INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

CO-EVOLUTION OF PRODUCTS & COMMUNITIES IN MASS COLLABORATIVE PRODUCT DEVELOPMENT – A COMPUTATIONAL EXPLORATION

Jitesh H. Panchal
Washington State University, Pullman, WA, USA

ABSTRACT

Mass collaborative product development (MCPD) refers to a paradigm where large numbers of people collaborate globally as communities to develop products and services. It is based on bottom-up evolution as opposed to traditional top-down decomposition. The evolution of products developed through MCPD processes is coupled with the evolution of associated communities. In this paper, the co-evolution of products and communities is modeled as a complex self-organizing system. The proposed approach is based on agent-based modeling and Social Network Analysis (SNA). It allows the modeling of participant behavior within MCPD scenarios and captures the information about i) products as modules and their interdependencies, and ii) participants as decision-making agents. The evolving community is modeled as a weighted directed graph and analyzed using SNA metrics. The following aspects of the product development processes are studied: evolution of modules, evolution of number of participants associated with modules, community structures, effort distribution, distribution of the number of collaborating partners, and the effect of product dependencies on the community structure.

Keywords: Mass Collaboration, Product Development, Agent-based Modeling, Evolution, Social Network Analysis

1. INTRODUCTION: CO-EVOLUTION IN MASS COLLABORATIVE PRODUCT DEVELOPMENT

Mass collaborative product development (MCPD) refers to a paradigm where large numbers of individuals work together in the form of loose networks of peers to produce goods and services [1]. It is different from traditional product development processes where individuals organized in hierarchical teams collaborate with each other on well defined tasks that are aligned to achieve the overall organizational goals. Examples of mass collaborative product development efforts include Wikipedia [2], Linux [3], Mozilla Firefox [4], and Apache [5]. Examples of physical product development projects that utilize these concepts are open source car (OScar) [6] and Open Prosthetics [7]. In the open source car project, the goal is to develop a car using open source principles. In the open prosthetics project, the objective is to share CAD models of prosthetic devices as open source designs, which can be further developed and refined by others.

The differences between mass collaborative product development and traditional product development are detailed in [8]. Traditional product development processes are top-down decomposition-based processes whereas mass collaborative product development processes are driven by the principles of evolution and self-organization. Traditional processes are based on the decomposition of the overall function of the product into sub-functions, and developing subsystems for satisfying individual sub-functions. The subsystems are designed by teams through well-defined tasks assigned to hierarchically structured teams. The participants work together on well-defined tasks towards achieving the common goals of the organization. In contrast, the organizational (community) structures in MCPD are relatively flat and consist of independent participants working on different aspects of the project according to their expertise and interests. Instead of assignment of tasks, the participants pick up tasks that they would like to work on. Due to the fundamental differences between the two types of product development processes, the factors that affect the design processes are very different. Panchal and Fathianathan [9] discuss various research issues related to engineering design where progress is necessary for further development of this area, including *a)* product realization processes, *b)* coordination between stakeholders, *c)* structure of product realization teams, *d)* collective learning and evolution, *e)* incentives to participants, *f)* product architectures, *g)* product co-design, *h)* product-service systems, and *i)* information and computation.

The importance of the alignment of product architecture and organizational structure is well known in the product development literature [10]. It is particularly important in MCPD processes due to the highly interrelated nature of the community and product evolution. Since no organizational structure is imposed at the start of the process, the structure evolves as more participants join the product development effort and collaborate with existing participants. According to Conway [11], "any organization that designs a system (defined more broadly than just information systems) will inevitably produce a design whose structure is a copy of the organization's communication structure". This illustrates the dependence of the product structure on the organizational structure. At the same time, the communication between different participants is based on the product structure and is driven by the dependencies between subsystems, implying the effect of product structure on organizational structure. Hence, the systems (products) and the organizational structures are highly interdependent in nature. This interrelated nature is particularly important in mass collaborative projects due to its evolutionary nature – the product evolves over time based on the contributions of the participants and the community structure evolves as more participants join the efforts. The co-evolution of the product and the community structures result in the self-organizing nature of MCPD processes.

Existing research on organizational structures in mass collaborative processes is mainly carried out in the domain of open source software (OSS) development. Various researchers have presented empirical studies based on the data from existing OSS projects. Different types of organizational structures have been shown to emerge in different projects. For example, the community structure of Linux project is represented as a pyramid, the community structure of BSD project is represented as concentric circles, etc. [12]. Crowston and Howiston [13] study the community structure of OSS projects using metrics for centralization and hierarchy. In a related effort, Valverde and Sole [14] study the hierarchy of OSS social networks. Xu and Madey [15] present a quantitative study of the interactions of the developer community at SourceForge and present the topological and evolutionary statistics for OSS social network. Madey and co-authors [16] present an agent-based model to investigate OSS communities. Weiss and co-authors [17] use the OSS mailing lists to study the community structure and its evolution. The authors also discuss the effect of project dependencies on information flow between participants. Ye et al. [18] present the interrelationship between the evolution of software systems and the communities.

