Agent-Based Modeling of Mass-Collaborative Product Development Processes

Jitesh H. Panchal
School of Mechanical and Materials Engineering
Washington State University, Pullman, WA 99164
E-mail: panchal@wsu.edu

Abstract
Mass collaborative product development refers to a paradigm where large groups of people compete and collaborate globally to develop new products and services. In contrast to the traditional top-down decomposition based design processes, the primary mechanism in mass collaborative product development is bottom-up evolution. Hence, the issues underlying mass collaborative processes are fundamentally different from those in traditional design processes. For example, instead of determining the best sequence in which activities should be carried out, the emphasis is on developing the right conditions under which product evolution can be fostered. Existing research on product development is primarily focused on top-down design processes. The evolutionary nature of mass-collaborative product development has received very little attention. Specifically, computational models for these processes have not been developed.

In this paper, a step towards understanding the fundamental processes underlying mass collaborative product development using a computational model is presented. The model presented in this paper is based on an agent-based modeling approach, which allows the modeling of the behavior of different entities within a product development scenario and the study of the effect of their interactions. The model captures the information about i) products as modules and their interdependencies, and ii) participants involved and their strategies. The benefits of the agent-based model in understanding mass collaborative product development are shown using a simple product model. The following aspects of the product development processes are studied: a) the rate of evolution of the individual modules and the entire product, b) product evolution patterns and the effect of the number of participants, c) the effect of prior work on product evolution, d) the evolution and distribution of participants, and e) the effect of participant incentives. The agent-based modeling approach is shown as a promising approach for understanding the evolutionary nature of mass collaborative product development processes.

Keywords: Mass collaboration, Agent-based modeling, Design processes, Product evolution, Game theory

Word Count: 8949 (excluding references and abstract)
1. Frame of Reference: Mass Collaborative Product Development

1.1. Overview of Mass Collaborative Product Development

Mass collaborative product development (MCPD) is an emerging paradigm of community-led innovation where masses of individuals work together in the form of loose networks of peers to produce goods and services [1]. Mass collaboration involves the collective action of large numbers of individuals. The individuals participating in MCPD projects are not organized as hierarchical teams. This is in stark contrast to the traditional product development where individuals organized in hierarchical teams collaborate with each other on well-defined tasks that are aligned to achieve the overall organizational goals. A well-known example of a product developed through mass collaboration is Wikipedia [2], which is developed by over 10 million volunteers collaborating over the Internet to create an encyclopedia which consists of about 9.5 million articles. Such collaboration is also evident in the open source software development projects such as Linux [3], Mozilla Firefox [4], and Apache [5]. A list of other examples of different types of projects that utilize the mass collaboration paradigm is presented by Panchal and Fathianathan [6]. The common success factor for all these projects is that a large number of individuals with diverse expertise and varied interests work together on different aspects of the problem.

The technological driving force behind mass collaboration is the ability to share information easily and inexpensively over the Internet. Hence, it is not surprising that the paradigm has emerged in information-based products such as software and encyclopedias. The use of mass collaboration is not only limited to information-based projects, however. It has also started to emerge in the physical product development domain. According to von Hippel [7], “physical products are information products during the design stage”. Examples of physical product development projects that utilize these concepts are open source car (OScar) [8] and Open Prosthetics [9]. 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. It is believed that the paradigm of mass collaboration in product development has a potential to play a significant role in socially relevant design [10]. One of the initiatives towards an online platform for mass collaborative design is Thinkcycle, which is intended for collaborative solution of problems in diverse domains such

Despite the encouraging success of various projects listed above, the utilization of mass collaboration in product development is still in its infancy. There are various technical, legal, economic, and social challenges associated with its implementation in practical settings. Significant research efforts are being devoted in the management and business research community [13-15] to understand how the intelligence of masses of individuals can be harnessed in product development. Related efforts include research on collective intelligence [16] and collective innovation [17]. In the domain of software development, efforts including [18-21] have been devoted to understanding the open source software development which is a great example of mass collaborative effort. Deshpande and Reihle [22] quantitatively show that the overall growth of open source software development is exponential. Other aspects such as social issues [23], legal and intellectual property issues [24] are also being studied.

From the standpoint of engineering design, Panchal and Fathianathan [6] discuss how the paradigm of mass collaboration applies to product development. The authors present an overview of the open research issues in the emerging field of MCPD. The authors identify various research areas related to engineering design where progress is necessary for further development of this area, including a) product realization processes, b) coordination between stakeholders, c) nature 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. In this paper, the focus is on addressing the process aspect of MCPD. The fundamental question in MCPD is: “How can mass collaboration be utilized effectively for product development?” To successfully answer this question, it is important to understand the factors that influence product development in a mass collaborative environment. To that end, a computational model is presented in this paper to study the effect of different factors affecting product evolution in MCPD processes. The model is developed using the agent-based modeling [25, 26] approach which is used for simulating the actions and interactions of autonomous agents in a network. The details of the model are presented in Section 2. The model is executed for an example product design problem. The results of the model execution are discussed in two sections – the evolution of the product is discussed in Section 3 and the participant related aspects are discussed in Section 4. Finally, closing comments are provided in Section 5. Before presenting
the model in detail, a comparison between traditional hierarchical design processes and mass collaborative processes is presented in Section 1.2.

1.2. Top-Down Decomposition-based Product Development Processes vs. Bottom-Up Evolutionary Processes

The traditional design processes are described as top-down processes where the information flows logically from the desired functionality of a product to a design that satisfies the functionality [27]. These processes are based on the assumption that the overall function of the product can be decomposed into sub-functions, and these sub-functions can be satisfied through the specification of sub-systems. The subsystems are designed by teams through well-defined tasks. The requirements flow from the highest level system to the lower level components and the design information systematically flows from the lower level components to the higher level systems. After all the subsystems and the components are specified, integration and verification are carried out to ensure that the system level requirements are satisfied, and there are no undesired interactions between the subsystems. The top-down decomposition-based view of the process is accurately captured by the systems engineering Vee model [28].

