## Contents

1 CAPPD: Product Development by Self-Organized Virtual Communities  
   *Jitesh H Panchal and Qize Le*  
   1.1 Introduction ...................................................... 1  
      1.1.1 Community-based Product Development - An Overview ........ 1  
      1.1.2 Overview of the Chapter .................................. 3  
   1.2 Network-based Analysis of Products and Communities ............... 4  
      1.2.1 Structures and Evolution of Communities .................. 5  
      1.2.2 Generative Network Models of Product Structure ........... 8  
      1.2.3 Analysis of the Interdependent Co-evolution of Products and Communities 10  
   1.3 Computational Modeling of the Effects of Individual Decisions .... 14  
      1.3.1 Agent-based Model of Community-based Product Development .... 15  
      1.3.2 Assumptions and Model Parameters ....................... 16  
      1.3.3 Outputs of the Model .................................... 17  
   1.4 Research Outlook ............................................. 20  
      1.4.1 Theoretical and Empirical Research ....................... 20  
      1.4.2 Platforms for Community-based Physical Product Development .... 21  
      1.4.3 Community-based Product Development in Education .......... 22  
   1.5 Acknowledgments ............................................. 22
1

CAPPD: Product Development by Self-Organized Virtual Communities

Jitesh H Panchal
Purdue University, West Lafayette, IN, USA, E-mail: panchal@purdue.edu

Qize Le
Washington State University, Pullman, WA, USA

1.1 Introduction

1.1.1 Community-based Product Development - An Overview

The paradigm of community-based innovation is being instantiated in a variety of forms such as open-source product development, innovation by communities within large organizations (aka Enterprise 2.0), crowdsourcing, and the emerging maker communities (see Figure 1.1). These concepts have not only been utilized in software development, but also in physical products such as open-source cars [1, 2] and 3D printers [3]. Recently, DARPA invested in exploring community-based development of systems through the Adaptive Vehicle Make (AVM) program [4], which has three
components: META, instant foundry adaptive through bits (iFAB), and fast adaptable next generation ground vehicle (FANG). The DARPA FANG program is focused on designing and developing a new heavy, amphibious infantry fighting vehicle (IFV) through prize-based design competitions including mobility/drivetrain challenge, chassis/structural challenge, and total platform challenge. To support collaboration, DARPA funded the development of VehicleFORGE [5], which is an open source development and collaboration environment for the creation of large, complex systems by numerous unaffiliated designers. Organizations such as Procter and Gamble [6] and Kraft [7] are also exploring ways to utilize communities to generate new ideas for product development. Ford, in collaboration with Bug Labs, is developing an open-source hardware and software platform called OpenXC to take advantage of social innovation for in-car connectivity [8]. Considering the potential applications, understanding and supporting community-based product development is an important emerging research area for the engineering design and systems engineering research communities.

The key characteristics that distinguish the emerging community-based product development paradigm from traditional product development are highlighted in Table 1.1. 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. The top-down decomposition-based view of the process is accurately captured by the systems engineering Vee model [9]. In an ideal systems development process, it is assumed that a) the processes are well defined in terms of activities and dependencies, b) the individuals and their competencies are aligned with the activities to be performed, c) the organizational goals are aligned with individual goals, d) the organizational structures are aligned with the processes, and e) the processes are aligned with the product structures. Over the years, these assumptions have resulted in various hierarchically structured approaches based on top-down problem decomposition, where the emphasis is on questions such as i) how should product development tasks be decomposed? ii) how can concurrency be achieved? iii) how should tasks be carried out to minimize the overall product development time? and iv) how should product development tasks be sequenced?

In contrast, the alignment between the system, the organizational structure, engineering processes, decisions, and activities cannot be assumed in community-based product development because they are driven by self-organization of individuals based on their personal interests. Hence, 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 activities. The participants’ contributions are not based on the pre-assigned tasks, and the product evolves over time based on the contributions of the participants. If the right initiatives are available, the participants contribute to the projects on their own. Such self-organized activities can be effective for simple
TABLE 1.1: Distinction between traditional firm-based and community-based production and innovation

<table>
<thead>
<tr>
<th>Category</th>
<th>Firm-based</th>
<th>Community-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Hierarchical</td>
<td>Loose network of peers</td>
</tr>
<tr>
<td>Participants</td>
<td>Guided by organizational goals</td>
<td>Self-directed</td>
</tr>
<tr>
<td>Proximity</td>
<td>Primarily local ties</td>
<td>Local and non-local ties</td>
</tr>
<tr>
<td>Systems</td>
<td>Systematically designed to targets</td>
<td>Evolve over time</td>
</tr>
<tr>
<td>Task assignment</td>
<td>Tasks assigned to participants</td>
<td>Participants self-select tasks</td>
</tr>
<tr>
<td>Project initiation</td>
<td>Initiated after market analysis</td>
<td>Project initiated by participants</td>
</tr>
<tr>
<td>Decision-making</td>
<td>Top-down hierarchical</td>
<td>Decentralized decision making</td>
</tr>
<tr>
<td>User involvement</td>
<td>Low during the development</td>
<td>Users are the developers</td>
</tr>
</tbody>
</table>

products such as individual 3D printed artifacts, or highly modular products such as Wikipedia. But for complex systems, such as automobiles and aircraft, their effectiveness is yet to be established. This open research issue is the driver for the motivational question in this chapter:

How can complex systems be effectively engineered by self-organized communities of self-directed individuals?

Due to the self-directed nature of communities, there are unique technical, organizational, social, and economic challenges associated with community-based development of complex systems. The key questions in such bottom-up development processes are: a) how does a product evolve? b) how are individual contributions distributed, recognized, and rewarded? c) how does participation change over time? d) what is the effect of interdependencies between product modules? e) how does the self-organization of communities affect product evolution? and f) what kinds of incentives are necessary to encourage participation. Due to the tight coupling between the product evolution, individual decisions, and community evolution, community-based product development represents a socio-technical system where both social and technical aspects must be analyzed concurrently. In this chapter, we present approaches to address some of the socio-technical issues pertaining to community-based product development. An overview of the chapter is presented next.

