

[16] R. English and C. M. Schweik, "Identifying success and tragedy of FLOSS commons: A preliminary classification of SourceForge.net projects," in *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development*. Washington, DC, USA: IEEE Computer Society, 2007, p. 11.

[17] A. Lih, "Wikipedia as participatory journalism: Reliable sources? Metrics for evaluating collaborative media as a news resource," in *Proceedings of the 5th International Symposium on Online Journalism*, Austin, TX, USA, 2004. [Online]. Available: http://staff.washington.edu/clifford/teaching/readingfiles/utaustin-2004-wikipedia-rc2.pdf

[18] J. Giles, "Internet encyclopaedias go head to head," *Nature*, vol. 438, no. 7070, pp. 900–901, December 2005. [Online]. Available: http://dx.doi.org/10.1038/438900a

[19] T. Chesney, "An empirical examination of Wikipedia's credibility," *First Monday*, vol. 11, no. 11, 2006.

[20] E. S. Raymond, "The cathedral and the bazaar," Thyrsus Enterprises, Tech. Rep. 3.0, September 11 2000.

[21] J. Lerner and J. Tirole, "The scope of open source licensing," *Journal of Law, Economics, and Organization*, vol. 21, no. 1, pp. 20–56, April 2005. [Online]. Available: http://dx.doi.org/10.1093/jleo/ewi002

Nicholas P. Radtke is a Ph.D. candidate and teaching assistant in Computer Science at Arizona State University. His research focuses on understanding and modeling the Free/Libre Open Source Software engineering processes.

Marco A. Janssen is an Assistant Professor on formal modeling of social and social-ecological systems within the School of Human Evolution and Social Change at Arizona State University. He is also the Associate Director of the Center for the Study of Institutional Diversity. His formal training is within the area of Operations Research and Applied Mathematics. His current research focuses on the fit between behavioral, institutional and ecological processes. In his research he combines agent-based models with laboratory experiments and case study analysis. Janssen also performs research on diffusion processes of knowledge and information, with applications in marketing and digital media.