The traditional product development processes are different from mass collaborative design processes in a variety of ways as summarized in Table 1. First, the organizational structure in traditional product development is hierarchical whereas in the MCPD, there is no well defined organizational structure. The product development efforts are carried out by loose networks of peers who are individually motivated to participate in the project. The organizational structure in mass collaborative projects is flat and it consists of independent participants who work on different aspects of project that they are interested in. A core group of participants, referred to as ‘catalysts’ by Brafman and Beckstrom [29] initiate the project and provide a platform for the other participants to contribute. In mass collaborative projects, tasks are not assigned to participants. On the other hand, the participants in a traditional product development team work together on well defined tasks towards achieving the common goals of the organization. The sequence of execution of tasks is based on pre-specified and well-defined project plans. In a traditional scenario, the product is systematically designed to meet the performance targets. The emphasis of the team throughout the project is on the satisfaction of all the constraints and meeting the goals as closely as possible.
The goals are achieved by assignment of tasks to the right experts in the team and the timeline is tightly controlled. On the other hand, in a mass collaborative scenario the participants’ contributions are not based on the pre-specified tasks and the product evolves over time based on the contributions of the participants. The emphasis of the core team throughout the project is on creating right conditions for product evolution, and providing the right incentives for the participants to contribute. If the right initiatives are available, the participants would contribute to the projects on their own. Instead of assignment of tasks, the participants pick up tasks that they would like to work on. Generally, no deadlines are specified.

<table>
<thead>
<tr>
<th></th>
<th>Traditional Product Development</th>
<th>Mass Collaborative Product Development</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Organizational Structure</strong></td>
<td>Hierarchical</td>
<td>Loose network of peers</td>
</tr>
<tr>
<td><strong>Participants</strong></td>
<td>Guided by higher level organizational goals</td>
<td>Self-interested</td>
</tr>
<tr>
<td><strong>Product</strong></td>
<td>Systematically designed in a targeted manner</td>
<td>Product evolves over time</td>
</tr>
<tr>
<td><strong>Team focus</strong></td>
<td>Enforcement of objectives and constraints</td>
<td>Provision of right conditions for product evolution</td>
</tr>
<tr>
<td><strong>Task assignment</strong></td>
<td>Tasks assigned to participants</td>
<td>Participants choose to work on tasks on their own</td>
</tr>
<tr>
<td><strong>Timeline</strong></td>
<td>Tightly controlled</td>
<td>Generally no target dates</td>
</tr>
<tr>
<td><strong>Project initiation</strong></td>
<td>Project initiated after careful market analysis</td>
<td>Project initiated by a small group of participants</td>
</tr>
<tr>
<td><strong>Decision-making</strong></td>
<td>Top-down hierarchical decision-making</td>
<td>Decentralized decision making</td>
</tr>
<tr>
<td><strong>User involvement</strong></td>
<td>Low user involvement. Product development carried out by a dedicated team</td>
<td>Generally no distinction between users and developers</td>
</tr>
<tr>
<td><strong>Product release</strong></td>
<td>Product released after major revisions</td>
<td>Product always under continuous development</td>
</tr>
</tbody>
</table>

In the former scenario, the project is initiated after a careful market survey and identification of users’ needs. However, in the latter scenario, a systematic evaluation of market potential is rarely performed. The project is initiated by small groups of individuals with personal interests or needs. Other participants join in at later stages when they see value in participating in the efforts. Similarly, in the traditional processes, the decisions are made in a centralized manner whereas in mass collaborative processes, the decisions are made in a decentralized manner. The user involvement in traditional product development is limited to the earlier needs identification phases and later testing phases only. The user involvement during the product development is generally low. Recently, various organizations have started adopting the concept of lead-user innovation [30, 31] where the key users of the products are involved in the entire product development process. In contrast to that, the users themselves are the primary
product developers in MCPD scenarios. Finally, different versions of the product are released after major revisions in traditional product development whereas in MCPD, the product is always in the state of continuous evolution [32, 33].

Due to the fundamental differences between the two types of product development processes, the factors that affect the design processes are very different. Top-down product development processes have been studied extensively in the engineering design literature. The efforts range from modeling design processes to optimizing them. Examples of approaches on modeling design processes include IDEF [34], PERT and GANTT charts [35], design structure matrices [36]. Design processes have been analyzed as networks of tasks that need to be sequenced to minimize factors such as cost, time, coupling, rework, and the effect of uncertainties; and to maximize factors such as concurrency, and utilization of available resources. Some of the key questions addressed in the literature are: 
a) how should product development tasks be decomposed? 
b) how can concurrency be achieved? 
c) how should tasks be carried out to minimize the overall product development time? 
d) what should the product architecture be to minimize interdependence between tasks? 
e) how should product development tasks be sequenced? 
f) how can downstream knowledge be utilized in the initial phases of product development processes? These aspects of product development processes are studied heavily in the areas of concurrent engineering [37], and multidisciplinary design optimization [38].