1.1.2 Overview of the Chapter

In this chapter, our goals are to highlight the issues within self-organized community-based product development, to discuss some of the results from the authors’ work, and to highlight the open research issues pertaining to the use of these concepts for physical product development. The research efforts discussed in this chapter utilize theories and computational techniques from the fields of engineering design, complex networks, sociology, and organization science. The research approach presented in this chapter is a combination of inductive and deductive techniques (see Figure 1.2). In the inductive techniques, existing products and communities are observed. Based on the observations, hypotheses about the behaviors of the individuals are generated. These hypotheses are used to establish new theories about the behaviors of individuals and product evolution. On the other hand, deductive techniques utilize theories to generate hypothesis, which are validated by observing real projects.

The body of the chapter is focused on two aspects:

- network-based analysis of the structures and evolution of the products and communities,
- computational modeling of product evolution through decisions made by individuals,
The first aspect, discussed in Section 1.2, is focused on the analysis of structures and evolution of products and communities. While a product's structure has significant implications on the product's quality and the complexity of product development process, recent studies within organizational science suggest that an organization's structure also has a direct impact on the product structures [10]. This interdependence is particularly important in community-based product development because there is no community structure imposed at the beginning of the process. Both the product structure and the community structure undergo dynamic co-evolution. Adopting a dynamic network-based approach where products are modeled as networks of interfacing modules, and communities are modeled as networks of collaborating participants, we present an inductive study in Section 1.2.

The second aspect, discussed in Section 1.3, involves computational modeling of product evolution as individuals make decisions about participation in these communities. Here, we present a deductive study, starting from a game-theoretic model of the participants' behaviors and deduce the evolutionary characteristics of the products. The game theoretic model of participants is instantiated in an agent-based model. The model captures information about products as networks, and participants as: decision makers who make decisions such as whether to participate or not, and which module to work on. The model is used to study a variety of aspects of community-based product development including the rate of evolution of the individual modules and the entire product, product evolution patterns and the effect of the number of participants, the effect of prior work on product evolution, the evolution and distribution of participants, and the effect of incentives to participants.

Finally, in Section 1.4, we discuss open research issues in the field of community-based product development. The discussion is primarily focused on physical products. Issues related to open-source hardware development, computational platforms for community-based physical product development, and engineering education are discussed.

### 1.2 Network-based Analysis of Products and Communities

In this section, we present network-based analysis of products and communities, and their interdependence. The structure and evolution of the communities is analyzed in Section 1.2.1. The evolution of products is presented in Section 1.2.2, and the analysis of the interdependence is presented in Section 1.2.3. We use a software product, Drupal, as an illustrative example throughout this section. Drupal [11] is an open-source content-management system which is used for the creation of community-based websites. A software product is chosen, in this chapter, instead of a physical product because of the availability of data regarding product structures and community structures. The concept of open hardware is relatively new and there is a lack of information necessary for analyzing the interdependence between the products and communities (discussed further in Section 1.4). Drupal has been under development since 2001. The analysis in this section is based on five major versions of Drupal core (2.0 through 5.0). Drupal is well developed with over 7000 community-contributed add-ons, known as contrib modules. Besides, the project also attracts more than 1000 developers. Drupal is selected because of its maturity and the availability of code for different versions.
1.2.1 Structures and Evolution of Communities

Communication between individuals is a way of achieving coordination between activities carried out by different individuals. Hence, the analysis of community structures that embody communication patterns between individuals is important for understanding community-based product development. Community structures have been analyzed in a variety of contexts such as in scientific collaborations and acquaintance networks [12, 13]. Within the domain of virtual product development, existing research is primarily focused on open-source software communities. Weber [14] discusses different types of community structures in various OSS projects. For example, the community structure of the Linux project reflects a pyramid structure whereas the community of the BSD project is represented as concentric circles. Crowston and Howison [15] analyze hierarchy and centralization in OSS communities of Apache, Savannah and SourceForge by employing social network analysis (SNA) metrics. Xu and co-authors [16], and Gao and Madey [17] study topological properties of open-source communities, including degree distribution, diameter, clustering coefficient, centrality and component distribution by modeling OSS communities as complex social networks. Some efforts have also been carried out on the evolution of the communities (e.g., Nakakoji et al. [18] and Weiss et al. [19]). Howison et al. [20] investigate the structure of OSS development communities over time using snapshots of data to understand the dynamics of social structures in OSS development communities. They examine three properties of the social structures, namely, centralization, network center, and stability of participation.

In this section, we present a network-based representation of the communities where nodes are individuals and the links are relationships (e.g., communication) between individuals. We start with a simple model of community structure by assuming that two individuals working on the same module during a specific version of a product are connected to each other. Based on this assumption, data from existing projects are used to answer the following questions: a) What are the structural and evolutionary characteristics of the community network? and b) How can we infer the behavior of the participants from the community structure? The inductive approach adopted in this section is to utilize degree distribution and SNA measures to quantify the structural characteristics of the networks. Based on the observed structural characteristics, a hypothesis about the participants’ linking behavior is generated. The hypothesis is validated by comparing real network with the networks generated by the hypothesized linking behavior.

Generation of networks

In the first step, raw data about the participants and the product modules that they contribute to are extracted. In order to study the evolution of the community network, the following information is collected: a) the joining dates of individuals, b) the dates of individuals’ first contribution, and c) their contributions to different modules. For the Drupal project, the information includes the users’ ID, the first and last time each user made a revision of this module (i.e., first and last commits), and the number of times each user revised the module. The data gathered are two-mode in nature, i.e., there are two types of entities: people and modules. The two-mode data are first converted into a bipartite network, \( G = \{S_1 \times S_2, E\} \), which consists of two disjoint sets of nodes \( S_1 \) and \( S_2 \) and a set of edges \( E \) such that each edge in \( E \) connects a node in \( S_1 \) to a node in \( S_2 \). Bipartite networks are also referred to as affiliation networks in the social network analysis literature [13]. An example of a bipartite graph is shown in Figure 1.3, where \( S_1 = \{a, b, c, d, e, f, g\} \) and \( S_2 = \{1, 2, 3, 4\} \).

