A Collaborative Group Developing Method of Open Source Software by Mimicking Altruism Behavior of Army Ant

Takumi Ichimura, Takuya Uemoto, and Shin Kamada

Abstract—Recently, the developing software project is complex tasks because many projects in developing process are processed parallel and many developers join two or more projects. To improve the efficiency of group developing software, various methods are executed and the analysis of their behaviors is important. The community of the developers are analyzed by using analyzing method of network structure. We develop the simulation system of developing software with altruism behaviors. The system was developed on the basis of developers behaviors of SourceForge.net. In this paper, we observed the community growing process under the agent's altruism behaviors. This paper reports the efficiency of altruism behavior in the software developing community.

Index Terms—Altruism behaivor, Analysis of Group development Process, Growing Process of Social Community

I. INTRODUCTION

A social community is composed of a group of individuals with the common interest and purpose. The size of community is determined by the number of people and their activities. We can consider that the attraction to the community increased according to the integration of the interaction of the personal behaviors, and then it grows to be a larger society.

In the community of open source software (OSS) developers and projects, we can observe the collaborative social network in the developing process of system. Madey et al.[1] has been developed the structural and the dynamic mechanisms that govern the topology and evolution of the OSS developer's social community to analyze the developer and project data from SourceForge [4]. According to their ideas, the collaborative social network is formed in global virtual self-organizing teams of software developers working on open source software projects and they developed the self-organizing simulation model. Because the self-organization mechanism is the important property of many social networks, a self-organizing network grows and develops structure without centralized decision making as shown in the real open source software community.

In their model, the developer in the open source software community is an agent. Some agents create a new project (community). The project is the team to develop a software

(system). The agent can select only an action representing a real developer's possible daily interaction in the process of developing system. The model has 4 kinds of behaviors: "Developers can create a project (create a project)", "Developers can join into a project (join a project)", "Developers can abandon a project (leave a project)" or "Developers can continue the current collaboration (no any actions)." By such the actions, a new project can be created and be grown, some projects are disappeared. The agent can contact to other agent in the another community and join two or more projects simultaneously. The change in the growing community can be observed through the simulation. The experimental result described in [1] showed the linear relation between the number of projects and developers.

The model described in [2] assumed that the project has some tasks and the agent has the skill for the developing software. The 'task' is the amount of work for the developing software. The 'skill' is the amount of works that the agent develops the program for a given time. Our developed agent based simulation model shows some spiral line in the relation between the community and the action of agents.

In this paper, we propose the collaborative group developing method by mimicking altruism behavior of army ant agent system[3]. The agent decides to join into the project on the based on the altruism behaviors by imitating the army ants. If the agent with high ability has some time for working finds the project with less work force, the agent leads to participate in the developing team. Their tasks will be increased, but the total efficiency of developing software will be increased. This paper discusses the relation between the number of altruism agents and total cost of projects through some experimental results by using simulation system for the Source Forge community data[4].

II. OPEN SOURCE COMMUNITY DATA

There are some software distribution web site for many open source communities such as GNU project. The Source Forge is one of popular software repositories and has the function of the management system of open source software. The website of Source Forge gives the information of the projects and the developers recorded in the database [5].

The data related to project and developer are obtained from the database and are used to investigate the relation between the number of project and developer who joins the project.

Fig.1 shows the logarithm of the number of project and developer at September 2014 (Schema Sf0914). Schema Sf0914 has 73 tables including 'User Group.' The table 'User Group' consists of 17 columns including 'user_id' and