These efforts are limited to top-down design processes where the tasks necessary for developing a product are well known, and the information flow between tasks are also known. In contrast, the MCPD processes are characterized by evolution and self-organization. Godfrey and Tu [39] authors present a computational model for the evolution of open source software projects. Nakakoji and coauthors [32] study the evolution patterns in open source software development. Lonchamp [40] presents a model for open source software development processes. These efforts are focused on open source software development projects only. The research in evolutionary design processes has been significantly limited in the product development domain. Initial conceptual exploration of modeling product development processes as dynamical systems is carried out by Klein and co-authors [41]. Similarly, Bar-Yam provides a conceptual exposition of evolution in product development under the umbrella of “evolutionary engineering” [42].
Based on the existing literature, it has been identified that some of the factors that affect the product evolution in a mass collaborative scenario include – the number of participants, product architecture, dependencies between product modules, work carried out prior to opening up the product development process (prior work), participant contributions, incentives for participants to contribute to the project, etc. Although different factors have been conceptually discussed by different researchers, a systematic computational analysis of the effect of these factors in MCPD has not been carried out in the literature. In this paper the objective is to develop a model that allows the study of these factors in an integrated manner. The corresponding questions studied in this paper are listed in Table 2.

Table 2 - Questions addressed in this paper using the agent-based model

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Rate of product evolution: How does a product evolve in a mass collaborative scenario?</td>
</tr>
<tr>
<td>2. Patterns of product evolution: How does the number of participants affect product evolution and the total effort invested in product development?</td>
</tr>
<tr>
<td>3. Effect of prior work: How does prior work affect the product evolution in mass collaborative processes? How does prior work affect the total effort required for MCPD?</td>
</tr>
<tr>
<td>4. Effect of coupling strengths: What is the effect of interdependencies between product modules in mass collaborative projects?</td>
</tr>
<tr>
<td>5. Evolution of participation: How are individual contributions distributed in the team? How does the extent of individual contributions change over time? How does participation in a mass collaborative project change?</td>
</tr>
<tr>
<td>6. Effect of incentives: How do individual incentives affect product evolution?</td>
</tr>
</tbody>
</table>

In order to address these questions an agent-based computational model is presented. The model is an abstraction of real mass collaborative design processes. In its current form, the model helps in gaining an understanding of the evolutionary aspects of MCPD. The details of the model are presented in Section 2. The product evolution aspects (questions 1-5 in Table 2) are studied in Section 3. The participant related aspects (questions 6 and 7) are studied in Section 4. Closing thoughts are presented in Section 5.

2. Proposed Agent-Based Model for Mass Collaborative Product Development Processes

2.1. Agent-Based Modeling: An Overview

Agent-based modeling is a technique used to simulate systems consisting of autonomous interacting entities called agents [25, 43, 44]. 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 simulation 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 [26], traffic simulations, organizational science, and computational economics, supply chains, and stock markets. Agent-based modeling is a \textit{micro} simulation technique that is based on modeling individual agent behavior, as opposed to the characteristics of the entire set of agents in a \textit{macro} simulation [45].

According to Tesfatsion [46], the primary objectives of using agent-based models are to gain a) \textit{empirical} understanding of a system, i.e., to understand why certain system level behaviors are observed, b) \textit{normative} understanding of a system, i.e., to discover new designs, c) \textit{qualitative insight} and theory generation, i.e., to systematically examine the system behavior under different initial conditions, and d) \textit{methodological} advancement, i.e., to provide new tools and methods for rigorous study through computational experiments. In this paper, the role of agent-based modeling is to gain a qualitative insight into MCPD processes and to understand the impact of different factors that affect the evolution of the products in such product development scenarios. The objective is not to perfectly recreate a design environment but is used as a tool for examining and understanding emergence of patterns in mass collaborative scenarios.

\section*{2.2. The Proposed Model for Mass Collaborative Product Development Processes}

The proposed model consists to two main entities: the \textit{product} and the \textit{participants} involved in developing the product, discussed in Sections 2.2.1 and 2.2.2 respectively. The model parameters and outputs are discussed in Section 2.2.3 and the scope and assumptions of the model are discussed in Section 2.2.4.

\subsection*{2.2.1. Product model}

In the proposed model, a product is represented as a directed graph consisting of a set of modules as nodes with interdependencies between them. Each module has two attributes – \textit{percentage completion} which represents the extent to which the module has been developed, and \textit{growth rate} which quantifies how fast the module grows. Similarly, the interdependencies are also associated with one attribute – the \textit{dependency strength}. In the product development literature, the strengths of dependencies have been modeled in different ways [47-49]. Pimmler and Eppinger [49] describe the strength of coupling between parameters and tasks qualitatively using words such as required, desired, indifferent, undesired, and detrimental. \textit{Required} information flow is necessary for functionality,
desired information flow is beneficial, but not absolutely necessary for functionality, indifferent information exchange does not affect functionality, undesired and detrimental information exchange causes negative effects. Numerical values ranging between +2 and -2 are assigned to these qualitative terms. Such quantification is based on experience and human judgment. Smith and Eppinger [50] presented a deterministic rework metric to represent the repetition of a task that depends on another task. Smith and Eppinger [51] later introduced a metric called rework probability to represent scenarios where there is uncertainty whether changes in one module will result in changes in another module or not.

In this paper, we adopt a metric similar to the “rework impact” proposed by Cho and Eppinger [48]. The dependency strength is measured as the amount of rework required in the target module if there were any change in the originating module. Through the modules and interdependencies, the product model essentially captures the architecture of the product.

Figure 1 - A simple illustrative example of product model

A simple illustrative example of a product model with two modules and one dependency is presented in Figure 1. Module 1 is 50% complete and Module 2 is 30% complete. The dependency strength from Module 1 to Module 2 is ‘5’, which implies that any change in module 1 will result in a 5% rework in Module 2. Hence, after a change in Module 1, the updated % completion of Module 2 would be 28.5%.