The links in the network connect a person with a module, representing a person working on a module. The bipartite network can be weighted or binary. If a binary matrix is used, then the links only represent the presence of relationships between people and projects. However, if a weighted network is used, the weights on the links can be used to represent the amount of effort invested by the participants on corresponding modules. An indicator of the amount of effort is the number of commits by a person to a module.
The bipartite graph $G$ can be transformed into two weighted undirected graphs $G_1 = \{S_1, E_1\}$ and $G_2 = \{S_2, E_2\}$ consisting of people ($S_1$) and modules ($S_2$) respectively. Figure 1.3 provides an illustration of graphs $G_1$ and $G_2$ derived from the bipartite graph. Two people in $G_1$ are connected by an edge if both of them jointly work on at least one module. Similarly, two modules in $G_2$ are linked if they have at least one common participant. The weights associated with edges $E_1$ represent the number of modules shared by a set of people. Similarly, the weights in graph $G_2$ represent the number of common participants shared by modules. These networks are also referred to as 1-mode $S_1$ and $S_2$ projections of the bipartite network. The projected graphs can also be represented in a matrix form as shown in the figure. An adjacency matrix of a network with $n$ nodes is an $n \times n$ matrix, where an element $a_{ij}$ denotes the weight on the edge from node $i$ to node $j$, and 0 denotes no connection between nodes $i$ and $j$. The diagonal elements are conventionally set to 1.

Network analysis of communities

After creating the adjacency matrix, the network properties are explored using different techniques. In order to characterize the key features of the community network, we use the network’s degree distribution. Degree is the number of nearest neighbors of a node. In an undirected graph, the degree of a node $s$ is the number of edges incident with $s$ and is denoted by $k_s$ [21]. The degree distribution, $P(k)$, of a network is defined as the fraction of nodes in the network with degree $k$. The degree is a node-level property whereas the degree-distribution determines the global characteristics of a network. In a bipartite network, two degree distributions corresponding to both types of nodes are important. The joint degree distribution of a network, $P(k_1, k_2)$, represents the probability that a randomly selected edge is connected to nodes with degrees $k_1$ and $k_2$. In the community network, the degree describes the number of relationships that a participant has. The degree distribution represents the fraction of participants in the community with the same number of relationships. The degree distribution is arguably the most important characteristic of a complex network. Degree distribution provides a key indication about the topology of a network. For example, Bernoulli random networks have a Poisson degree distribution. Exponential random graphs have an exponential degree distribution. Scale-free networks have a power-law degree distribution [22]. Recent research suggests that degree distributions of many real-world networks satisfy a power law, $y = bx^a$ where $b$ and $a$ are constants and $y$ denotes the number of nodes with degree $x$. Scale-free networks have a property that only a few nodes (called “hubs”) have a high degree, while most other nodes are only connected to a few nodes. Scale-free networks have different characteristics as compared to Bernoulli random networks. The degree distribution can provide insights not only about the structure of networks, but also about their evolution. Clustering coefficient is another measure for the topology of a network. It is the probability that two nearest neighbors of a vertex are also the nearest neighbors of one another.

The degree of a person in the bipartite network suggests how active a participant is. People with higher degree participate in a greater number of projects. Similarly, higher degree of a project in the bipartite network indicates that a greater number of people work on a project. Similarly, the degree of people in the 1-mode network indicates the extent of collaboration of an individual with
other participants. As opposed to degree, which is a node-level attribute, the clustering coefficient is a neighborhood-level measure. The clustering coefficient reflects the “cliquishness” of the mean closest neighborhood of a vertex. Higher clustering coefficients indicate greater number of communication pathways between members of a subgroup.

The degree distributions for three different versions of Drupal are shown in Figure 1.4. It is observed that during each snapshot of the network, degree distribution has a similar form - power law. The only difference is in the parameters of the power law. As shown in the figure, the coefficient increases while the exponent decreases with time. In a growing network, such a degree distribution has been shown to be a result of preferential attachment [22]. Preferential attachment means that the probability of attachment of a new node to an existing node is proportional to the degree of the existing node. Based on this observation, our hypothesis is that preferential attachment mechanism underlies the growth of the Drupal network also.

**Simulation model of community network evolution**

In order to validate the hypothesis, we present a computational model whose input is the behavior of the nodes (people) and the output is the bipartite network at various time-steps. The behaviors of the nodes in the network include activities such as new individuals joining the network, creation of new modules, and participation of existing individuals on existing modules. The model is based on the assumptions that only one new participant joins the network in each time-step, the probability of an existing participant creating a new module is constant, and the number of links added between existing participants and existing modules is constant in each time-step. During each time-step, the network grows in three ways. First, a new individual is added to the model during each time-step. The new individual either initiates a new module with probability $p_a$, or contributes to an existing module with probability $(1 - p_a)$. If the new participant creates a new module, a link between the new participant and the new module is created. Otherwise, the new participant preferentially links to one of the existing projects based on their degree. Second, existing participants can also create new modules. The probability that a new module is created by existing participants is determined by the ratio of total number of people and modules. Finally, existing participants can work on existing modules. The links are added between projects and participants using the preferential attachment mechanism. The degree distributions of the Drupal bipartite graph and the bipartite graph generated using this model are shown in Figure 1.5.

Further analysis of the network using Social Network Analysis (SNA) measures is provided.
CAPPD: Product Development by Self-Organized Virtual Communities

FIGURE 1.5: Comparison of degree distribution between Drupal product and the network generated using the model

in [23]. SNA is a theoretical and methodological paradigm for examining complex social structures [21]. The following measures are used to compare the network generated by the model and the real network: clustering coefficient, diameter, density, connectedness, and degree centrality. UCInet [24], which is one of the SNA tools, is used for the analysis.