Despite the studies in the open source domain, the fundamental dynamics governing the co-evolution of product and community structure is not completely known. Prior work is mainly focused on empirical studies of existing projects. As highlighted by Panchal [8] mass collaborative product development efforts are significantly influenced by factors such as the initial product structure, participants' preferences, initial amount of work, etc. An understanding of the impact of these factors is important in gaining a holistic understanding of the dynamics of mass-collaborative processes. In this paper, a step towards the exploration of the interrelated nature of products and organizational structures in mass collaborative product development efforts is presented. The approach used in this paper is based on agent-based modeling [19, 20] to simulate the decisions of participants, their interactions with the product, and collaboration with other participants in a mass collaborative effort. The model is used to study the effect of various factors affecting the co-evolution of products and community structures. The community structures are characterized using metrics from Social Network Analysis [21, 22]. An overview of agent-based modeling and social network analysis are provided in Section 2. The details of the agent-based model are presented in Section 3. The results from the model for an illustrative scenario are presented in Section 4; and closing thoughts are presented in Section 5.

2. BACKGROUND OF APPROACHES USED IN THIS PAPER

2.1. Agent-based Modeling

Agent-based modeling is a technique used to simulate systems consisting of autonomous interacting entities called agents [19, 23-25]. These agents represent various different types of entities such as cells, plants and animals in biological simulations, atoms and molecules in chemical simulations, individuals and organizations in financial simulations, and vehicles in traffic simulations. Agents have their own behaviors (and goals). An agent acts based on the limited information available to it about its environment. The primary advantage of an agent-based model is that it allows the study of emergent behavior of complex systems in a bottom-up fashion. The agent-based modeling technique has gained significant popularity in social sciences [20, 23], traffic simulations, organizational science, and computational economics, supply chains, and stock markets. Agent-based modeling is a micro-simulation technique that is based on modeling individual agent behavior, as opposed to the characteristics of the entire set of agents in a macro simulation [24]. Various commercial and open source tools such as Swarm, Repast, and Netlogo are available for agent-based modeling. For more details, please refer to [19].

2.2. Social Network Analysis

Social Network Analysis (SNA) is the study of social relationships in the form of nodes and arcs [21, 22]. The nodes in a social network are individual actors, and arcs are relationships between the actors. For example, a network of friends can be modeled as nodes representing people and arcs representing friendship. Social networks can be either directed or undirected. The arcs may also carry weights to represent strengths of the relationships between actors. For example, in a friendship network, the strengths of friendships can be represented as weights on the arcs. SNA involves the study of the structure of social networks using various metrics (such as clustering, hierarchy etc. discussed in this section) which quantify different characteristics of the networks. Social network analysis has been used to study organizational structures in OSS literature [14, 16]. In this paper, various metrics from SNA are used to quantify the nature of organizational structures generated from the agent-based model presented in Section 3. Specifically, the following metrics are used in this paper: clustering, hierarchy, and centrality. The details of these metrics are provided in [22].

2.2.1. Clustering

The *density* of a network refers to the proportion of all possible ties that are actually present. In a weighted network, the density corresponds to the average strength of ties over all possible ties. The neighborhood of an actor refers to the set of all actors directly connected to that actor. Two types of clustering coefficients are defined – overall graph clustering and weighted graph clustering. The former is the average of the densities of the neighborhood of all the actors. The latter assigns a weight to the neighborhood densities proportional to the size of the neighborhoods.

2.2.2. Krackhardt's Dimensions of Hierarchy

Krackhardt [25] developed a set of metrics to measure the extent of hierarchy in a network. The metrics are based on the deviation from an ideal hierarchy – an *out-tree* graph, which is a directed graph in which all points are connected and all but one node has an in-degree of 1 (i.e., all actors in the graph have a single superior node). A network can deviate from this ideal hierarchy in four different dimensions. These dimensions are quantified by the following four measures:

- *Connectedness:* If all the actors are connected in a unitary structure, then the graph is more hierarchical. The connectedness measure is based on the ratio of the number of pairs that are reachable relative to the number of ordered pairs.
- *Hierarchy:* Reciprocal relations imply equal status (i.e., not hierarchical). Hence, the degree of deviation from pure hierarchy is measured by counting the relations that have reciprocated ties relative to the number of pairs with any tie.
- *Efficiency:* In a pure out-tree, each actor has a single boss. The amount of deviation from this is measured by counting the difference between actual number of links and the maximum possible number of ties (greater difference implies greater inefficiency).
- *Least upper bound:* In an ideal hierarchy, the command must be unified (i.e., each pair of actors should have an actor that directs ties to both). The deviation from that is measured as the number of pairs of actors that do not have a common boss relative to the number of pairs that could.

2.2.3. Centrality

The centrality metrics are based on the position of an actor within the network. An actor has power if it has a favored position and has more opportunities to share information and influence other actors. Actors with more ties have greater power due to their choices to communicate with other actors.

Three different types of centrality are defined: degree centrality, closeness centrality, and betweenness centrality. Actors that are more central to the structure have higher degree (i.e., connectedness) tend to have favored positions and hence, more power. This notion of centrality is referred to as degree centrality. Closeness centrality focuses on the distance from each actor to another. Betweenness centrality is associated with the notion of relations being central to the network. *Centrality* is a concept that applies to individual actors within a network whereas *centralization* measures how unequal the distribution of centrality within the network is. In this paper, degree centralization is used for quantifying the community structure in MCPD processes.
The metrics described in this section are used to model the communities resulting from the proposed agent-based model which is discussed next. The networks presented in this paper are analyzed using a software application called UCINet [26].