2.2.2. Participants and their Strategies
The participants in the MCPD processes are modeled as agents. The number of participants in the model is represented by parameter \( N \). These participants make decisions based on their preferences about their benefits and costs. Each agent has some benefits in participating in the product development process, which may be due to a variety of reasons such as personal satisfaction, ability to use the product for personal use, ability to gain new knowledge, or the possibility of gaining recognition in the community. The benefits of participation have been studied in the open source software development context [20, 52, 53]. In this paper, all possible types of benefits are accounted for using a single parameter called value \((V_k)\) to a participant \( k \). In addition to the value to the participant,
each agent incurs some cost in participating in the product development process. The simplest factor resulting in participant cost is the personal time invested in the project. The cost to participant $k$ is also modeled using a single parameter ($C_k$) in the model.

Each participant also has a decision model that represents whether a participant decides to participate in the product development process or not. The model of decision-making by the participants used in this paper is based on the involuntary altruism model presented by Baldwin and Clark [54]. The involuntary altruism model is based on a game called “public provision of private goods” [55]. The game was first applied to open source software development by Johnson [56]. There are various similarities between MCPD and open source software development. Hence, the involuntary altruism game is used to model the strategy of the participants.

The normal form of the game of involuntary altruism, as presented by Baldwin and Clark in [54], for two participants $P_1$ and $P_2$ is shown in Table 3. Each participant has value ($v$) and cost ($c$) associated with a product development task, and has the option to decide whether to contribute to the effort or not. If both the players decide not to contribute, the task is not completed and the value to both of them is 0. If one of the participants decides to contribute and the other does not, then the non-contributing participant receives the entire value ($v$) and does not incur any cost. Hence, the resulting value to the non-contributing participant is $v$. Since the contributing participant also incurs the cost $c$, the resulting value for the contributing participant is $(v-c)$. There are two Nash equilibria for the game. The Nash equilibria correspond to the strategies where one participant works and the other does not. The mixed-strategy Nash equilibrium for the game is the strategy where each participant contributes with a probability of $\alpha^* = (1-c/v)$. If there are $N$ participants in the game, the mixed strategy equilibrium is the strategy where each participant contributes with a probability: $\alpha_{k}^* = (1-c/v)^{1/N-1}$

Further details of the game are provided in [54]. In the model presented in this paper, this mixed-strategy Nash equilibrium (where each participant contributes with a probability of $\alpha_{k}^*$) is used for modeling the participants’ decision making strategy.

Table 3 – Normal form of the game of Involuntary Altruism [54]

<table>
<thead>
<tr>
<th>$P_1$ Does not contribute</th>
<th>$P_2$ Does not contribute</th>
<th>$P_2$ Contributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$ Does not contribute</td>
<td>0, 0</td>
<td>$v$, $(v-c)$</td>
</tr>
<tr>
<td>$P_1$ Contributes</td>
<td>$(v-c)$, $v$</td>
<td>$(v-c)$, $(v-c)$</td>
</tr>
</tbody>
</table>
2.2.3. Model Parameters and Outputs

As a summary, the parameters of the model and the outputs are listed in Table 4.

Table 4 – Parameters in the agent-based model for MCPD

<table>
<thead>
<tr>
<th>Product Model</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Number of modules ($M$)</td>
<td>• Number of participants ($N$)</td>
</tr>
<tr>
<td>• Dependencies between modules</td>
<td>• For each participant</td>
</tr>
<tr>
<td>• Strengths of dependencies ($D_{ij}$)</td>
<td>○ Cost of participation ($C_k$)</td>
</tr>
<tr>
<td>• Growth rate for each module ($g_i$)</td>
<td>○ Value of participation ($V_k$)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The evolution of individual modules and the entire product over time</td>
</tr>
<tr>
<td>• Time of completion of the project</td>
</tr>
<tr>
<td>• Total effort required to complete the project</td>
</tr>
<tr>
<td>• Average effort invested per participant</td>
</tr>
<tr>
<td>• The distribution of efforts among the participants</td>
</tr>
</tbody>
</table>

To initialize the agent-based model, $N$ participants are created and their initial costs and values are assigned. The model is executed in cycles where in each cycle, all the agents make their individual decisions once about whether they want to contribute to the product development effort or not using the mixed equilibrium strategy. During each cycle, each agent decides in favor of contribution with a probability of $\alpha_0 \cdot N^*$. In other words, if the probability $\alpha_0 \cdot N^*$ is $1$, the agent decides in favor of contribution during each cycle. If the probability is less than one, the participant decides in favor of contribution during a fraction of the total number of cycles, depending on the value of $\alpha_0$. If, during a cycle, an agent decides to contribute, he/she works on the least developed module in the product and the module grows by a factor $g_i$, the module’s growth rate. It is assumed that as the product grows, the value to an individual in contributing to the product increases. The cost of participation to a participant is assumed to be constant throughout the execution of the model. The process is continued until all the modules reach a completion level of 100%.

Various outputs of the model are studied. First, the evolution of the modules is captured in terms of changes in their percentage completion over time. In this model, time is measured in terms of the number of cycles. The time of completion refers to the number of cycles required for the complete development of the product. The total effort is quantified as the sum of the number of cycles during which each participant contributes their effort (i.e., total person cycles). The average effort invested by the participants is measured as the average number of cycles during which each participant contributes to the project. The distribution of effort among the participants is important because it shows how the contribution rate varies among the participants.
2.2.4. Model Scope and Assumptions

The model presented in this paper represents an abstraction of the real MCPD scenarios. The model is based on the following assumptions:

- The percentage growth rate of all modules in a given cycle is assumed to be equal. This assumption is closer to reality when all modules are of almost similar complexity and require similar amounts of effort.
- The growth rate of each module is pre-determined and is expressed as a percentage of current progress.
- The dependency strengths are expressed as rework required on the modules. The rework is expressed as the percentage of the extent of development during a given cycle.
- The strategy used by individuals is to decide whether to work at a given interval or not. It corresponds to the mixed equilibrium strategy in the game of involuntary altruism.
- When a participant decides to contribute on the product, he/she selects the least developed module to work on.
- Only one participant can work on a module at a given time. In other words, a module gets locked when a participant starts working on it. This is similar to the check-out process often used during software development. This prevents multiple participants making conflicting changes to a particular module.
- All the participants have the same expertise. There is no difference between the modules in terms of the expertise required.