1.2.2 Generative Network Models of Product Structure

In addition to modeling the community structure, modeling the structures of products is important to determine the effect of complexity on community-based product development. The role of product structure has been well-documented in product development [25]. It has been shown that different kinds of product architectures have different effects on the product development. For example, modular product architectures significantly improve the efficiency of product development processes because of clear partitioning of sub-functions resulting in fewer dependencies between corresponding tasks. Yassine and Wissmann [26] point out that appropriately designing the product architectures could even lead to effective strategies for increasing service levels. While various studies related to product architectures have been conducted for hierarchical product development (see Table 1.1), there are relatively fewer studies analyzing the product structures in community-based virtual product development. The analysis is important because in contrast to the hierarchical product development, complete product architecture is not predefined, but generally evolves as a result of the (self-directed) contributions of the participants. Hence, in addition to the initial product structure, the evolution of the product structure also influences the development of the product with impacts on productivity, complexity, rate of product development, and product quality. An understanding of the effect of the product structure and its evolution facilitates the identification of the types of products that are suitable for community-based production, and the design of product structures for existing systems to support the emerging product development paradigm.

Since community-based product development is bottom-up in nature, the evolution of the product structure should also be modeled in a bottom-up manner. In this section, we present such a bottom-up model of the generation of the product structure. The generative product model is based on the addition and deletion of nodes (i.e., product modules) and links (i.e., the dependencies between modules). The model is statistical in nature, and can be estimated from the product structure information available at different points in time. The hypothesis in the proposed model is that six network evolution mechanisms at the node level determine the evolution of product structures. The evolution mechanisms can depend on a number of structural parameters or product-related aspects. In this section the process is idealized by assuming that the mechanisms are only dependent on the degrees of nodes.
The six node-level mechanisms for product evolution are as follows:

1. **Addition of new nodes.** The primary growth mechanism of a network is the addition of new nodes. This mechanism corresponds to the addition of new modules, functions, classes, etc. to address new requirements, specifications and features. Similar nodes can be defined for physical products also. The trends in the number of additional nodes can be determined by comparing consecutive versions of the product structure. The level of abstraction considered by Le and Panchal [27] is the function-call level. The network is also referred to as a function-call network.

2. **Removal of existing nodes.** Existing nodes may be removed from a product network because existing features may no longer be needed or are replaced by new features. The number of existing nodes removed from a product can also be determined by comparing consecutive versions of the product.

3. **Linking of new nodes with existing nodes.** After new nodes are added, these nodes can be linked to existing nodes by new interfaces. The probability of a new node linking to existing node is assumed to be a function of the degree of the existing node. To determine the probability density function, two consecutive versions of the product network are compared.

4. **Linking of new nodes with each other.** Since new nodes do not have any initial links, it is assumed that the new nodes first link with existing nodes and then link with new nodes. After the new nodes link with the existing ones, the degree of a new node is referred to as the “initial degree”. This initial degree is used to determine the probability of creation of links between two new nodes.

5. **Linking of existing nodes with each other.** New links can also be added between existing nodes. This corresponds to the addition of new function calls between two existing functions. The probability that an existing node is attached to other existing nodes is assumed to be a function of the target node’s degree.

6. **Removal of existing links.** Existing links can be removed if the existing nodes are removed or if the links between two existing nodes are no longer used. The probability of removal of existing links between existing nodes is calculated similarly by comparing consecutive versions of the product network.

The primary difference between this model and the existing degree-based models such as the Barabasi-Albert model [22] and its variants is that existing models assign a pre-specified functional form to the probability functions for each mechanism. However, the model discussed above does not pre-assign any probability function. These functions are determined and parameters are estimated based on the observed data.

Le and Panchal [28] utilize the evolutionary mechanisms to model the evolution of the product structure for Drupal. To initialize the network generation process, the authors use three types of initial networks with similar size: an Erdos Renyi random network [29], a scale-free network, and Drupal version 2 network. The structures of the networks generated using three types of initial networks are compared with the product structure networks from version 2 to version 5. A comparison of degree distribution among models with three types of initial networks and Drupal is shown in Figure 1.6. It is observed that although the degree distributions of the initial networks are substantially different, by generating the product structure using the mechanisms discussed above, the degree distributions of the three models converge to the degree distribution of Drupal project. This indicates the robustness of the node-level mechanisms in modeling the product evolution. The authors also compare other network characteristics including average degree, average density, diameter, clustering coefficient, average shortest path, propagation cost and clustered cost. The results indicate
that the mechanisms can potentially be used to model the evolution of product structures in Drupal. Further details of the results for Drupal and two other software products are presented in [28].

**FIGURE 1.6:** Comparison of degree distribution between Drupal product and the network generated using the model

### 1.2.3 Analysis of the Interdependent Co-evolution of Products and Communities

In the engineering design literature, social aspects related to the communities and the technical aspects related to the product architectures have mostly been studied independently. However, in community-based product development, these two socio-technical aspects are interdependent. The dependence of products on communities is clearly highlighted by Conway’s law [10,30] - 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. At the same time, the communication between different participants is driven by the interrelationships between individuals, implying the dependence of community structure on product structure. The interdependence between products and organizational structures has also been highlighted by Sosa et al. [31], and Yassine and Wissmann [26] for traditional hierarchical product development.

The literature on the interdependence between products and organizations is primarily in the software development domain, and is focused on determining whether the product structure matches with the organizational structure or not. The rationale is that in order to address the technical dependencies in the product, the corresponding individuals working on those aspects of the product must communicate with each other. This is a hypothesis in the organizational science literature, and is referred to as the “mirroring hypothesis” [32]. The extent of the similarity between the product structure and the community structure is referred to as socio-technical congruence [33]. Existing literature [33] indicates that there is a positive correlation between the socio-technical congruence and productivity of the organization.
Detailed understanding of the interdependence is particularly important for directing community-based virtual production and innovation because no community structure is imposed at the beginning. It evolves as new participants join the effort. Similarly, the product architecture also evolves over time. The interdependent co-evolutionary dynamics of products and communities is not limited to information-based systems such as software but is also applicable to physical systems. In this section, we analyze the interdependence between the evolution of product structures and the community structures (see Figure 1.7). The results in this section are based on the analysis of Drupal product and community structures, further detailed in Le and Panchal [34].