3. PROPOSED AGENT-BASED MODEL FOR STUDYING COEVOLUTION OF PRODUCTS AND COMMUNITY STRUCTURES

3.1. Product Model

The product model used in this paper is represented as a directed graph consisting of modules and dependencies between modules. An example of a product model is shown in Figure 1, which consists of nine modules (labeled from 0-8) and fifteen dependencies. Each module is associated with two key parameters: *percentage completion* and *growth rate*. The percentage completion denotes the extent to which a module is complete at a given point in time. The growth rate is used to quantify how fast a module grows in each cycle when worked upon by a single participant. It is expressed as a percentage increase in the module, and is calculated based on the percentage completion at that point in time. The default growth rate used in this paper is 1%. For example, if the percentage completion of a module is 50% and one participant works on it in one cycle, then the resulting percentage completion of the module is 50.5%.

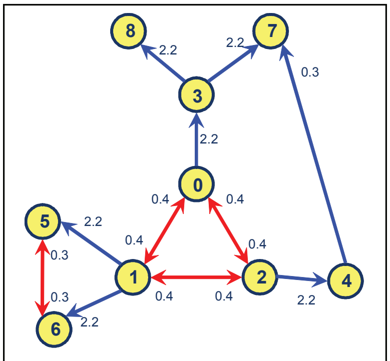


Figure 1 - Example product architecture used in this

Each of the dependencies between two modules is associated with a parameter called *dependency strength*. The strength of dependencies is modeled as the amount of rework required in the target module due to the changes in the originating module. For example, in Figure 1, the dependency from Module 0 to Module 3 is 2.2, which implies that if there is a change in Module 0 it results in a 2.2% rework in Module 3. Similar to the growth rate, the rework is also calculated based on the percentage completion of the target module at that time. Some of the module pairs (such as Modules 0 and 3) have unidirectional dependencies while others (such as Modules 0 and 1) have bi-directional dependencies implying that both the modules are coupled with each other. Note that the bi-directional dependencies are modeled as two separate dependencies. The dependency strength is modeled after the "rework impact" metric proposed by Cho and Eppinger [27, 28].
The product model shown in Figure 1 is used throughout the rest of the paper as an illustrative example of a product developed using MCPD process. This product model was used in prior work related to this topic. It consists of a set of *core modules* that strongly influence each other and a set of *external modules* that are relatively independent of each other but strongly depend on the core modules [29]. Modules 0, 1, and 2 are core modules whereas the other modules are the external modules. The dependencies between the core modules are bi-directional whereas the dependencies between core modules and external modules are unidirectional. The rationale behind choosing such a product structure is that it represents a wide range of products developed using mass collaborative processes. Further details are provided by Panchal [8].

3.2. Participants and their Decisions in the Proposed Model

The participants in a mass collaborative product development process are modeled as agents. Each participant is associated with a parameter called the *probability of participation*. There are three decisions that each participant makes during each cycle – a) deciding whether to contribute to the effort or not, b) deciding which module to contribute to, and c) deciding which participants to collaborate with. These decisions affect the speed and patterns in which products are developed and the structure of the communities developed around the product development effort.

3.2.1. Deciding Whether to Participate or Not

The parameter 'probability of participation' quantifies the probability that a particular participant contributes to the product development effort during a particular cycle. The probability of participation is dependent on various factors such as the cost associated with participation (which generally refers to the investment of effort by the participant), the benefit of participation (which can be due to a variety of factors such as personal satisfaction, recognition, etc.) Panchal [8] models the preferences of participants using two parameters - cost and value. The participation scenario is modeled using the game of involuntary altruism [30]. The decisions are modeled as the mixed equilibrium strategy of the involuntary altruism game. Based on the mixed equilibrium strategy, each participant determines a probability (α_N*) based on which he/she participates in the product development effort. The probability is given by: $\alpha_N^* = 1-(c/v)^{1/N-1}$ where c, v and N are the cost and the value and the number of participants respectively.
In this paper, instead of modeling the cost and benefits of participants separately, the decision model is simplified by directly introducing a parameter called the probability of participation (α_p). Further, as the product develops, more participants may see the value in the product being developed and may get interested in the mass collaborative effort. Hence, the probability of participation increases as the product develops. This is modeled by assuming that the probability of participation (α_p) of each participant grows at a rate proportional to the percentage completion of the entire product ($K_p.C_{percent}$), where K_p is a constant and $C_{percent}$ is the percentage completion of the project.

3.2.2. Deciding which Module to Contribute to

Having decided whether to contribute to the product development effort or not, the next decision is to choose the module to contribute to. The decision to join a particular module is governed by preferential attachment. While making this decision, the primary factor is the participants' experience on different modules. Participants generally contribute to modules that they are most familiar with. Hence, a participant determines the module to work on with a probability (α_m) that is proportional to the amount of contribution that he/she has made to a particular module $\alpha_m = K_m.E_{i,j}$ where K_m is a constant and $E_{i,j}$ is the participant i's effort on module j.
In addition to that, the participants are also likely to participate in the development of dependent modules. The dependencies between modules increase a person's likelihood of working on dependent modules. This is true in real mass collaborative efforts because of the changes in the dependent modules resulting from the changes in originating modules. This aspect is modeled in this paper by increasing the probability of contribution on that module by a factor that is proportional to the strength of dependency between a pair of modules. In other words, the dependency strengths in the product model are also used to model the probability of participation in related modules.
The decision is applicable to participants who have contributed before. However, the participants who have not contributed to any of the modules cannot use this decision criterion. Their preference for a particular module is proportional to the number of participants who have contributed to that module. This decision is based on the power-law distribution observed in the OSS literature. Xu and Madey [15] observe that developers sequentially choose more popular projects to join. The projects with high number of participants (i.e., the most popular projects) attract even more participants [17].