'group_id'. We retrieved all data from 'sf0914.user_group' and counted the number of records according to group_id. As a result, the number of developers to each project is calculated. The horizontal axis is the number of developers and the vertical axis is the number of projects for the numbers of developers. As shown in Fig. 1, we can observe that the relation decreased linearly for a certain period and then the line spreads from the certain point like delta. In this paper, we investigate the inflection point and the expanse of the delta and the equation of line is determined according to empirical studies.
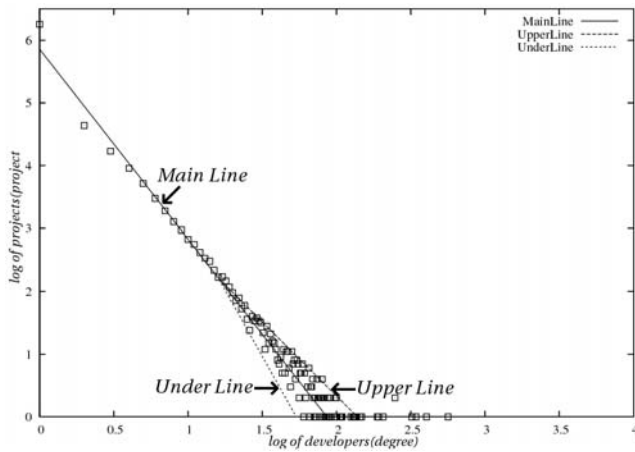

Fig. 1.   The distribution for project and developer in Source Forge

III. Simulation Model

In our developed simulation system, there are 2 kinds of agents: 'major agent' and 'minor agent'. The agent has the own function of creating, joining to, or leaving from the project. Because some developers becomes the leadership in the real world of software developing, the simulation model has the 2 kinds of agents behaviors inspired by the interaction behavior of army ant system [6], [7].

In our model, a major agent creates the project and a minor agent makes the participation behavior to the project and the leaving behavior from one. A major agent and a minor agent is a project manger and developer, respectively. When some minor agents join the same project, they worked to share the task of software development. The 'task' and the 'skill' means the numerical values representing the amount of estimated codes for the software development and the agent's coding capability in the periodical time, respectively.

SourceForge can provide open source softwares which are classified into 10 categories such as "Audio & Video", "Business & Enterprise", "Communications", "Development", "Home & Education", "Games", "Graphics", "Science & Engineering", "Security & Utilities", "System Administration" [4]. If a major agent creates the project of software, it must determine the project's category. The development work will have two or more types of tasks during developing the software. There are 3 kinds of common types of tasks; 'network communication processing', 'database processing', and 'graphic processing'. The minor agent cannot join the project, if it does not have the skill for the corresponding category. If two or more agents join a project, the task for the developing project is shared to each agent equality.

A. Task and Skill

A project has one or more tasks in the software development. Let P_i^k be the amount of task k in the project i. P_i^k takes a value in $[0, 2]$. The sum of P_i^k for all tasks is defined as $\mathbf{P_i} = \{P_i^1, P_i^2, \cdots, P_i^n\}$.

A skill for the agent j in task k is defined as S_j^k. The initial value is a given random number in $[0, 1]$. The agent works the allocated task with the agent's skill S_j^k. That is, the time to finish the task is calculated by dividing the amount of task of an agent W_{ij}^k by the agent skill S_j^k. The working time for each project to which the agent joins is added. The total time is the summation of T_j for each agent as follows.

$$T_j = \sum_i T_{ij} = \sum_i \frac{W_{ij}^k}{S_j^k}, \tag{1}$$

where W_{ij}^k is the amount of the task k which is assigned to the agent j in the project i. That is, $W_{ij}^k = \frac{P_i^k}{N_i}$. T_j satisfies the condition $0 \leq T_j \leq T_{limit}$. The upper limit of time T_{limit} is 24, because of 24 hours a day.

B. The behaviors of agent

In [1], the proposed model has 4 kinds of behaviors: "Developers can create a project (create a project)", "Developers can join into a project (join a project)", "Developers can abandon a project (leave a project)" or "Developers can continue the current collaboration (no any actions)." In this paper, the model has 2 kinds of agents: the major agent and the minor agent. The following actions are implemented by 2 kinds of agents, respectively.

- Major agent
  - Create a project
- Minor agent
  - Join a project
  - Leave a project
  - Take nothing action

A minor agent selects an action from 3 kinds of behaviors per a step and it works according to the selected behavior by the agent's action algorithm.

C. Create a project

A major agent creates a new project,
- when it starts to develop a new software. In this case, a created project is called New one.
- when the agent creates the sub projects to share the large tasks in the project which is created by the major agent. In this case, a created project is called Sub one.