The approach adopted in this section is as follows. Product and communities are jointly modeled as a hybrid network with two types of nodes and three types of links. The first type of nodes is the product modules (sub-systems or components). The product modules are linked through spatial, structural, material, energy, and information dependencies between the modules. In software products, nodes can refer to classes, files, or functions. The second type of nodes is the individuals that form the community. The links between individuals represent communications between them. Within virtual product development, this information exchange can be through online chat, modification requests, emails, and blog links. The information about these links can also be obtained through questionnaires and surveys of the individuals.

In addition to the two types of links described above, links can also be established between the modules and the participants. These links represent the participation of an individual on a product module. These links are referred to as the “participation links” which result in a bipartite network, as used in Section 1.2.1. Based on the participation links, we can define a module set of an individual.
as the set of modules that a given individual works on, and a participant set of a module as the set of individuals linked to a module. The three link types (individual-individual, module-module, and individual-module) described above are utilized to generate hybrid network models representing products, communities and participation simultaneously, as shown in Figure 1.8. The communication links between individuals are weighted to indicate the amount of communication, and the weights on the interfaces between modules represent the strengths of interfaces. The participation links are not weighted.

The hybrid network is used to test specific hypotheses derived from the mirroring hypothesis. Examples of hypotheses include: a) if the number of interfaces of product modules increases then the number of new communication links between people working on those modules also increases, and b) if the number of a participant’s communication links increases, the number of new interfaces of corresponding modules also increases. The approach involves comparing consecutive versions of products and communities. To validate the second hypothesis, each participant is selected in turn and the number of newly created communication links of that participant with other participants is determined. Similarly, the number of newly created interfaces between modules within the participant’s module set and the modules outside the modules set is calculated. The hypothesis is valid if the number of newly created communication links increases with an increase in the number of newly created interfaces of corresponding module sets.

Sample results of the analysis are shown in Figure 1.9. The results are shown for the product evolution from version 5 to 6 and from version 6 to 7. The top part of the figure is generated by determining the number of new interfaces of each module for consecutive versions of the product. For each product module, the participant set is determined and the number of new communication links of the participant set is evaluated (see illustration on top left in Figure 1.9). In the bottom figures, the number of new interfaces is plotted against the average number of new communication links between the corresponding pair of participant sets. Both the top and the bottom figures show that as the number of new interfaces between product modules (files) increases, the number of communication links between individuals also increases. Based on the results, it is concluded that for Drupal, product structures significantly influence the communication patterns between individuals. Similar analysis is also carried out to determine the effect of new communication links on the product structure. However, the impact of communication patterns on the product structures is not evident. For detailed analysis of the results, see [34].

Cluster comparison

If there are dependencies among a set of product modules, then the participants working on those modules are expected to closely collaborate with each other. Hence, they should cluster together as communities within community networks. To validate this hypothesis, Le and Panchal [34] propose an approach based on cluster comparison. In this approach, the communities anticipated based on the product dependencies are determined and compared with the actual communities. The anticipated communities can be derived based on the knowledge of the product network and the participation links.

The actual communities are determined by clustering the community network shown in Figure 1.8. The anticipated communities can be determined by utilizing clustering algorithms on the product networks and determining the corresponding clusters of participants contributing to different product clusters. Alternatively, the anticipated communities can also be determined by utilizing bipartite clustering algorithms on the bipartite network with individuals and modules linked through participation links. Within the latter approach, communities of participants are identified based on the participation links only. The anticipated communities are then compared with those obtained by clustering the community network. Comparisons between clusters are carried out for each version to determine the evolutionary characteristics.

The extraction of clusters can be performed using different algorithms. Rytsareva and cou-
Authors [35] use the modularity-based algorithm proposed by Newman-Girvan [36]. Using this method, a modularity function is determined for each partition. The best partition of the network is the one that maximizes the modularity across all possible partitions. The Louvain method [37] is used to reduce the computational effort in finding clusters. The modularity-based approach proposed by Barber [38] is used for bipartite networks. Modularity-based community detection methods are used because they generate good quality clusters at low computational cost, the user does not need to specify the number and the size of clusters, and these algorithms are available in a number of commercial and open-source network analysis applications.

**TABLE 1.2:** Comparison of the Rand Index quantifying the overlap between communities

<table>
<thead>
<tr>
<th>Drupal version</th>
<th>Product network clustering</th>
<th>Bipartite network clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.6178</td>
<td>0.7054</td>
</tr>
<tr>
<td>5</td>
<td>0.6449</td>
<td>0.7148</td>
</tr>
<tr>
<td>6</td>
<td>0.6566</td>
<td>0.7161</td>
</tr>
<tr>
<td>7</td>
<td>0.6770</td>
<td>0.7331</td>
</tr>
</tbody>
</table>

The overlap of clusters between two partitions of the network is measured using the Rand index [39], which is a measure of the similarity between two data partitions. The Rand index can be expressed as $R = \frac{a + b}{\binom{n}{2}}$ where $a$ is the number of pairs of participants who are in the same cluster in both partitions being compared, $b$ is the number of pairs of participants who are in different clusters in both partitions; $n$ is the total number of participants. Table 1.2 shows the extent of overlap between the anticipated communities and the actual communities, measured using the Rand index.
It is observed that the overlap between the communities is significant ($R > 0.6$). Moreover, the overlap increases as the product evolves from version 4 through 7. This indicates that even though the alignment is not enforced by any higher level authority, the community structure gets increasingly aligned with the product. The techniques discussed in this section can help communities and product development organizations in determining the extent of socio-technical congruence and the aspects of the product where congruence is high.

### 1.3 Computational Modeling of the Effects of Individual Decisions

As discussed in Section 1.1, within community-based product development, the product evolves as a result of the independent actions of the participants. In this section, we present a deductive study of the effect of participants’ decisions on the product’s evolution. We start with a theory about how rational decision-makers would make decisions, and deduce the evolutionary trends in the product. The participants decide whether to contribute to the product development or not, which modules to contribute to, and when to contribute. These decisions are generally driven by various intrinsic and extrinsic motivations [40]. The participation decisions are also affected by the decisions made by other participants. Hence, a game theoretic model of decisions made by participants is used.

