ANALYSIS OF THE STRUCTURE AND EVOLUTION OF AN OPEN-SOURCE COMMUNITY

Hao-Yun Huang, Qize Le and Jitesh H. Panchal*

School of Mechanical and Materials Engineering
Washington State University, Pullman, WA 99163 (USA)

ABSTRACT
Open-source processes are based on the paradigm of self-organized communities as opposed to traditional hierarchical teams. These processes have not only been successful in the software development domain, but are increasingly being used in the development of physical products. In order to successfully adapt open-source processes to product realization there is a need to understand how open-source communities self-organize and how this impacts the development of products. Towards the direction of fulfilling this need, we present an analysis of an existing open-source community involved in developing a web-based content-management platform, Drupal. The approach is based on the analysis of networks using techniques such as social network analysis, degree distribution, and hierarchical clustering. Openly available information on the Drupal website is utilized to perform the analysis of the community. The data is transformed into two weighted undirected networks: networks of people and networks of Drupal modules. Both the structure of these networks and their evolution during the past six years are studied. Based on the analysis, it is observed that the structure of the Drupal community has the characteristics of a scale-free network, which is similar to many other complex networks in diverse domains. Key trends in the evolution of the networks are identified. Finally, a predictive model is presented to provide potential explanations for the observed structures and evolutionary trends.

Keywords: Open-source processes, communities, product development, evolutionary networks

* Corresponding Author. E-mail: panchal@wsu.edu
1. FRAME OF REFERENCE: OPEN-SOURCE PROCESSES

1.1. Open-Source in Software and Hardware

During the past two decades, open-source processes have gained significant popularity in the software development domain. Various products such as Linux, Apache, and Mozilla have shown that open-source processes can be as successful as the processes followed by traditional organizations. The concept of open-source has not only been used in the software development domain but has also been recently implemented in physical product development. Examples include open-source 3D printers [1], electronics prototyping platforms [2], cell phones [3], cars [4, 5], prosthetics [6], machine tools, robots, and other socially-relevant design projects [7]. In physical product development, open-source refers to the openness of information such as design details, schematics, CAD models, bills of materials, associated software, etc. The success of open-source processes in physical product development is driven by the fact that physical products are also information products during the design phase, and has accelerated due to the reducing cost of 3D printing capabilities.

Open-source processes emerged in software products earlier than in physical products because software products have characteristics particularly suitable for open-source processes. The development of both software products and physical products can be divided into four phases: design, manufacturing, distribution, and upgrade. In the design phase, both software and physical products can be viewed as information-based products [8] defined using requirements, functions and detail designs, which can be recorded electronically. In the manufacturing phase, software products can be manufactured (programmed) by individuals using computers, while physical products need to be manufactured by specific physical tools and machines. In the distribution phase, software products can be shared through the Internet at (almost) zero cost, but physical products need to be transported from one site to another. In the upgrade phase, software products can be upgraded by simply re-compiling the upgraded code whereas physical products need to be re-manufactured. By comparing software and physical products in the four product development phases, it is clear that software products are easier to manufacture, distribute and upgrade than physical products, even by individuals without many resources. Hence, the differences between software and hardware products are clear. However, there are still some
fundamental similarities in the way open-source processes are utilized both in software and hardware. We are particularly interested in the similarities in the way open-source communities emerge and the products gradually evolve.

Further, emerging technologies are making open-source processes more applicable to physical product development. For example, with the increasing availability of 3D printing capabilities, open-source processes have also started gaining popularity in physical product development [9]. In the design phase, physical products are also information-based products whose documents can be easily shared by individuals. So the open-source processes can be applied during the design phase, examples include Open Source Car [4, 5], and Open Prosthetics Project [6]. With the development of the rapid prototyping technology, open-source processes can also be extended into product manufacturing. A 3D printer is a convenient tool for individuals to prototype physical products using CAD models. In addition, some physical products such as electronic hardware can be manufactured directly from electronic design documents. These designs can be downloaded from the Internet. In these cases, the manufacturing and upgrading phases can also be carried out through open-source processes. In a recent article, Anderson [10] projects that open-source will revolutionize the way in which innovative products are designed and developed. Given the increasing adoption of open-source as a new paradigm for product realization, it is becoming important to understand its underlying dynamics.