If the major agent j creates a new project, the agent works to follow next behavior's algorithm.
1) The probabilities New or Sub of creating a project for each case are given appropriate values.
2) If a major agent creates a new project, an agent checks the condition that the probability New equals to or greater than the random value $[0, 1]$ by using Monte Carlo method.
3) If a major agent creates a sub project to share the tasks, the major agent determines by Eq.(2) and Eq.(3).

$$Sub_i^k = \frac{P_i^k}{N_i^k}, \tag{2}$$

where Sub_i^k is the average amount of tasks of an minor agent in the developing task k of the project i. P_i^k is the task k assigned to the agent and N_i^k is the number of agents in the project i.

$$\begin{cases} IF \ \ Sub_i^k \geq Sub_{threshold} \ \ THEN \ CreateProject \\ IF \ \ Sub_i^k < Sub_{threshold} \ \ THEN \ NotCreate, \end{cases} \quad (3)$$

where $Sub_{threshold}$ is a given value to share the tasks and is a threshold to create a sub project.

The major agent checks the value of Sub_i^k for each task which should be shared in the point of load balance. Once a sub project is created, some minor agents with skills which is corresponded to the category of developing task will participate to it.

D. Join a project

The minor agent j joins the project according to the following process.

1) The minor agent j selects one project which it has not participated yet.
2) The probability to be joined in the selected project is determined by Eq.(5).
3) The minor agent determines to join the project by Eq.(6).

If the estimated time till the end of the task exceeds the threshold T_{limit}, the minor agent gives up the joining the project.

1) Selection of project with Network growing model: When the agent selects a project, some strategies to a network growing model are required. In this paper, Barabasi-Albert(BA) model with dynamic fitness[8] is adapted to decide the behaviors of agents. BA model is an algorithm for generating random scale-free networks based on the preferential attachment mechanism. Scale-free network is widely used in natural and human-made systems such as social community. In BA model, an edge is connected to the node with high number of degrees in high probability[9]. BA model with dynamic fitness employs the fitness of node is dynamically changed [8]. A degree of software is the number of agents worked in the software development and a minor agent selects a software based on this model. In this model, the linkage to the other agent is implemented by using BA model, but a new node is not added.

The selection probability S_i for the software i by a minor agent is defined as follows.

$$S_i = \frac{\eta_i N_i}{\sum \eta_i N_i}, \quad (4)$$

where η_i is the time decreasing function representing the fitness of software i and N_i is the degree of software, that is, the number of participating agents. The dynamic fitness function realizes the growing community, the stability and the attenuation of community growing.

2) Probability of joining a project: The probability of participation to a project is calculated by the tasks of the project and the skill of the agent j. If the agent does not have enough skills for the category of the project i, the probability is 0. Otherwise, the probability is calculated as follows.

$$J_{ij} = \frac{\sum_{k=1}^{n}(P_i^k - \sum_{j=1}^{n} S_j^k)}{\sum_{k=1}^{n} P_i^k} \quad (5)$$

$$\begin{cases} IF \ \ J_{ij} \geq J_{threshold} \ \ THEN \ JointProject \\ IF \ \ J_{ij} < J_{threshold} \ \ THEN \ NotJoin, \end{cases} \quad (6)$$

where $\sum_{k=1}^{n}(P_i^k - \sum_{j=1}^{n} S_j^k)$ is how tasks of the project have reached the completed work, J_{ij} is calculated by dividing it by the total amount of tasks $\sum_{k=1}^{n} P_i^k$. If J_{ij} is larger than threshold $J_{threshold}$, a minor agent joins a project.

E. Leave a project

The minor agents can leave from the working project by the following algorithm.

1) The minor agent j selects one project among all projects to participate.
2) The probability of leaving to participate the project is calculated.
3) A minor agent decides the project to leave by Eq.(7).