The individual contributions affect the rate at which the product grows. The rate of the product evolution is further influenced by the complexity of the product. Due to the interdependencies between different modules within a product, changes made to one module may result in changes in other modules, thereby affecting the overall development of the product. Since these changes are made through independent contributions of the participants to self-selected modules, the decisions made by individuals and the product structures must be considered simultaneously to determine the evolutionary characteristics of the products. Our goal in this section is to present one such computational model. Specifically, we present an agent-based model that accounts for the decisions made by the individual participants and the interdependencies between the product modules. The inputs and the outputs of the model are illustrated in Figure 1.10.

**FIGURE 1.10:** Overview of the agent-based model of community-based product development
1.3.1 Agent-based Model of Community-based Product Development

The proposed model consists of two types of entities: the product and the participants involved in developing the product. These are described next.

Product model

A product is represented as a directed weighted graph consisting of a set of modules as nodes with interdependencies between them. Each module is associated with two attributes: a) percentage completion, and b) growth rate. Percentage completion quantifies the extent of completion of a module at a given time. A module is complete when the percentage completion reaches 100%. The growth rate represents how fast the module grows. It is used to quantify the differences among modules in the amount of effort and time required for their development. A high growth rate represents a module that can be developed faster than a module with a low growth rate. The growth rate is expressed as a percentage value. For example, a module with a percentage completion of 10% and a growth rate of 5 will reach a percentage completion of 10.5% in the following time step and a percentage completion of 11.025% after two time-steps. To initiate the growth of modules, the percentage completion is set to a small value (0.01%).

A link between two modules represents the interdependencies between them. Each link is associated with an attribute, percentage rework, which quantifies the amount of rework that must be carried out on a target module if there are changes in the originating module. Percentage rework is modeled after the work of Cho and Eppinger [41]. A simple example is shown in Figure 1.11. In this figure, a link originates from Module 1 and terminates at Module 2, which indicates the dependency of Module 2 on Module 1. When Module 1 changes, the amount of rework on Module 2 is equal to the extent of completion of Module 2 multiplied by the percentage rework from 1 to 2. If the percentage completion of Module 2 is 30% and the percentage rework of the link is 5%, then any change in Module 1 will result in a rework of 1.5. Hence, the resulting percentage completion of Module B will be 28.5%.

Participants and their strategies

The participants in the model are modeled as decision-making agents. The number of participants in the model is represented by N. Participants make decisions about whether to participate or not based on their preferences for benefits and costs associated with an activity. Each agent has some benefits in participating in the product development process, which may be due to personal satisfaction, ability to use the product for personal use, ability to gain new knowledge, or the possibility of gaining recognition in the community [40]. In this model, all possible types of benefits are accounted for using a single parameter called value (\( v_p \)) to a participant \( p \). 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 \( p \) is also modeled using a single parameter (\( c_p \)) in the model.

Each participant also has a decision model that represents whether to participate and contribute to the product development process or not. The decision is also dependent on the decisions of other participants. Hence, a game-based model of the interaction is used. The decision model used in this paper is based on the involuntary altruism model presented by Baldwin and Clark [42]. The involuntary altruism model is based on a game called public provision of private goods, which was first applied to open source software development by Johnson [43].

The normal form of the game of involuntary altruism, as presented by Baldwin and Clark in [42], for two participants \( P_1 \) and \( P_2 \) is shown in Table 1.3. Each participant has value (\( v \)) and cost (\( c \))
TABLE 1.3: Normal form of the game of involuntary altruism in a two-player scenario

<table>
<thead>
<tr>
<th>P₂ does not contribute</th>
<th>P₂ contributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁ does not contribute</td>
<td>0, 0</td>
</tr>
<tr>
<td>P₁ contributes</td>
<td>(v - c), v</td>
</tr>
<tr>
<td></td>
<td>(v - c), (v - c)</td>
</tr>
</tbody>
</table>

associated with a product development task, and has the option 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 \( \alpha^* = \left(1 - \frac{c}{v}\right) \). If there are N participants in the game, the mixed strategy equilibrium is the strategy where each participant contributes with a probability:

\[
\alpha^*_N = 1 - \left(\frac{c}{v}\right)^{\frac{1}{N}}
\]  

Additional details of the game are provided in [42]. This mixed-strategy Nash equilibrium (where each participant contributes with a probability of \( \alpha^*_N \)) is used in the agent-based model for modeling the participants’ decision making strategy.

1.3.2 Assumptions and Model Parameters

The model is implemented in Netlogo [44], which is an easy to use agent-based modeling tool. The agent-based model is initialized using N participants and their initial parameters (costs and values). The model is executed in cycles. In each cycle, all the agents make their individual decisions using the mixed equilibrium strategy once. Each agent decides whether to contribute to the product development effort or not. During each cycle, each agent decides in favor of contribution with a probability of \( \alpha^*_N \). During a cycle, if an agent decides in favor of contribution, then he/she works on the least developed module (say \( i \)) in the product, and the module grows by a factor \( g_i \), the module’s growth rate.

The value (v) associated with each agent can change with time. It is assumed that as the product grows, the value increases. The cost of participation to a participant is assumed to be constant throughout the execution of the model. The cycles are continued until all the modules reach a completion level of 100%. The cycles correspond to the product evolution process and the number of cycles to completion indicate the time required for completion of the product through the evolutionary process.