1.2. Open-Source Processes - A Collective-Systems Perspective

Open-source processes are significantly different from traditional product development processes because they are based on bottom-up design by self-organized communities as opposed to top-down design by hierarchical organizations. They are driven by participants choosing their activities based on their own goals and interests instead of being driven by top-down hierarchical control as in the case of traditional product development. Open-source products are always under continuous development and evolution. Similarly, the open source communities are under continuous evolution. Hence, traditional hierarchical models of products and processes prevalent in systems engineering are limited in providing insights about the analysis and design of open-source processes. Further insights can be gained by viewing open-source process as collective systems.
Collective systems, according to Tumer and Wolpert’s definition [11], consist of interacting “agents” who strive to maximize their private utility through their local behaviors. The system’s global performance is measured in terms of a world-utility function. In open-source processes, the participants may be considered as agents with their own goals. These goals may range from the need to achieve certain functionality to gaining recognition [12]. Interactions between the agents are through the (software or hardware) products they work on. The world-utility function can be defined in terms of the development of the product. Tumer and Wolpert [11] and Namatame [13] highlight that collective systems are associated with two types of problems: forward (analysis) problem and inverse (design) problem. The forward problem involves finding the system-level performance based on a set of agent behaviors and their interactions. On the other hand, the inverse problem is concerned with the determination of “suitable” agent behaviors and interactions to achieve desired system-level performance. Recently, significant efforts have been devoted by economists and social scientists on the forward problem related to open-source processes, with the goal of understanding the private utilities and interactions of agents in successful projects [12, 14-17]. Their focus is on answering questions such as: Why do people contribute for free? What motivates people to contribute? Why is open-source successful? However, the inverse (design) problem has received little attention. The inverse problem is particularly important for the successful implementation of open-source processes within product development, and hence, of relevance to design engineering research community.

One of the key factors in the inverse (design) problem in open-source is the structure and evolution of the community and their effect on the evolution of products. It is known that “teams with the same composition of members can perform very differently depending on the patterns of relationships among the members” [18]. The effect of organizational structure on the product is well recognized in traditional product development, [19]. According to Conway [20], “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”. Hence, the organizations strive to align the organizational structures with the product structures. However, in the case of open-source processes, no organizational structures are imposed at the beginning of the process. The structure of the organization evolves as new participants join and collaborate with existing participants. The collaboration between different participants
is based on the product structure and is driven by the dependencies between subsystems, implying the effect of product structure on community structure. Hence, in open-source processes, the products and communities undergo interdependent co-evolution. In order to successfully utilize open-source processes for product realization, we believe that the knowledge of this interdependent co-evolution is crucial. The knowledge can be gained by understanding a) the structure and evolution of communities, b) the structure and evolution of products, and c) the interdependence between structures and the evolution of communities and products.

The focus in this paper is on the first aspect, i.e., understanding the structure and evolution of open-source communities. The key questions are:

- What is the structure of open-source communities?
- What is the evolutionary behavior of these communities?
- What are the underlying local behaviors that result in their evolutionary behavior?

Our goal in this paper is to take a step towards answering these questions. The developed insights can be used to address the inverse (design) problem in open-source processes. The approach used in this paper is to focus on a specific open-source project, and to analyze the structure and evolution of the associated community. A predictive model is proposed for the underlying local behaviors of individuals that possibly explain the resulting structure and evolutionary behavior of the community.

The structure of the paper is as follows. In the following section, relevant literature on the structure and evolution of communities is discussed and the gaps are identified. The proposed approach, involving the analysis of an existing open-source community, is discussed in Section 3. The results from the execution of the approach for a specific open-source community are presented in Section 4. A simple model to simulate the bottom-up evolution of open-source communities is presented in Section 5. Finally, closing thoughts are presented in Section 6.