The probability of leaving a project is calculated as follows.

$$L_{ij} = \sum_{k=1}^{n}(\frac{P_i^k}{N_i^k} - S_j^k) \quad (7)$$

$$\begin{cases} IF \ \ L_{ij} \geq L_{threshold} \ \ THEN \ LeaveProject \\ IF \ \ L_{ij} < L_{threshold} \ \ THEN \ NotLeave, \end{cases} \quad (8)$$

where $\frac{P_i^k}{N_i^k}$ is the amount of task per an agent, L_{ij} is calculated by subtracting the agent skill from $\frac{P_i^k}{N_i^k}$, and $L_{threshold}$ is a given threshold value. If the load of the agent j (L_{ij}) is greater than $L_{threshold}$, the agent j leaves from a project i.

In Eq.(7), the minor agent can leave from a project with ease, when the amount of the task of the project is high or the skill of agent is low or in both situations.

IV. AGENT BEHAVIOR WITH ALTRUISM BEHAVIOR

A. Altruism Behavior in Army Ant Systems

In animal societies, self-organization is the theory of how minimal complexity in the individual can generate greater complexity at the population. The rules specifying the interactions among the components in the system are implemented by using only local information without global information. In the study of social evolution, army ant performs altruism as one behavior of complexities, where each individual reduces its own fitness but increases the fitness of other individuals in the population. Such behaviors seem to be involved acts of self-sacrifice in order to aid the others. In evolutionary biology, such a behavior is called reciprocal altruism. The concept was initially developed to explain the evolution of cooperation as mutually altruistic acts[10]. The basic idea is close to the strategy of "equivalent relation" in the study of strategic decision making.

Army ants are characterized by their two different phases of activities, a nomadic phase and a stationary phase. During the nomadic phase, army ants move during the day to capture insects, spiders, and so on. The stationary phase starts when the larvae pupate for a few weeks. Moreover, army ants build a living nest with their bodies instead of building a nest like other ants. Each ant will hold on to the other legs and form a linked chain or a ball structure. This behavior is known as a bivouac. This allows the bridging of an empty space.

By inspiring Ichimura et al. developed the multi-agent simulation system to execute more realistic altruistic behavior where two or more kinds of agents realize the division of roles in army ants[7]. Such behaviors also seen in the collaborative social networks. Such observations of behaviors in the computer simulation strongly will contribute to the shift to knowledge and power from the individual to the collective. We will explore how agent-based modeling and simulation can be used as a research technique to study collaborative social networks[2], [7].

B. Decision Rule for Altruism Behavior

If the agent participates in the project, the agent makes a link to the other project which it has already joined. The link means the relation among agents. Even if the link is not a directed link, there is a weak relation to communicate with the corresponding agent via other agents. In this paper, we define the number of links between agents as the 'distance.'

Under the environment, the minor agent j_1 moves in the following. $P(Alt)$ is the probability for altruism behavior and its initial value is given.

1) If the $P(Alt)$ is larger than random value at each step, the agent makes altruism behavior.
2) The agent with altruism behavior finds the agents j_2 within distance r.
3) All agents in the project i_1, which the agent j_1 participates, calculate Eq.(9) for the evaluation of project i_2 which the selected agent participates, and they decide to participate the project i_2 according to the evaluation.

$$R_{lm} \times \frac{P_i}{N_i} \geq A_{threshold}, \tag{9}$$

where R_{lm} is a binary value. If the distance between the agent in the project i_1 and the agent in the project i_2 is within r, R_{lm} is 1. Otherwise, 0. P_i is the task amount of project i_2, N_i is the number of agents in the project i_2, $A_{threshold}$ is a threshold value. If T_j is larger than T_{limit}, even if the agents with altruism behavior cannot participate to the project.