These assumptions define the scope of applicability of the model. The model is simple enough to aid in understanding the implications of various parameters in the model and extracting the dynamics of the MCPD processes. The future work will focus on developing a more comprehensive model by relaxing some of these assumptions.

In the following section, an example of a MCPD scenario is presented. The example is used to explore the effect of various parameters on the evolution of the product and the product development process.

2.3. Example problem

Generally, the products developed using mass collaborative approaches consist 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 [57]. Hence, the dependencies between the core modules are bi-directional whereas the dependencies between core modules and external modules are one way from the former to the latter. Such product
architecture [58] is chosen because it represents a wide range of products developed using mass collaborative processes. The examples of such product architecture are abundant in the open source software products, e.g., Linux [3] and Drupal [59]. In the case of Linux, the kernel represents the core set of modules such as memory manager, process scheduler, system call interface, virtual file system, and network interface, whereas examples of the external modules are applications, utilities, games, and security modules [60]. Another example of such a framework includes Drupal which is a web-based content management platform. The core consists of user management modules, general site configuration, etc. The external modules of Drupal such as e-mail, e-commerce, and file management are developed by user community and enhance the basic functionality of the product. The external modules also help in customization of the platform for user specific needs. These products differ in terms of the number of modules in the core and the external modules. In the domain of physical product development, an example is a computer with various core modules such as the motherboard and external modules such as video card, monitor, keyboard, etc. A cell phone with replaceable face plates can also be considered to have such an architecture where the basic cell-phone device is the core module and the face plates are the external modules. Such an architecture also supports mass customization because the external modules can be used to maximize external variety and the internal modules can be used to minimize internal variety.

The simple test example of a product used in this paper consists of nine modules and their interdependencies as shown in Figure 2. The core modules include Modules 0, 1, and 2. These external modules include modules 3, 4, 5, 6, 7, and 8. The strengths of dependencies between modules are also provided in the figure. For example, the dependency strength from Module 2 to Module 4 is 2.2, which implies that for a change in Module 2, the amount of rework required in Module 4 is 2.2%. The model is developed using NetLogo 4.0.2 [61].

![Figure 2 – Dependency Strengths in the Example Problem](image-url)
The specific values of the parameters used in the model are listed in Table 5. Unless otherwise mentioned, these values of the parameters are used in the results presented in the following sections. The default number of participants used in the model is 1000. The growth rate for each module is set at 1% of the current completion rate per cycle. The cost function is initialized as a random distribution in the interval \([1 \text{ to } 51]\) over all the participants. For the first cycle, the value function is randomly initialized within the interval \([1 \text{ to } 11]\). In order to simulate the effect of increasing value as product development proceeds, the value is represented as \(5 \times \text{(product completion)}\).

### Table 5 – Values of the parameters used in the model

<table>
<thead>
<tr>
<th>Parameters values used in the model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of participants</strong></td>
</tr>
<tr>
<td><strong>Growth rate</strong></td>
</tr>
<tr>
<td><strong>Cost</strong></td>
</tr>
<tr>
<td><strong>Value (benefit) to participants</strong></td>
</tr>
<tr>
<td><strong>Participants’ Strategy</strong></td>
</tr>
</tbody>
</table>

The results from the execution of the model for this example are discussed in the rest of the paper. The discussion is divided into two sections. In Section 3, a study of the product evolution along with the factors affecting the total effort and time required for the completion of the project is presented. In Section 4, aspects related to participants such as their incentives, strategies and evolution of participation structure are presented.

3. Understanding Product Evolution in Mass Collaborative Product Development

3.1. Rate of Product Evolution

One of the first aspects that the model allows us to study is the rate at which the individual modules as well as the entire product evolve. The evolution of the individual modules is shown in Figure 3. In the figure, four different types of modules are shown – a core module (Module 2), an external module that depends on the core module (Module 4), a coupled external module (Module 5), and a dependent external module (Module 7). Due to the symmetric nature of the product model, the evolution of Modules 0, 1, and 2 are similar.
The rate of evolution of the modules is dependent on a) the relationships with other modules, and b) the manner in which the growth function is defined. Note that despite the fact that during each cycle, the participants do not distinguish between the core and the external modules the time required for the evolution of different modules is different. As seen in the figure, the growth of the external modules does not start until the core module (Module 2) is completely developed. This is because of the fact that when the core module is under development, any change in the core modules results in rework in the external modules. After the core module is completely developed, the growth of the external modules can take place. The evolution of Modules 4 and 5 occurs almost at the same time (Module 5 slightly lagging Module 4). Module 7 evolves after all the other modules are completed due to the fact that it is influenced by the core modules and the other external modules. Module 7 does not influence any other module.

The evolution of the entire product is shown in Figure 4. Initially, the rate of growth of the product is very low. However, as soon as the core modules are fixed (~6000 cycles), the rate of evolution of the product increases significantly. Even though there were no clear project phases defined by the participants, the product evolution trend
clearly shows various phases of development – a) development of the core modules (0, 1, and 2), b) development of external modules that depend on the core modules (3, 4, 5, and 6), and c) the development of the second layer of external modules (7 and 8).