The model is used to analyze various different outputs of the model. 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 product. The distribution of effort among the participants is important because it shows how the contribution varies among the participants.
Assumptions: The model presented in this chapter represents an abstraction of the real community-based product development, and is based on various 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. It is also assumed that the growth rate for 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 during a given interval. It is assumed to be based on the mixed equilibrium strategy in the game of involuntary altruism. It is assumed that a decision maker knows the the values and costs for all other decision makers. When a participant decides to contribute on the product, it is assumed that he/she selects the least developed module to work on. Only one participant is assumed to 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 checking out the module, as is the case in software development. This prevents multiple participants from making conflicting changes to a particular module. All the participants are assumed to have the same expertise. There is also no difference between the modules in terms of the expertise required. These assumptions define the scope of applicability of the model. North and Macal [45] highlight that agent-based models should be developed in an incremental manner so that the emergent phenomena can be first identified and understood before adding the details and complexity of real scenarios. The model is simple enough to aid in understanding the implications of various parameters in the model and simulating the dynamics of the processes. The model can be refined in the future by relaxing some of these assumptions.

1.3.3 Outputs of the Model

The agent-based model can be used to simulate various aspects of community based product development. These aspects include the evolution of products, the distribution of contributions by the participants, the effects of incentives on the rate of product evolution, and the effects of product architectures. In this section, we present some of the results from the model. Further details of the results are provided in [27,46].

Evolution of the product

An example of the evolution of the product with the number of cycles is shown in Figure 1.12. This figure is for the product shown in Figure 1.10. The product has two types of modules: core modules and peripheral (external) modules. This structure is used to represent the core-periphery network structures observed in the products developed by communities [47]. The core modules significantly influence the peripheral modules but the effects of changes in peripheral modules on the core modules is small. Modules 0, 1, and 2 are the core modules and Modules 3 through 8 are the peripheral modules.

In Figure 1.12, it is observed that initially, the rate of product development is small. This corresponds to the development of the core modules. Any changes in the core modules result in the rework of peripheral modules. Hence, the overall completion of the product is slow. After the core modules are sufficiently developed (at around 6000 cycles), the rate of evolution of the product...
increases significantly. This is primarily because after the core modules are developed, the development of the peripheral modules takes place. The results indicate that even though there are no clearly defined project phases defined by the participants, the product development shows various phases of development: a) evolution of the core modules (0-2), b) development of the external modules (3-6) that depend on the core modules only, and c) the development of the second layer of external modules (7 and 8).

Distribution of effort

Figure 1.13 shows a histogram of the number of participants as a function of the amount of effort invested throughout the product development. From the figure, participants can be categorized into two types based on the extent of their contribution: primary contributors and secondary contributors. The primary contributors participate throughout the project and the amount of effort they invest in the project is significantly higher than the rest of the contributors. The secondary contributors join late in the product development process. Their main contribution is on the peripheral modules. It is observed that the number of primary contributors is significantly smaller than the number of secondary contributors. In the figure shown, about 80% of the contributors are secondary contributors. Only about the 20% participants are primary contributors. This trend is similar to the contributions in open source software development [48].

Effects of incentives

The model can be used to analyze the effects of incentives on the product development. As discussed in Section 1.3.2, the value from participation, \( v \), is assumed to increase linearly as the extent of the product development increases \( (v = k \times (% \text{ completion})) \), where \( k \) is the rate at which the value increases with the development of the product, referred to as the value function coefficient. The parameter \( k \) can be used to model the effect of additional incentives for participation. Figure 1.14 shows the effect of increasing \( k \) on the total effort and the distribution of efforts among participants. As \( k \) increases, the total effort required for the completion of the product reduces. The amount of time required for the development of the product also reduces. After increasing \( k \) beyond a certain value (\( k = 30 \) in this case), any further increase does not correspond to further reduction in the total effort. The value function coefficient also affects the distribution of effort. This is shown in the figure using two histograms for \( k = 2 \) and \( k = 50 \). When \( k \) is low, most of the participants are secondary participants. On the other hand, when \( k \) is high, the number of primary participants increases significantly. For \( k = 50 \), the number of primary participants is almost the same as the number of secondary participants.
Effect of Product Modularity on Community-based Product Development

Modularity of the product is defined in various different ways, as discussed by Gershenson and coauthors [49]. In general, modularity refers to the degree to which a product’s architecture is composed of modules such that the interactions within the modules is significantly greater than the interactions among different modules. Various measures have also been proposed in the literature to quantify product modularity. Two examples include singular-value modularity index (SMI) and the degree of modularity (DOM). To calculate SMI, singular value decomposition (SVD) is performed on the matrix representation of the product network. The SMI index measures the average, weighted decay rate of sorted singular values in the system [50]. In this section, we use the Degree of Modularity (DOM) measure because of its simplicity. DOM involves first calculating the modularity of individual modules using the modularity of each module. Module modularity is defined as the level of independence of a module from the other modules within a product. Highly independent modules have higher module modularity. Mathematically, module modularity [51] for a module $i$ is represented as:

$$M_i = \frac{(m - 1)x_{\text{max}} - x_{i+}}{(m - 1)x_{\text{max}}} \tag{1.2}$$

$$x_{i+} = \sum_{j=1, j \neq i}^{m} x_{ji} \tag{1.3}$$

In the equations above, $x_{ji}$ is the dependency represented from the $j^{th}$ module to the $i^{th}$ module, $x_{\text{max}}$ is the maximum value of $x_{ji}$, and $m$ is the number of modules. The DOM of a product is the average level of modularity of modules, mathematically represented as:

$$M = \frac{\sum_{i=1}^{m} M_i}{m} \tag{1.4}$$

Le and Panchal [27] utilize the agent-based model to analyze the effect of degree of modularity on the time required for product evolution. The model is utilized to model the evolution of a cell phone. The results are presented in Figure 1.15. It is observed that as the degree of modularity increases, the number of cycles required to complete the product evolution decreases. Hence, modularity is helpful in community-based product development also. When the degree of modularity is less than a certain threshold (0.89), the development of the overall product does not reach 100%. This is because the amount of rework is greater than the growth of the modules. The figure also shows that for the DOM ranging within [0.89 0.91], the decoupling results in a rapid decrease in the evolution time of the product. Beyond the DOM of 0.91, the evolution time decreases slowly.