V. SIMULATION RESULTS

This section describes the effects of the growing community by introducing the concept related to the tasks and the agent skills through the simulation results. In this paper, we prepared the simulation environment with 1,000 major agents and 20,000 minor agents. The threshold values $J_{threshold}$ in Eq.(6)and $L_{threshold}$ in Eq.(8) are used to find the optimal set by using the following patterns. The parameters are important to decide the action of the minor agents.

$$(J_{threshold}, L_{threshold}) = (0.0, 0.0), (0.5, 0.0), (1.0, 0.0),$$
$$(0.0, 0.5), (0.5, 0.5), (1.0, 0.5), (0.0, 1.0), (0.5, 1.0), (1.0, 1.0)$$

Fig.2 shows the relation of the number of developers and the numbers projects at the 1,000 iterations. The horizontal axis is the number of developers and the vertical axis is the number of projects for the numbers of developers. For 9 kinds of parameter sets, the simulation results as shown in Fig.2 are observed. The relation between the developers and the projects shows the linearity, if the number of developers is less than 1. Otherwise, the relation has the spread in the direction of the number of projects. In case of 1 or more

developers, the curve of the relation depicts the spiral shape with polar coordinate. We tried to express the characteristic pattern as the following equations.

$$\begin{cases} r & = a(\theta) \times \theta \\ a(\theta) & = 0.07 - 0.07 \times \frac{\theta}{2\pi} \end{cases} \tag{10}$$

$$\begin{cases} r & = a(\theta) \times \theta - \delta \ (\theta_1 \leq \theta) \\ \delta & = 1.6 \times \frac{\theta - \theta_1}{2\pi}, \end{cases} \tag{11}$$

where δ means the width in the spread range. θ is the angle in the polar coordinate and θ_1 is the angle at the appropriate point where the number of developers reaches 1.

The characteristic patten drawn by Eq.(10) and Eq.(11) can be seen in 9 kinds of parameter sets.
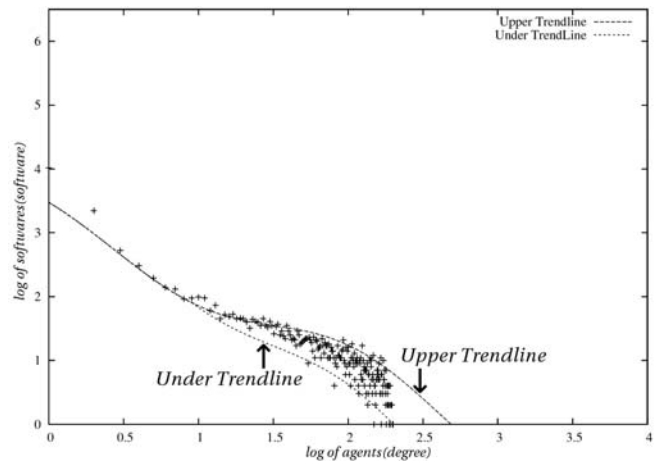

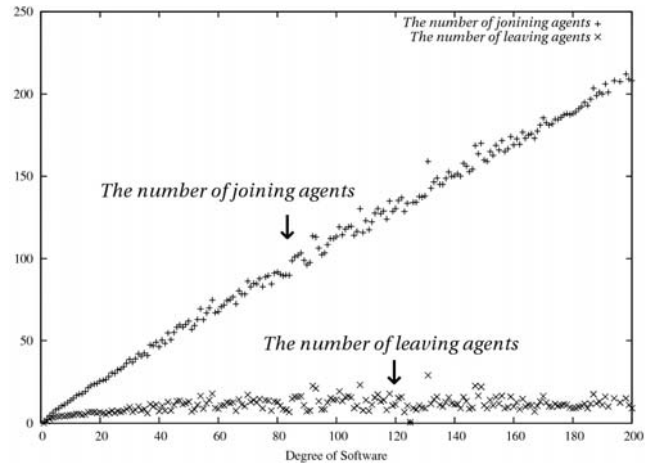Fig. 2. The relation between developers and projects with BA model