3.2.3. Deciding which Participant to Collaborate with

After deciding the module to work on, the last decision modeled in this paper is the decision to identify the specific participants to collaborate with. This decision is based on the observation in the open source literature that OSS developer-networks are scale free networks whose degree distribution follows a power law. Xu and Madey [15] discuss that the OSS network grows as the participants sequentially join the projects. Further, these networks are governed by preferential attachment, i.e., the probability of two nodes being connected is related to the node's degree. In this paper, the individuals decide to collaborate with specific participants with a probability (α_c) which is proportional to the target participants' contribution to that module, $\alpha_c = K_c.E_{i,j}$ where K_c is a constant and $E_{i,j}$ is the effort of participant i on module j chosen in Section 3.2.2.

3.2.4. Development of the Community

Having decided the participant to collaborate with, the participant creates a directed link with the collaborating partner. This link is called a ‘community link’. The network of participants connected via the community links represents the structure of the community. The community links are associated with a weight signifying the strength of that community link, which is quantified as the number of times two participants collaborate with each other. In other words, the community is modeled as a weighted directed graph. During their first collaboration, a community link is instantiated and the link strength is set to 1. If two participants have already collaborated with each other once, and they decide to collaborate again, the strength of the community link is increased by 1. To help the participants make these decisions, each participant records the amount of contribution made to each module and the community links with different people. Based on the connections between different people and the strengths of the links, the structure of the community is modeled using the SNA metrics described in Section 2.2.

3.3. Initialization and Execution of the Model

The model is initialized by creating a specified number of total participants (*N*). Each participant is randomly assigned a probability of participation within a range of [0 P_{UB}] where P_{UB} is the upper bound of probability of participation. A uniform random distribution function is used to assign the probabilities of participation. After initializing the participants, a certain number of initial participants are assigned to each module. These participants represent the group of initially active participants. The initial number of participants can either be equal for all the modules or can be different to model different scenarios. The initial active participants are randomly assigned values for initial amount of contributions for corresponding modules. An active participant is connected with all the other active participants associated with the corresponding modules through community links. This represents the fact that the initial participants are closely connected with each other. In other words, the network representing the community of initial participants associated with a module is completely connected.

After the model is initialized, it is executed in cycles. Each cycle represents one unit of time. During each cycle, all participants make the three decisions described in the previous section, i.e., each participant decides whether to contribute or not based on his/her probability function. If the decision is in favor of contribution, he/she selects the module to contribute to, and then decides the participant to collaborate with. Based on the contribution of the participants to different modules, the entire product evolves and through the collaboration between different participants, the community also evolves with time.

In the following section, one instantiation of the model is executed and the representative results from the model are presented. The initialization parameters for the results presented in the following section are shown in Table 1.

Table 1 - Values of initialization parameters used

Parameter	Values of the parameters used
Total number of participants	1000
Growth rate of modules	1%
Initial number of participants working on each module	4
Constant K_p for probability of participation	0.00005
Decision Parameters *(K_m, K_c)*	1.5 each
Initial upper bound of participation probability (P_{UB})	0.1
Initial work performed by initial participants	1%

4. RESULTS FROM THE EXECUTION OF THE MODEL

The model is executed for the product model shown in Figure 1 and the parameter values in Table 1. The evolution of modules and the number of participants associated with different modules are discussed in Section 4.1. The community structure and the effort distribution within the community are presented in Section 4.2. The effect of product dependencies on the community structure is illustrated in Section 4.3.

4.1. Evolution of Modules and Number of Participants Associated with Different

Modules

The evolution of modules over time is shown in Figure 2. In the figure, the evolution of Modules 0, 1 & 2 (which are the core modules), Modules 5 & 6 (that are dependent on the core modules) and Modules 7 & 8 are shown. Modules 3 and 4 are not shown for the sake of clarity of the figure. Note that the Modules 6 and 7 have the highest evolution rates whereas the evolution of the core modules (0, 1 & 2) is the slowest. It is clear that the rate of evolution is dependent on the dependencies between modules. In an earlier publication, Panchal [8] considered the effect of dependencies on module evolution. The assumption in the model by Panchal [8] is that only one person works on a module during a given cycle. Since only one person works at a time on a module, the evolution is only governed by the module dependencies. To simulate the effect of the collaboration between participants within a community, that assumption is relaxed in the current paper. Hence, the rate of evolution of modules is also dependent on the number of participants working on those modules.