2. REVIEW OF RELEVANT LITERATURE

Existing literature on open-source processes is primarily focused on open-source software (OSS) development because of highly developed processes, large number of communities, and significant amounts of data on OSS development. A general discussion of the factors affecting the success of OSS development is presented by Weber [21]. OSS is a public good provided by volunteers – the "source
code” used to generate the programs is freely available to read, use and modify [22]. An OSS project is typically initiated by an individual or a small group with ideas which can realize their intellectual, personal, or business interests [23]. Various researchers have presented empirical and quantitative studies on the structure of OSS communities based on the data from existing OSS projects. Raymond [24] describes the Linux development community as a “Bazaar”. Cox [25] presents initial thoughts of “town councils” structure in OSS community based on Linux 8086 project. The author conceptually illustrates the community structure for Linux 8086 project. Weber [21] discusses different types of organization structures in various OSS projects. For example, the community structure of the Linux project reflects a pyramid structure whereas the community structure of the BSD project is represented as concentric circles. The structures concluded by Weber are based on direct observation of communities without rigorous mathematical analysis. Crowston and Howison [16] discuss community centralization in OSS development communities by analyzing data from the bug-tracking system in SourceForge. The authors demonstrate that the community centralization or decentralization is not a characteristic of OSS projects. Crowston and Howison [26] later analyze hierarchy and centralization of the OSS communities of Apache, Savannah and SourceForge by employing social network analysis (SNA) measures. They conclude that large projects are less centralized and hierarchical, as compared to smaller projects. Xu and Madey [27] discuss role distribution and degree distribution in the SourceForge community. Xu and co-authors [28], and Gao and Madey [29] 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. They also observe small-world [30] and scale-free [31] network properties in the SourceForge community. Xu et al. [32] present the structure of OSS communities by calculating the modularity of networks, which is defined as the fraction of edges within communities minus the expected value of the same quantity if edges fall in a random network, and analyzing the groups that exist in the SourceForge network.

The studies discussed above are focused on analyzing the community structures. Some efforts have also been carried out on the evolution of the communities. White et al. [33] introduced the analysis of social structure over time using snapshots of data. Nakakoji et al. [34] discuss the evolution of communities in the form of role changes of the members in OSS communities, and conclude that there
are two factors determining the evolution of OSS communities: the existence of motivated members, and the social mechanisms of communities. Weiss et al. [35] trace the evolution of a community by taking snapshots of its membership at regular intervals and establish a major hypothesis that OSS communities grow through a process of preferential attachment. de Souza et al. [36] represent a framework for software modules and software developers, and study software project communications at two points in time. The authors analyze the movements of developers across different modules of software systems. Howison et al. [37] 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. Wiggins et al. [38] analyze the dynamics of OSS development communities and find a variation in communication centralization and decentralization in the OSS development communities. OSS is a special case of mass-collaborative product development [39]. Panchal [40] uses the agent-based technique to model the evolution of products in such bottom-up processes. Panchal later extends the model to explore the co-evolution of communities and products [41]. Le and Panchal [42] study the effect of product architecture on the evolution of products in mass-collaborative processes.

Existing studies are limited in the analysis of open-source communities because of the lack of:

a) simultaneous analysis of structure and evolution,

b) comprehensive analysis of the different aspects of the community structure,

c) predictive models to explain the evolutionary behavior of the communities, and

d) integrated analysis of the evolution of communities and the products.

In this paper, we perform a comprehensive study of the community structure and evolution. The study is based on the data from Drupal [43], which is a software tool for building community-based websites. The reason for studying Drupal is that there is freely available data, detailed documentation, and a highly developed community associated with this project. Besides, Drupal is widely used as a basic framework of web development and is a very successful community-based website building tool. The authors are also studying the Sourceforge community to assess the generality of the results. The objective is to understand the evolutionary characteristics of open-source projects that span both software and physical products. The study of OSS projects will lead to a) fundamental knowledge which can be applied to both
software products and physical products based on their commonalities in the design phase, and b) new
techniques enabling individuals involved in manufacturing, upgrade and distribution phases.