The rate of evolution of the product is inversely proportional to the strength of couplings. If the coupling strengths are significantly strong, it is even possible that the product evolution does not initiate. For example, in the specific problem presented in this paper, if the coupling strengths between the core modules are greater than or equal to 0.5, the modules do not grow at all because the rate of growth is the same as the amount of rework required in each module.

### 3.2. Patterns of Product Evolution and the Effect of Number of Participants

One of the key aspects to be studied is the effect of the number of participants (N) on the evolution of the individual modules and the overall product. Using this model, it is noticed that the pattern of evolution of the product changes significantly as the number of participants changes. The evolution patterns for N = 1000, 600, and 400 are shown in Figure 5, Figure 6 and Figure 7 respectively. Note that when the number of participants is high (N=1000), the development of the modules is almost **sequential**. The development of the core module (Module 2) is completed first. Then, the development of the external modules (Modules 4 and 5) begins, and the development of Module 7 begins after Modules 4 and 5 are completed. The pattern is sequential in spite of the fact that at each cycle, the participants work on the least developed module. At the other end of the spectrum where the number of participants is low (N = 400), a **concurrent** pattern of development is evident. In this case the evolution of both the core modules and the external modules happens at the same pace. In between the two ends of the spectrum (N=600), the evolution rate is a mix between entirely sequential and entirely concurrent.

The primary reason for these different evolution patterns is that in each cycle, if a participant decides to contribute to the product (based on the strategy), he/she selects the module that is least developed. If the number of active participants during a cycle is greater than the number of modules, then all the modules are developed in each cycle according to the growth rate. This causes the activation of all the dependencies, necessitating rework governed by the dependency strengths. In such a scenario, the effective growth rate of a module is equal to the difference between the growth rate and rework. In the example problem under consideration, the effective growth rate of
Modules 0, 1, and 2 is thus 0.2 (= 1 – 2*0.4). If the core modules are not developed completely, the growth rate of module 4 is less than zero (= 1 – 2.2). Hence the growth of Module 4 cannot start until the core modules are completely developed. This results in a sequential evolution pattern.

If the number of active participants in a cycle is less than the number of modules then during each cycle, only the modules that have lower percentage completion are worked on. During each cycle, the modules that are lagging on development are picked up by the participants and developed until the percentage completion exceeds the other modules at which time those other modules are picked up for development. This provides a balance to the development rate across all modules. In other words if the number of participants is low, then all the modules are developed at almost the same time resulting in a concurrent evolution pattern. Further, since some of the modules may not change, all the dependencies requiring rework are not active during each cycle. Note that the evolution pattern in this scenario also resembles a team-based top-down product development scenario.
In addition to the evolution of modules, the time taken for evolution of the solution is also different for different values of N. As can be noticed in Figure 5 through Figure 7, the time required for completion of the project increases as the number of participants reduces. To study the effect of number of participants on the total effort, a parametric study is performed. For this parametric study, it is assumed that all the participants have a fixed Value of 100 and a fixed Cost of 50. The results of the parametric study are plotted in Figure 8. There are two stable product evolution patterns and one transition pattern. As the N is increased from a lower value to a higher value, there are three distinct patterns of evolution as discussed above – concurrent pattern for low values of N, transition pattern for intermediate values of N, and sequential pattern for higher values of N. The corresponding evolution patterns of the modules are also shown in the figure.

The total effort required for product evolution in the concurrent pattern is constant with varying N. Similarly, the total effort in the sequential pattern is also constant with varying N, although the amount of the total effort is lower than that in the concurrent pattern. However in the transition pattern, the total effort reduces monotonically as the number of participants increase. There are two critical values of N that represent the transition from one pattern to another. Below the lower critical value, any reduction in the number of participants does not result in increase in the overall effort. Above the higher critical value, any increase in the number of participants does not result in any reduction in the overall effort.
From the standpoint of MCPD, it is interesting that a) there are these patterns of evolution of the product, b) the number of participants affects self-organization into different patterns, and c) the overall time and effort are dependent on the pattern of evolution. So, by directing the product development in the direction of appropriate pattern, it is possible to achieve the desired goals. The number of participants is only one of the factors that drives the process towards a particular pattern. Other factors include the product architecture, strength of couplings, etc. A catalyst of a MCPD process can use these insights to determine how to orchestrate the project so that the desired conditions are achieved.

3.3. Effect of Prior Work on Product Evolution

One of the key aspects to consider in MCPD projects is to decide when to open up the development process for mass collaboration. The proposed model allows us to study the effect of the initial percentage development of the modules. Consider a scenario where each of the modules is developed to 80% by a dedicated team before it is opened up for community based development. The evolution of the modules and the overall product development in that scenario is shown in Figure 9.

Figure 8 - Effect of Number of People on the Total Effort and Module Evolution Patterns
As can be noted in the figure, only the core modules (Module 2) have a positive growth rate. The evolution of external modules is negative, i.e., the percentage completion of modules 4, 5, and 7 reduces with the increase in the number of cycles. This results in a negative growth rate of the overall project until around 100 cycles. In fact, the percentage completion of Module 7 goes almost down to zero before it starts increasing again. This is due to the rework caused by changes in the other strongly dependent modules that affect Module 7. Due to the strong interdependencies, most of the external modules are re-developed due to the strong coupling between the core and the modules.

As discussed before, the rework mainly depends on the strength of dependencies. In some scenarios where the coupling strength is low, the growth rate of all modules may be positive. For example, assume a scenario where the strength of dependency between module 2 and module 4 is 0.4 (instead of the default value of 2.2). The evolution of modules in this scenario is shown in Figure 10. In this scenario, the growth of all modules is positive. Further, the sequence of evolution of the modules is different from the one in Figure 9. Module 4 is the first one to be completed,
whereas the development of Module 2 is completed at the end. This is due to the fact that with the change in coupling strength, the core modules do not have a strong influence on the development of the other modules.