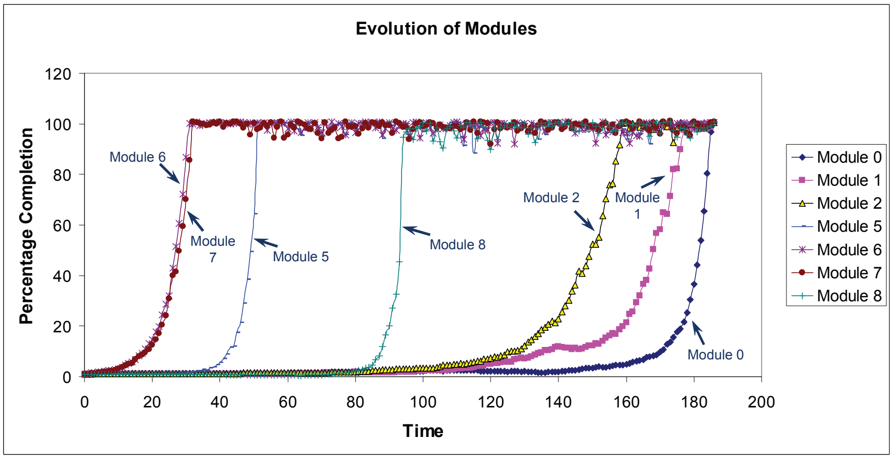


Figure 2 - Evolution of modules

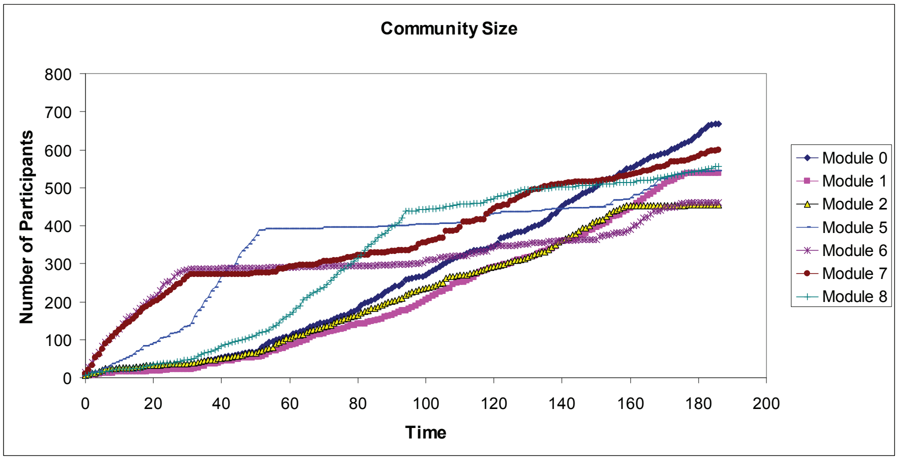


Figure 3 - Growth of number of participants associated with individual modules

The number of participants associated with different modules for this simulation run is shown in Figure 3. Note that the number of participants grows the fastest for Modules 6 and 7. This steep rise in the number of participants associated with this module is due to the decision rule that new participants have a higher probability of working on the modules that already have higher number of participants (see Section 3.2.2). After a module has grown sufficiently, the number of new participants becomes almost constant. For example, after 25 time steps, the number of people joining the network for Modules 6 and 7 stabilizes. Note that due to the dependencies of Modules 6 and 7 with other modules, there is significant amount of rework required as the other modules are developed. There is a slow growth in the community even after 25 time steps due to the rework. The evolution of core modules (0, 1, and 2) is slow because of two reasons – high coupling between modules and low number of participants.

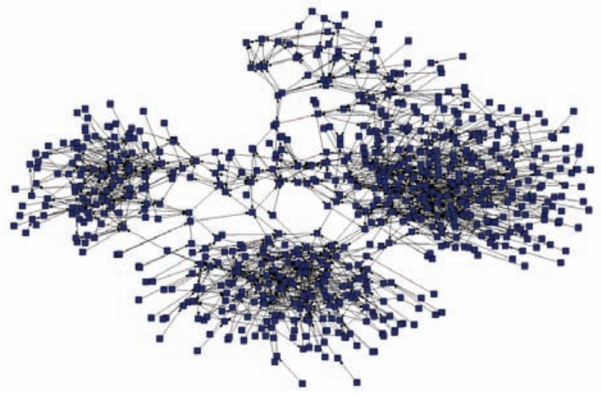


Figure 4 - Community structure (t = 25)

4.2. Community Structures

As participants collaborate with other participants on developing the modules, the community grows around the product development effort. As the product development proceeds, different clusters start emerging within the community. The evolution of these clusters is based on the interdependencies between modules. A sample network structure at time step equal to 25 is shown in Figure 4. Various clearly identifiable clusters are observed in the network structure. The interdependent modules have community structures that are more closely connected whereas the modules with weaker dependencies have clearly identifiable participant clusters. This is due to the decision rule involving collaboration with participants on related modules. The structure of this network can be characterized in terms of the SNA metrics presented in Section 2.2. The values of these metrics for the diagram shown in Figure 4 are listed in Table 2.