The authors also analyzed the effects of decoupling different modules on product evolution. The effects of three different decoupling sequences are presented. The degree of modularity versus evolution time for three decoupling sequences ($S_1$, $S_2$, and $S_3$) are plotted in Figure 1.16. It is observed that even for the same DOM values, the evolution times can be different. This is because DOM is a measure of product structure that depends only on the dependencies between modules. It does not account for the dynamics of community-based product development processes. Using the knowledge, the best decoupling sequence can be used to reduce the development time.
In summary, even a simple agent-based model such as the one discussed in this section can be used to gain significant insights into the socio-technical aspects of the community-based product development. According to Tesfatsion [52], agent-based models can serve various purposes in understanding socio-technical systems. First, agent-based models can provide empirical understanding into why certain system level behaviors are observed. Second, they can aid in normative understanding of the system and to discover new designs. Third, they can provide qualitative insights and aid theory generation by systematically examining the system behavior under different initial conditions. Finally, they can provide methodological advancements through new tools and methods for rigorous study through computational experiments. In this chapter, the model has primarily been used to generate qualitative insights by starting with a game-theoretic model of participants’ decision making strategies. The goal is not to create a predictive model that perfectly models all the nuances of community-based product development scenarios.

1.4 Research Outlook

Despite the emergence of various communities for developing specific products, fundamental research from a systems realization standpoint towards understanding how communities can develop complex products and systems is still in its infancy. Community-based product development poses unique challenges due to the interrelated social and technical aspects. Hence, further research in this area must account for the socio-technical aspects in a holistic manner. In this section, we discuss some of the promising research directions for the Computers and Information in Engineering (CIE) and engineering design research communities.

1.4.1 Theoretical and Empirical Research

In this chapter, we started with the core research question driving our efforts: “How can complex systems be effectively engineered by self-organized communities of self-directed individuals?” Panachal and Fathianathan [53] highlight some of the research issues including coordination between stakeholders, nature of teams, collective decision-making, system architectures, and information and computation. In this chapter, two research efforts being undertaken by the authors are presented. The first effort is on using network-based approaches to analyze the structure and interdependent co-evolution of products and communities. The second effort is on using game-theoretical models of participant behaviors to simulate product evolution.

The first effort utilizes data from existing projects to empirically validate hypotheses (e.g., mirroring hypothesis) while the second effort relies on theoretical models of behavior to generate trends and hypotheses. These efforts represent initial steps towards gaining fundamental understanding about community-based product development. Further research in this area would integrate the inductive and deductive research approaches to gain in-depth knowledge of the factors affecting community formation and evolution. There is significant opportunity to leverage existing work in
Research Outlook

21

diverse fields such as sociology, empirical economics, and game theory, and to apply the knowledge in product development.

Community-based product development is an ideal starting point for socio-technical analysis within engineering design research because of the readily available data related to communities and products (such as open source software and hardware). Gathering such data for socio-technical analysis of hierarchical organizations is highly challenging at large companies due to the competitive nature, and is difficult at defense organizations due to the security issues.

Finally, further research is needed to understand scenarios that combine bottom-up and top-down product development approaches. Throughout this chapter, community-based product development has been viewed from a strict bottom-up perspective and firm-based product development has been viewed from a strict top-down perspective. The differences have been highlighted in Table 1.1. However, these two extremes are idealizations of two different scenarios. Real product development scenarios may integrate both these extremes in different proportions. For example, open source communities may have some organizational aspects that are better represented as hierarchical organizations. Similarly, firms may employ bottom-up techniques to improve their product development. Additional research is required to understand the effects of integrating both these models. It also indicates that the work presented in this chapter is not only relevant to community based product development, but also to product development firms.

1.4.2 Platforms for Community-based Physical Product Development

The studies on community-based product development are particularly prominent for software because of easier quantification of product dependencies, relative ease of gathering the data, and well-documented communications. These dependencies can be automatically extracted and network representations of products can be automatically generated. The strengths of dependencies can also be quantified by techniques such as measuring the number of times a function is called by another function. In contrast, the interfaces within physical products can be of different types, e.g., spatial, structural, energy-related, material-related, and information-related. For physical products, such dependencies are generally extracted by interviewing engineers and design experts. The strengths of relationships are also more difficult to quantify in physical products. Additionally, the software development processes are generally better documented than physical product development processes due to the presence of tools such as concurrent versioning systems and bug management systems.

Due to the lack of dedicated platforms for community-based development of physical products, open-source hardware projects (e.g., RepRap 3D printer) are currently being managed and coordinated on platforms, such as GitHub, which were primarily developed for open-source software. These platforms allow participants to share files (such as CAD models and STL files), which are the outcomes of the design process. However, the knowledge used for the design itself is not captured.

Currently, community-based innovation has primarily been limited to software products. However, to support community-based development for physical products, there is a need to develop new tools that integrate some of the information management functionalities provided by the product lifecycle management (PLM) systems with the community building functionalities provided by the forges such as SourceForge and GitHub (see Figure 1.17). Dassault Systemes has taken initial steps in this direction by exploring a platform called 3DSwYm [54]. The platform’s initial functionalities are similar to social networking platforms, supporting forum-style communication between community members, and sharing of CAD data. However, in order to truly enable physical product development, further research is needed.
1.4.3 Community-based Product Development in Education

Emerging paradigms of engineering design also result in new demands on how the next generation engineers are educated. While many approaches have been reported in engineering design education literature, the majority are focused on educating students for efficiently operating in today’s enterprises operating in the hierarchical product development paradigm. Educational approaches and courses for preparing the students for the emerging global-innovation based community-based innovation are virtually non-existent. Engineering students are currently not being prepared for a “collaborative, contribution-based environment where the role of an enterprise shifts to orchestration and facilitation of endeavors between groups of individuals” [55]. This requires a transformation in engineering design education towards educating a new generation of engineers, who would require new competencies and skills than those focused in current engineering and design courses.

1.5 Acknowledgments

The authors gratefully acknowledge the financial support from the National Science Foundation through the CAREER Grant CMMI-1265622. The authors also thank their collaborators, including HaoYun Huang, Inna Rytsareva, and Ananth Kalyanaraman, who have provided valuable expertise and insights during this work.

References

References


References