Fig. 3. The number of the participate agents and the leaving agents

Moreover, we consider the relation between the behaviors of the agent and the number of agents belonging to the project. Fig.3 shows the number of participating agents and the leaving agents for the tasks in the project. The horizontal axis is the number of tasks of the project and the vertical axis is the number of agents which take any actions. In case of the participating action and the leaving action, as the number of developers becomes large, the distribution approximates linearly. In addition, Fig.4 shows the amount of tasks and
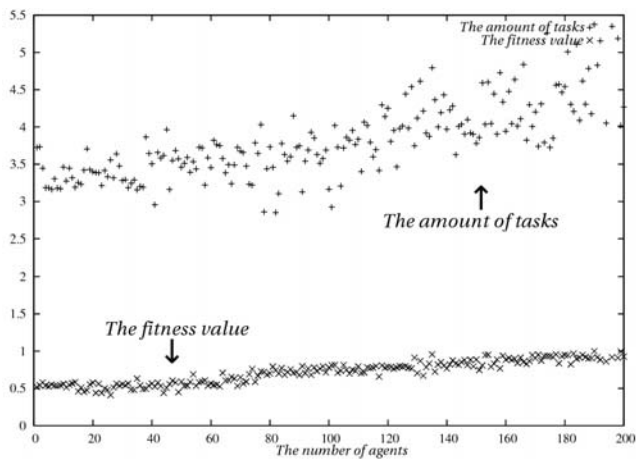
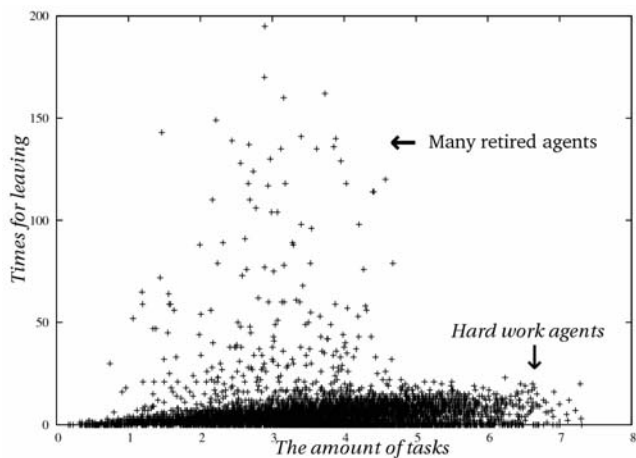Fig. 4.   The relation between the task and the fitness

Fig. 6.   The relation between the projects and developers with altruism behaviors

Fig. 5.   The leaving agents for the amount of tasks

software, the software was counted. The developing for the agent with altruism behavior shows the good performance.

TABLE I
TOTAL DEVELOPING TIME

	Software	average time
w.o. altruism	6555	7.70
w. alturism	26119	3.74

Fig. 7 shows the leaving agents for the amount of tasks with altruism behaviors. As compare with Fig.5, the agents with altruism behaviors shows that the leaving agents from the projcts are decreased.

Fig. 7.   The leaving agents for the amount of tasks with altruism behaviors

the fitness when the task of the projects is created. We can see that the number of tasks in the project is large as the amount of tasks and the value of the fitness becomes high. Because the agent can select the tasks with high fitness easily by Eq.(4) and the agent can participate the project with the large tasks easily by Eq.(5).

Fig.5 shows the changing number of leaving times as the amount of task increases. The increase tendency of the leaving agents appeared in the direction of large tasks of the software. Too many tasks in the developing software keep the agents at a distance. Especially, if there are many agents with the low capability in the developing software, that is, the load per agent is high, the leaving agents may increase. The situation may occur within 5 tasks in the software.

VI. SIMLATION RESULTS WITH ALTRUISM BEHAVIORS

Fig. 6 shows the relation between the projects and developers with altruism behaviors. As compare with Fig.2, the number of participated software was decreased in the log number of agents with $[0.0, 0.5]$. On the other hand, in the $[0.5, 1.5]$ the number was increaed. The simulation result shows that the growing process occurs by the agents with altruism behaviors.