Table 2 – Values of the SNA metrics for the community structure shown above

Metric	Value
Connectedness	0.5025
Hierarchy	0.6374
Efficiency	0.9971
Least Upper Bound (LUB)	0.7824
Overall Graph Clustering Coefficient	0.035
Weighted Clustering Coefficient for the Graph	0.033
Network Centralization	0.81%

The distribution of the amount of effort invested in the mass collaborative effort at time step equal to 25 is shown in Figure 5. Note that the participants' effort follows an exponential distribution with very few participants investing greater amount of effort and significantly greater number of participants investing small effort. Similarly, the number of connections with collaborators is also exponentially distributed as shown in Figure 6. Very few participants have large number of collaborations with other participants and a large number of participants have only few collaborators. This trend is analogous to the one found in the open source literature [31].

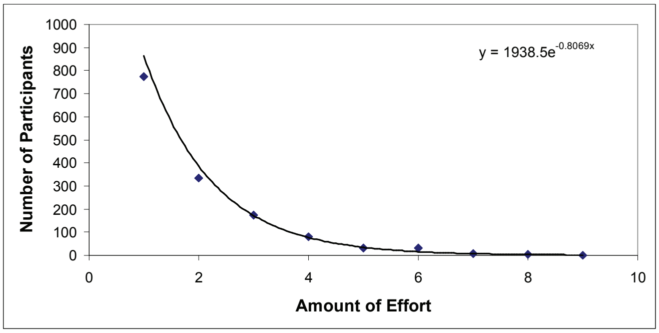


Figure 5 - Distribution of Effort (t=25)

Table 3 – SNA metrics for five different runs

Run #	Connectedness	Hierarchy	Efficiency	LUB	Clustering	Weighted Clustering	Centralization
1	0.5025	0.6374	0.9971	0.7824	0.035	0.033	0.81%
2	0.4968	0.6504	0.9971	0.7693	0.041	0.032	1.56%
3	0.4982	0.7204	0.9973	0.6937	0.040	0.034	1.01%
4	0.4870	0.6902	0.9971	0.7240	0.038	0.031	1.16%
5	0.4787	0.6742	0.9972	0.7436	0.043	0.037	0.91%

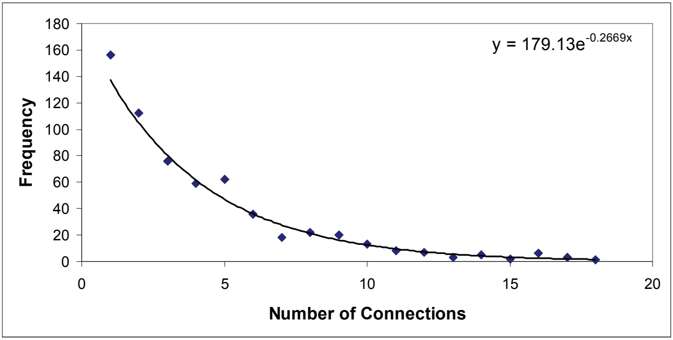


Figure 6 - Distribution of number of connections (t=25)

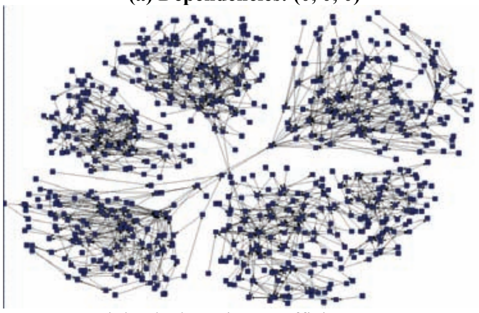
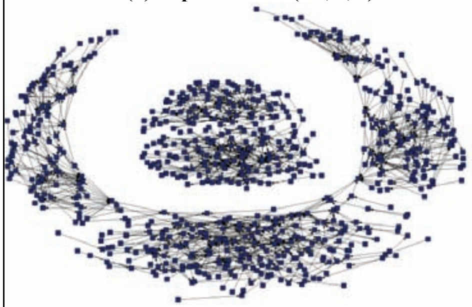
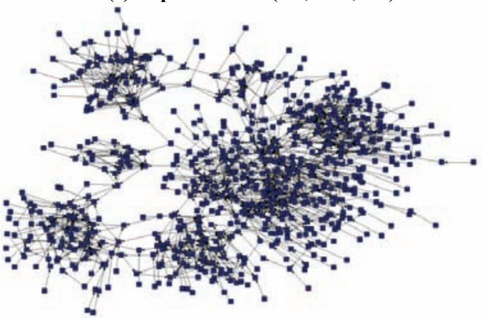
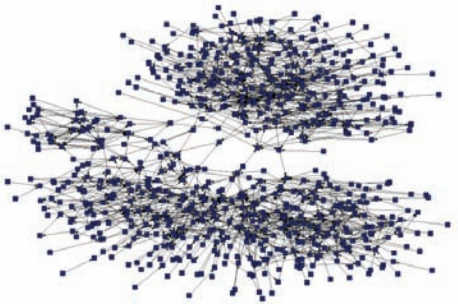
It is important to note that the results shown in Table 2 are from a single execution of the model. The model is inherently stochastic due to the random initialization of parameters such as amount of initial effort and participants' preference, and due to the decisions made during the process using probabilistic rules. This results in different model outputs for each execution of the model. The SNA metrics for five different runs are shown in Table 3. The variation in the results is significant due to the fact that the model is based on positive feedback which amplifies the parameters such as the number of participants working on different modules. This highlights the complex dynamical nature of mass collaborative processes. In order to determine the conditions under which targeted evolution can take place, there is a need to determine robust conditions under which evolution can take place.