Figure 10 - Evolution of modules for an initial percentage completion of 80% and core module → external module coupling changed from 2.2 to 0.4

Figure 11 - Changes in the total effort as the initial percentage completion changes

The changes in the total effort with the increase in initial percentage completion are shown in Figure 11. When the initial percentage completion is low (0 – 10%), the curve is significantly steeper than when the percentage completion is high (10 - 100%). The implication of this trend on the product development process is that with some initial work within a closed team before using mass collaborative process, significant amounts of time can be saved. This insight can be used to determine a) how much initial work is required before a project should be opened up for mass collaboration, and b) the effect of dependencies between modules on this decision.

3.4. Effect of Dependency Strengths on Product Evolution

The effect of dependency strength on the total effort required for product evolution is shown in Figure 12. The impact of dependencies between the core modules is shown on the top and the impact of dependencies from core modules to external modules are shown at the bottom. As expected, as the dependency strength increases, the
amount of effort required to develop the complete product increases. It is observed from the figure that the relationship between effort and dependency is almost linear in the second case, whereas is strongly non-linear for the first one. The effort required increases exponentially with the increase in dependency strength. These figures provide insight into developing ways to expedite the MCPD process. In order to reduce the amount of effort required to develop the product, it is important to reduce the dependencies between the core modules than between the core modules and external modules.

Figure 12 – Effect of dependencies between core modules and from core modules to external modules on total effort

4. Aspects Related to Participants in the Mass Collaborative Product Development Processes

4.1. Evolution and Distribution of Participants’ Contributions
In this section, we study how the participation in the product development changes along the process. In Figure 13(Top), a histogram showing the number of participants as a function of the amount of effort invested throughout the project is presented. From the shape of the histogram it is observed that there are two types of contributors –
primary contributors and secondary contributors. The primary contributors are the ones who participate throughout the project and the amount of effort they invest in the project is significantly higher than most of the contributors. The secondary contributors are the ones who join in the later phases of the project and mainly contribute to the external modules. The number of primary contributors is generally significantly smaller than the secondary contributors. For example, in the figure shown, nearly 20% of the contributors are primary contributors and the remaining 80% are secondary contributors. This trend is analogous to the contributions in open-source software development where the primary contributors are referred to as committers [53].

![Effort Histogram](image1)

![Number of Active Participants](image2)

*Figure 13 – (Top) Histogram of participants with different amounts of effort invested in the project, and (Bottom) the growth of active participants (both parts of this figure are for value function coefficient = 5.0)*

In Figure 13 (Bottom), we study the growth of the number of new participations in the product development process. A plot of the number of active participants is shown as a function of the number of cycles, where a participant is active if he/she has contributed at least something to the project so far. The plot consists of four phases of participation:
• Phase A (Initial participation): In this phase, a set of participants who have personal benefits from contributing to the project start developing the product. This is one of the main motivations of people participating in open source software development projects [20, 52, 53].

• Phase B (Slow growth): The same initial group of participants (committers) continues working on the project until significant progress is made and other participants see the value in it.

• Phase C (Accelerated growth): After significant progress has been made to the project and other participants see the value in the project, they join the effort in large numbers.

• Phase D (Complete participation): In this phase, most of the people are involved in the project and have contributed at-least something to the project.

Note that the exact shape of the curve depends on various factors such as the initial distribution of cost and value functions. However, the overall trend in the increase in active participants remains similar.

4.2. Effect of Incentives

Various mechanisms can be used to motivate the participants to contribute to the project. In order to provide the right incentives to the participants, it is important to understand the impact of the participants’ value function on the total effort required to complete the project. As discussed earlier, the benefit from participating in the product development process is simply modeled as an increasing function of the percentage completion of the entire project: 
\[ V = k \times (\% \text{ completion}) \], where \( k \) is the value function coefficient. As \( k \) increases, the participants receive greater value for a given \% completion of the project and hence are more willing to participate in the project.

The effect of modifying the incentives for participation has been modeled by changing the numerical value of the value function coefficient \( (k) \). The total effort required for the project is plotted for different values of \( k \) in Figure 14. It is observed from the figure that as \( k \) increases, the total effort required for completion of the project reduces. The amount of time required for completion of the project also goes down. After certain value of the coefficient (in this case, \( k \sim 30 \)), any further increase does not correspond to the reduction in total effort. This trend is mainly related to the different product evolution patterns as discussed earlier in the paper. The evolution patterns of the modules and the product are also dependent on \( k \).
It is noticed that as the value function coefficient increases the distribution of efforts also changes. This is shown in the figure by two histograms corresponding to the points with \( k=2 \) and \( k=50 \). When \( k \) is low, the effort histogram is heavily concentrated on the low effort (left) side i.e., most of the participants are secondary contributors. However, when \( k \) is high, the histogram is more balanced throughout the spectrum implying that there is almost equal number of primary and secondary contributors. Although the specific shapes of the curves depend on various factors such as the product architecture, strength of couplings, number of participants, etc., the general trend remains the same.

Based on the results shown in this section, it is clear that the manner in which people get involved is closely tied to

\begin{enumerate}
  \item their incentives,
  \item the product structure and couplings.
\end{enumerate}

Furthermore, the proposed model helps in identifying these relationships so that appropriate conditions for self-organization can be determined.