Table I shows the total average time for developing softwares. If there is an agent in the process for developing
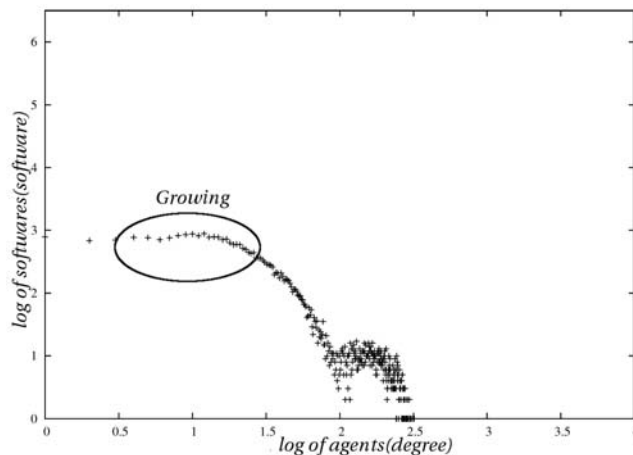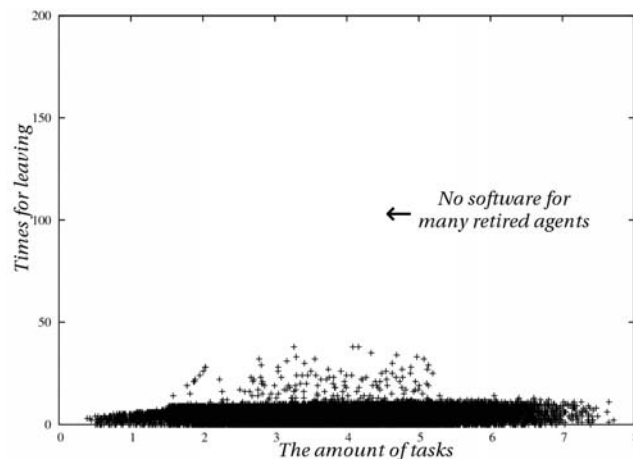
VII. CONCLUSION

With respect to community construction in software development, we examined with the agent based simulation with representing the relation between the task of project and the skill of the agents. The action that an agent selects software project by BA model with dynamic fitness. We can see that the project in the software with the larger tasks and with the higher fitness are easy selected by many agents. Because the participating probability becomes greater in case of the large task and the larger fitness value.

The solution for the effectiveness of software development will be expected for more complex system development. The altruism behavior, which is a characteristic behavior of army ant and is a behavior that an army ant sacrifices its own body for the benefit of another ants, was good performance for software developing community. We will investigate the user's behavior in social network such as Facebook in near future.

REFERENCES

[1] G.Madey, Y.Gao, V.Freeh, R.Tynan, C.Hoffman, *Agent-based Modeling and Simulation of Collaborative Networks*, Proc. of Americas Conference on Information System(AMCIS2002) pp.1839-1842(2003).

[2] T.Ichimura, and T.Uemoto, *Analysis of the Social Community Based on the Network Growing Model in Open Source Software Community*, Proc. of IEEE 8th International Workshop on Computational Intelligence and Applications (IWCIA2015), pp.149-153(2015)

[3] T.Uemoto, and T.Ichimura, *A Proposal of Growing Process of Social Community by Altruism Behavior n Army Ant System - Its Application to Group Development Method of Open Souce Software-*, Proc. of the workshop of Computational Intelligence, The society of Instrument and Control Engineers(to appear in 2015)(Japanese).

[4] SourceForge.net, https://sourceforge.net/ (retrieved 2015/06/22)

[5] SourceForge Opendata, http://srda.cse.nd.edu/mediawiki/index.php/Making_Queries (retrieved 2015/06/22)

[6] T.Ichimura, T.Uemoto, A.Hara, *Emergence of Altruism Behavior for Multi Feeding Areas in Army Ant Social Evolutionary System*, Proc. of 2014 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2014), pp.176-181(2014).

[7] T.Ichimura, T.Uemoto, A.Hara, K.J.Mackin, *Emergence of Altruism Behavior in Army Ant Based Social Evolutionary System*, Springer-Plus, A Springer Open Journal, Vol.3, 712, doi:10.1186/2193-1801-3-712(2014).

[8] Y.Gao. G.Madey, V.Freeh, *Modeling and Simulation of the Open Source Software Community*, Agent-Directed Simulation Conference(2005).

[9] A.L.Barabasi, R.Albert, *Emergence of scaling in random networks*, Science 286, pp509-512(1999).

[10] Trivers RL, *The evolution of reciprocal altruism.* ,Q Rev Biol, Vol.46, pp.35-67(1971)