4.3. Effect of Product Dependencies on the Community Structure

As mentioned earlier in the paper, the product dependencies have a significant effect on the community structure. In order to explore the effect of these dependencies on the community structure, four different combinations of the dependencies between core modules (Modules 0, 1 & 2), dependencies from core modules to dependent modules (Modules 3, 4, 5 & 6) and the dependencies between dependent modules (Modules 7 & 8) are chosen. The model is executed for these dependencies and the corresponding network structures for time step 25 are shown in Table 4. The dependencies for these

four combinations are listed in the table. For example, for the result set (a), dependencies are (0,0,0) which means that all the modules are independent of each other. It is observed in the network diagram that there are a large number of independent components within the network. In the second set, the dependencies are (0.3, 0, 0) which means that only the core modules are dependent on each other. All the other modules are independent. Similarly, the dependencies in results (c) and (d) are (0.3, 0.15, 0.5) and (0.3, 0.3, 0.15) respectively.
The community structures are characterized in terms of the weighted clustering coefficient and the hierarchy metrics. The weighed clustering coefficient decreases monotonically from 0.056 to 0.038 as we go from case (a) to case (d). Similarly, the hierarchy metric also reduces monotonically from 0.7162 to 0.6435. This shows the effect of reducing the module dependencies on the hierarchical nature of the communities.

Table 4 – Network representations of communities emerging for different dependencies between modules

(a) Dependencies: (0, 0, 0)	**(b) Dependencies: (0.3, 0, 0)**
 Weighted Clustering Coefficient: 0.056 Hierarchy: 0.7162	 Weighted Clustering Coefficient: 0.046 Hierarchy: 0.6916
(c) Dependencies: (0.3, 0.15, 0.5)	**(d) Dependencies: (0.3, 0.3, 0.15)**
 Weighted Clustering Coefficient: 0.042 Hierarchy: 0.6643	 Weighted Clustering Coefficient: 0.038 Hierarchy: 0.6435

5. CLOSING THOUGHTS

In this paper, an agent-based model for understanding the interrelated nature of product evolution and community structure is presented. The model consists of two types of entities – the products and the participants. The products are modeled as directed graphs with modules and interrelationships between them. The participants are modeled as agents with decision rules. As the participants work on developing the product, they collaborate and develop associations with each other, thereby resulting in a community structure. The community structure is characterized using a weighted directed graph where the weights represent the strength of association among different participants. The structure of the community is modeled using Social Network Analysis metrics. The model is executed and the results are shown in this paper for a simple product structure.

Most of the prior work in the domain of MCPD processes is based on empirical study of successful open source software development projects. Even in these empirical studies, the relationship between product structure and organizational structure has not been sufficiently explored. The primary contribution of this paper is that it presents a step towards understanding the interdependent nature of product structure and organizational structure as emphasized by the Conway's law [11]. It is shown that the evolution of products is dependent on various factors including the participants and their collaborations. This study is important from the standpoint of understanding the underlying mechanisms of mass collaborative processes. Without this understanding, it is difficult to replicate the success of such processes. Agent-based modeling presents a promising approach to studying the highly coupled dynamics of the product evolution in mass collaborative processes. According to Tefstein [32] , an agent-based model can be used for various purposes. In this paper, the purpose is to gain insights into mass collaborative processes and the factors that affect product and community evolution. The objective is not to exactly simulate all the details of a specific mass collaborative project. The model complements the empirically based efforts to understand mass collaborative processes particularly in the domain of open source software development.
This work is an initial step in the direction of gaining complete understanding of mass collaborative processes. Significant amount of work remains to be done in this area. Specifically, the current model focuses on the growth of a fixed number of modules. However, in a real world scenario, the product structure itself may evolve, resulting in an increase in the number of modules over time. Further, a phenomenon called 'forking' is common in the open source software development, which means that at a point in time, developers take an instance of the product (software code) and start independent development on it, creating a distinct piece of product (software). Two separate product development efforts can be carried out in parallel. Future work would involve simulating the forking phenomenon. Finally, it is important to validate the model using data from a real world product development scenario.

REFERENCES

[1] Tapscott, D. and Williams, A. D. *Wikinomics: How Mass Collaboration Changes Everything*, 2006, Penguin Group (USA).

[2] Wikipedia - The Free Encyclopedia. 2008, [cited 2008, January 23]; Web Link: http://en.wikipedia.org/wiki/Main_Page.

[3] Linux, *Linux Online - About the Linux Operating System.* 2008, [cited 2008, January 23]; Web Link: http://www.linux.org/info/index.html.

[4] Mozilla. 2008, [cited 2008, January 23]; Web Link: http://www.mozilla.org/about/.

[5] Apache Software Foundation, *The Apache HTTP Server Project.* 2008, [cited 2008, January 23]; Web Link: http://httpd.apache.org/.

[6] Oscar Project, *Oscar: Reinvent Mobility.* 2008, [cited 8 February 2008]; Web Link: http://www.theoscarproject.org/.

[7] Open Prosthetics *The Open Prosthetics Project: An Initiative of the Shared Design Alliance.* 2008, [cited April 11, 2008]; Web Link: http://openprosthetics.org/.