5. Closing Thoughts
The field of MCPD is in its early stages. The mechanisms underlying the mass collaborative processes are different from traditional processes in that they are driven by bottom-up evolution as opposed to top-down decomposition. The successful implementation of mass collaborative processes requires a deeper understanding of the underlying evolutionary dynamics of these processes. The agent-based computational approach to modeling MCPD processes presented in this paper is a step towards gaining the required understanding. The approach is based on modeling the products and the participants. The products are modeled as modules and their interdependencies, and the participants
are modeled using their strategy defined in terms of the participation costs and values. As mentioned earlier, the objective is to gain a qualitative insight into mass collaborative processes and the factors that affect product evolution. The objective is not to exactly replicate all the details of a specific mass collaborative project.

Despite the simplicity of the model, various insights into the key aspects of mass collaboration (see Table 6). The model allows us to study the relationship of the rate of product evolution with the modules and their dependencies. Through the model, it is shown that in mass collaborative processes, different kinds of patterns (such as sequential and concurrent) can emerge even without explicitly planning for them. Further, depending on the kind of product evolution pattern, the overall time for product development and the effort may be different. The development efforts carried out prior to opening up the product for mass collaboration have a significant effect on the product evolution. Depending on the percentage completion, some of the modules may need to be significantly reworked. The model also provides insights into the specific modules that can be developed before opening up the process. The model also helps in identifying the key dependencies that can be modified (reduced) to expedite the evolution in the mass collaborative process. The model can be used to understand how the community grows and helps in identifying ways to accelerate community growth. Finally, the model helps in understanding the effect of incentives to the participants on the overall time required for product development, the effort involved, and the distribution of effort among participants.

Table 6 - Insights gained from the agent-based model presented in this paper

<table>
<thead>
<tr>
<th>Aspects of Mass Collaboration</th>
<th>Insights from the model presented in the paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Rate of product evolution</td>
<td>Relationship to modules and dependencies</td>
</tr>
<tr>
<td>2 Patterns of product evolution</td>
<td>Emergence of patterns such as sequential and concurrent</td>
</tr>
<tr>
<td>3 Effect of prior work</td>
<td>Determining how the prior effort should be invested</td>
</tr>
<tr>
<td>4 Effect of strengths of dependencies</td>
<td>Identifying key dependencies that affect the evolution rate</td>
</tr>
<tr>
<td>5 Evolution of participation</td>
<td>Understanding how the community grows</td>
</tr>
<tr>
<td>6 Effect of incentives</td>
<td>Understanding how incentives affect the overall time, and effort</td>
</tr>
</tbody>
</table>

Although a comparison of different product architectures is not performed in this paper, the model can be used to study the effect of product architecture on product evolution and to identify how to modify the architecture to increase the rate of product evolution. Each of the aspects of mass collaboration shown in Table 6 can be individually analyzed in greater details in the future for different kinds of scenarios. North and Macal [43] pointed out that agent-based models should be developed in an incremental manner so that the basic emergent phenomena can be first identified and understood before adding more complex aspects of specific scenarios. The future research
objectives are to extend this model by a) applying the model to a real case study, b) modeling different types of tasks undertaken by participants in a mass collaborative scenario, c) considering collaborations between the participants and dynamic formation of teams, d) modeling the learning behavior of participants as the product evolves, e) considering different kinds of participant strategies, f) modeling the variations in participants’ competencies and behavior, g) modeling the details of evolution of modules, and h) modeling organizations that utilize mass collaborative processes, where some aspects may not be open for mass collaboration. There are also further opportunities to perform experimental and analytical studies of mass collaborative processes. The author acknowledges that many practical processes include a combination of both top-down and bottom-up aspects. The model will be extended in the future to account for this.

The primary research contribution in this paper is a systematic computational study of mass collaborative design processes and the determination of various factors and their effect on the product evolution. Through the model, the highly coupled nature of product evolution, the participants’ strategies, their values and costs is demonstrated. Further, in this paper, MCPD processes are studied from an evolution perspective as opposed to the traditional top-down decomposition perspective. The approach presented in this paper represents a step towards understanding how managers and business executives can make best use of mass collaboration in gaining value for their organizations. It has applications not only in the open source software development arena but also in exploring the application of mass collaborative development of physical products. The author believes that the approach has a potential to provide a foundation for the simulation of open innovation processes that are increasingly being adopted by various organizations.

6. References


List of Figures

Figure 1 – A simple illustrative example of product model
Figure 2 – Dependency strengths in the example problem
Figure 3 – Evolution of individual modules
Figure 4 – Evolution of the entire product
Figure 5 – Evolution of individual modules for N = 1000
Figure 6 – Evolution of individual modules for N = 600
Figure 7 – Evolution of individual modules for N = 400
Figure 8 – Effect of number of people on the Total Effort and Module Evolution Patterns
Figure 9 – Evolution of modules and overall product for an initial percentage completion of 80%
Figure 10 – Evolution of modules for an initial percentage completion of 80% and core module external module coupling changed from 2.2 to 0.4
Figure 11 – Changes in the total effort as the initial percentage completion changes
Figure 12 – Effect of dependencies between core modules and from core modules to external modules on total effort
Figure 13 – (Top) Histogram of participants with different amounts of effort invested in the project, and (Bottom) the growth of active participants (both parts of this figure are for value function coefficient = 5.0)
Figure 14 – Effect of value function coefficient on the total effort and effort distribution

List of Tables

Table 1 – Differences between traditional and MCPD scenarios
Table 2 – Questions addressed in this paper using the agent-based model
Table 3 – Normal form of the game of Involuntary Altruism [54]
Table 4 – Parameters in the agent-based model for MCPD
Table 5 – Values of the parameters used in the model
Table 6 – Insights gained from the agent-based model presented in this paper