3. APPROACH ADOPTED FOR THE ANALYSIS OF STRUCTURE AND EVOLUTION OF OPEN-
SOURCE COMMUNITIES

The approach adopted is this paper is based on network analysis. The communities are modeled as
social networks, defined by participants connected by collaboration links. A social network is defined by a
set of interrelated social entities. Social network analysis has been used to analyze diverse systems such
as author and paper networks [44], online communities on websites such as Yahoo and Flickr [45], and
OSS communities.

3.1. Data Collection

In the first step, raw data about the participants and the product modules they contribute to are
extracted from the database. This raw data is used to derive information about the relationships between
individuals and their related modules. The raw data can be in the form of a simple table which shows the
participants and the modules. 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.

<table>
<thead>
<tr>
<th>Commiters for Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
</tr>
<tr>
<td>Last commit</td>
</tr>
<tr>
<td>First commit</td>
</tr>
<tr>
<td>Commits</td>
</tr>
<tr>
<td>Scott Reynolds</td>
</tr>
<tr>
<td>2 weeks ago</td>
</tr>
<tr>
<td>35 weeks ago</td>
</tr>
<tr>
<td>13 commits</td>
</tr>
<tr>
<td>Thomas_Zahedda</td>
</tr>
<tr>
<td>10 weeks ago</td>
</tr>
<tr>
<td>11 weeks ago</td>
</tr>
<tr>
<td>4 commits</td>
</tr>
<tr>
<td>skilthin</td>
</tr>
<tr>
<td>16 weeks ago</td>
</tr>
<tr>
<td>2 years ago</td>
</tr>
<tr>
<td>291 commits</td>
</tr>
<tr>
<td>Jayesh</td>
</tr>
<tr>
<td>30 weeks ago</td>
</tr>
<tr>
<td>1 year ago</td>
</tr>
<tr>
<td>379 commits</td>
</tr>
<tr>
<td>robertthoeglass</td>
</tr>
<tr>
<td>1 year ago</td>
</tr>
<tr>
<td>2 years ago</td>
</tr>
<tr>
<td>135 commits</td>
</tr>
<tr>
<td>louise</td>
</tr>
<tr>
<td>1 year ago</td>
</tr>
<tr>
<td>1 year ago</td>
</tr>
<tr>
<td>1 commit</td>
</tr>
</tbody>
</table>

Sample information obtained from Drupal website

<table>
<thead>
<tr>
<th>Obtained information assembled in a table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name</td>
</tr>
<tr>
<td>-------------</td>
</tr>
<tr>
<td>Module A</td>
</tr>
<tr>
<td>Module A</td>
</tr>
<tr>
<td>Module A</td>
</tr>
<tr>
<td>Module B</td>
</tr>
<tr>
<td>Module B</td>
</tr>
<tr>
<td>Module C</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Figure 1 - Illustration of the data-gathering step

Figure 1 (left) is a sample information table from www.drupal.org, which includes information about
the participants’ activities on a module named “Activity”. It contains information about the user name
("User" column), the first and last time each user made a revision of this module ("First commit" and "Last
Commit" columns), and the number of times each user revised this project (the “Commits” column). Each
Drupal module has an information table similar to Figure 1 (left). After collecting the information from all
the modules, the overall information table, as shown in Figure 1 (right), is generated. Community
networks are created from the information table to model the relationships within the community, as discussed in the following.

3.2. Generation of the Networks

The data presented in Figure 1 has a two-mode nature [46], i.e., there are two types of entities: people and modules. The two-mode data is first converted into a bipartite network consisting of two types of nodes - people and projects. The development of each module is a project in Drupal. Hence, each module represents a project node and each participant represents a person node in the network. Such a network is called a bipartite network, \( G = \{S_1 \cup S_2, E\} \), 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 and two-mode networks in the social network analysis literature [47]. An example of a bipartite graph is shown in Figure 2, where \( S_1 = \{a, b, c, d, e, f, g\} \) and \( S_2 = \{1, 2, 3, 4\} \). In the case of the OSS network, assume that \( S_1 \) and \( S_2 \) represent people and projects respectively.

The links in the network connect a person with a project. Hence, a link represents 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. In Figure 2, a binary bipartite graph is illustrated.