[8] Panchal, J. H. Agent-based Modeling of Mass Collaborative Product Development Processes. *Journal of Computing and Information Science in Engineering, in press*, 2009.

[9] Panchal, J. H. and Fathianathan, M. Product Realization in the Age of Mass Collaboration. In *ASME Design Automation Conference*, New York City, NY, USA, 2008. Paper Number: DETC2008-49865.

[10] Sosa, M. E., Eppinger, S. D., and Rowles, C. M. The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. *Management Science*, 2004, 50(12), pp. 1674-1689.

[11] Conway, M. E. How do Committees Invent. *Datamation*, 1968, 14(5), pp. 28-31.

[12] Weber, S. *The Success of Open Source*, 2004, Harvard University Press.

[13] Crowston, K. and Howison, J. Hierarchy and Centralization in Free and Open Source Software Team Communications. *Knowledge, Technology, and Policy*, 2006, 18(4), pp. 65-85.

[14] Valverde, S. and Solé, R. V. Self-Organization Versus Hierarchy in Open-Source Social Networks. *Physical Review E*, 2007, 76(4), pp. (046118)1-8.

[15] Xu, J. and Madey, G. *Exploration of the Open Source Software Community*, *NAACSOS* 2004, Pittsburgh, PA.

[16] Madey, G., Freeh, V., and Tynan, R. Modeling the F/OSS Community: A Quantitative Investigation. In *Free/Open Source Software Development*, 2004, (S. Koch, Editor), (Idea Publishing).
[17] Weiss, M., Moroiu, G., and Zhao, P. Evolution of Open Source Communities. In *Proceedings of the International Conference on Open Source Systems*, Springer, 2006, pp. 21-32.
[18] Ye, Y., Nakajoki, K., Yamamoto, Y., and Kishida, K. The Co-Evolution of Systems and Communities in Free and Open Source Software Development. In *Free/Open Source Software Development*, 2005, (S. Koch, Editor), (IGI Publishing, Hershey, PA), pp. 59-82.
[19] Bonabeau, E. Agent-based Modeling: Methods and Techniques for Simulating Human Systems. *Proceedings of National Academy of Sciences*, 2002, 99(3), pp. 7280-7287.
[20] Axelrod, R. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*, 1997, Princeton, NJ, Princeton University Press.
[21] Wasserman, S., and Katherine F. *Social Network Analysis: Methods and Applications*, 1994, Cambridge, Cambridge University Press.
[22] Hanneman, R. A. and Riddle, M. *Introduction to Social Network Methods.* 2005, [cited 2008 January 12]; Web Link: http://faculty.ucr.edu/~hanneman/
[23] Epstein, J. M. and Axtell, R. *Growing Artificial Societies: Social Science from Bottom-Up*, 1996, Cambridge, MA, The MIT Press.
[24] Davidsson, P. Multi Agent Based Simulation: Beyond Social Simulation. In *Multi Agent Based Simulation (LNCS Vol. 1979)*, 2000 (Springer Verlag).
[25] Krackhardt, D. Graph Theoretical Dimensions of Informal Organizations. In *Computational Organization Theory* 1994, (L. Erlbaum Associates Inc., Hillsdale, NJ, USA), pp. 89 - 111.
[26] Borgatti, S. P., Everett, M.G. and Freeman, L.C. *Ucinet for Windows: Software for Social Network Analysis*, 2002, Analytic Technologies, Harvard, MA.
[27] Cho, S.-H. and Eppinger, S. D. A Simulation-Based Process Model for Managing Complex Design Projects. *IEEE Transactions in Engineering Management*, 2005, 52(3), pp. 316-328.
[28] Cho, S.-H. and Eppinger, S. D. Product Development Process Modeling Using Advanced Simulation. In *ASME 2001 Design Engineering Technical Conferences - Design Theory and Methodology Conference*, Pittsburgh, Pennsylvania, 2001, pp. Paper Number: DETC2001/DTM-21691.
[29] Leadbeater, C. *We-think: The Power of Mass Creativity*, 2008, Profile Books Ltd.
[30] Baldwin, C. Y. and Clark, K. B. The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science*, 2006, 52(7), pp. 1116-1127.
[31] Healy, K. and Schussman, A. The Ecology of Open Source Software Development. In *Paper presented at the annual meeting of the American Sociological Association*, Atlanta, GA, 2003.
[32] Tesfatsion, L. Agent-based Computational Economics: A Constructive Approach to Economic Theory. In *Handbook of Computational Economics, Volume 2*, 2006, (L. Tesfatsion and K.L. Judd, Editors), (Elsevier), pp. 831-880.

Contact: Jitesh H. Panchal
Washington State University
School of Mechanical and Materials Engineering
Pullman, WA 99164 USA
Phone: +1-509-715-9241; Fax: +1-509-335-4662; E-mail: panchal@wsu.edu
URL: http://www.mme.wsu.edu/people/faculty/faculty.html?panchal

Jitesh H. Panchal is an Assistant Professor in the School of Mechanical and Materials Engineering at Washington State University. He received his B.Tech. from IIT Guwahati (India), and MS and PhD in Mechanical Engineering from Georgia Institute of Technology, Atlanta. His research interests are in the field of mass-collaborative design and multilevel design. He is a member of ASME and ASEE.