![Figure 2 - Example of a bipartite network and the two derived networks](image)

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 projects (\( S_2 \)) respectively. Figure 2 provides an illustration of graphs.
G₁ and G₂ derived from a bipartite graph. Two people in G₁ are connected by an edge if both of them share at least one project. Similarly, two projects in G₂ are linked if they have at least one common participant. The weights associated with edges E₁ represent the number of projects shared by a set of people. Similarly, the weights in graph G₂ represent the number of common participants shared by projects. These networks are also referred to as 1-mode S₁ and S₂ 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 x n matrix, where an element aᵢⱼ 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.

Some researchers promote the direct analysis of bipartite networks instead of projecting them to 1-mode networks [46-48]. However, the primary focus in this paper is on the community structure which is better represented as a 1-mode projection of the people-project bipartite network. Hence, the projected 1-mode graphs (G₁ and G₂) are primarily used for analysis in this paper.

3.3 Measures for Network Analysis

The adjacency matrix serves as a basic input for the network analysis process. After creating the adjacency matrix, the network properties are explored using different approaches. In this paper, the following approaches are used to determine the characteristics of the networks: social network measures, degree distribution, and hierarchical clustering.

3.3.1 Social Network Analysis (SNA) Measures

In order to characterize the key features of the OSS network, we use Social Network Analysis (SNA) measures [49]. SNA is a theoretical and methodological paradigm for examining complex social structures [50, 51]. The following SNA measures are used: degree, clustering coefficient, diameter, density, connectedness, and degree centrality.

a) **Degree** is the number of nearest neighbors of a vertex [52]. In an undirected graph, the degree of a vertex ν is the number of edges incident with ν and is denoted by deg(ν) or kν [53]. The degree distribution, P(k), of a network is defined as the fraction of nodes in the network with degree k [54]. 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 \( k_2 \) [55, 56]. The joint degree distribution is different from the conditional probability \( P(k_2|k_1) \) which measures the probability that a given node of degree \( k_1 \) is connected to a node of degree \( k_2 \). 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.

b) **Clustering coefficient** is the probability that two nearest neighbors of a vertex are also the nearest neighbors of one another [52].

c) **Diameter** is the largest distance between any two nodes of a connected network [53]. The diameter of a network indicates how "big" the network is. Hence, diameter provides information about how large the community is.

d) **Density** of a network is the average proportion of links incident with nodes in the network [57]. The density of a network ranges from 0 (if there are no links present) to 1 (if all possible links are present). A network with density of 1 is also called a complete network. The density of a network with \( n \) nodes and \( m \) links is:

\[
density = \frac{2m}{n(n-1)}
\]

(1)

e) **Connectedness** represents the ratio of the number of pairs in the directed graph that are reachable relative to the number of ordered pairs.

f) **Degree Centrality** measures the degree of inequality or variance in the network as a percentage of that of a perfect star network of the same size. For a network \( G=(V,E) \) with \( n \) nodes, the degree centrality \( C_D(v) \) for a node \( v \) is [58]:

\[
C_D(V) = \frac{\text{degree}(v)}{n-1}
\]

(2)

The degree centrality of a network \( G \) is:

\[
C_D(G) = \frac{\sum_{i=1}^{n} [C_D(v_i) - C_D(v_i)]}{n-2}
\]

(3)
where \( C_0(v) \) is the highest degree centrality of a node in the network. \( C_D(v_i) \) is the degree centrality of a node \( i \) in the network. The degree centrality of the entire community is also called degree centralization.

While there are dozens of measures for network data [51], we have chosen this subset because each measure in this subset provides important information about the community network structure for product development. 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 network diameter is a global network-level measure. It indicates how fast information can be spread within the entire community. A small diameter can describe the existence of the "small world phenomenon" [59] within the community. The density of a network indicates the average number of connections. High density of the community means that on average each participant has a large number of relationships that enable participants to communicate with each other. Connectedness, as the name implies, indicates whether there are disconnected sub-groups within the network. It again indicates whether information from one person can reach the other nodes. Finally, degree centralization measures the heterogeneity among participants in the community.

SNA tools are used for calculating the measures discussed above. The most widely used tools are Structure [60], UCInet [61] and Network Workbench [62]. Other SNA tools are discussed by Huisman et al. [63] and Freeman [58]. UCInet is used for the results presented in this paper.

3.3.2. Degree Distribution

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 [64]. Exponential random graphs have an exponential degree distribution. Scale-free networks have a power-law degree distribution [65]. Recent research
suggests that degree distribution in many real-world networks satisfies 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 significant insights not only about the structure of networks, but also about their evolution.

3.3.3. Hierarchical Clustering

Clustering is an approach for assigning a set of objects into subsets (clusters) such that the objects within a cluster are closer to each other as compared to the objects in different clusters [66]. Hierarchical clustering involves recursive clustering using previously assigned clusters [67]. At the highest level of the hierarchy, all objects are within one cluster. At the lowest level, each object is its own cluster and the number of clusters is equal to the number of objects. Between the highest and lowest levels, various intermediate levels of clusters are generated based on the similarity (or closeness) or distance between different objects. Various measures such as Euclidean distance, Manhattan distance, maximum distance, Mahalanobis distance, and cosine similarity are commonly used. Hierarchical clustering is used in statistical data analysis, pattern recognition, and data mining applications. In weighted networks, the weights can be used to represent the similarity or dissimilarity between nodes. For the OSS social networks discussed in this paper, the weights represent the closeness between people and modules. Clusters of people represent participants working closely with each other. The result of hierarchical clustering is a tree with closely related nodes closer to each other and the dissimilar nodes distant from each other. The relative sizes of clusters and their overlap convey significant information about the network structure.

4. CASE STUDY - DRUPAL NETWORK

The approaches discussed in the previous section are utilized to analyze the structure and evolution of the Drupal community (www.drupal.org). Drupal is an open-source content-management system, which allows the creation of community-based websites. The Drupal framework consists of a core and a large number of modules developed by users using open-source techniques. Drupal has been chosen for this study because of its strong community and the easy access to participant and module data for analysis.
The Drupal project was started in 2000 and it currently has a large community of contributors. There are different types of users who interact with the Drupal community. Passive Users are users who only download and use the software but do not contribute to the code. Active Users contribute to the discussion board and identify bugs but do not modify the code. Co-developers modify the codes, fix bugs, and add new features to the software. Core developers contribute largely to the core of the Drupal code and coordinate co-developers’ work. Project leaders are the project administrators who manage the direction of the entire project. For the analysis presented in this paper, only co-developers, core developers, and project leaders are considered. Activities such as the identification of bugs and contributions to discussions on the bulletin boards are not considered in this paper. The bipartite network created for the analysis of Drupal community consists of two types of nodes – a) the people (participants), and b) the projects (modules) they contribute to. The analyses discussed in Section 3 are carried out for the Drupal data. The analysis of the structure of networks is discussed in Section 4.1 and the analysis of network evolution is presented in Section 4.2.

4.1. Analysis of Network Structure

4.1.1. General Discussion of the Network Data

The data was collected for Drupal 5.x in August 2009. The data consists of 1907 projects (modules) and 1217 participants who contributed the code during the nine years from the start of the project in the year 2000. The data is used to create the bipartite network consisting of people and project from which two networks, discussed in Section 3.2, are created. The two networks are referred to as people network and project network. The characteristics of the two networks are listed in Table 1.

Table 1 - Characteristics of the People and Project networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Average Degree</th>
<th>Centralization</th>
<th>Average Distance</th>
<th>Average Density</th>
<th>Clustering Coefficient</th>
<th>Connectedness</th>
</tr>
</thead>
<tbody>
<tr>
<td>People</td>
<td>24.62</td>
<td>1.26%</td>
<td>2.86</td>
<td>0.020</td>
<td>0.74</td>
<td>0.446</td>
</tr>
<tr>
<td>Project</td>
<td>42.14</td>
<td>2.71%</td>
<td>2.87</td>
<td>0.022</td>
<td>0.83</td>
<td>0.545</td>
</tr>
</tbody>
</table>

It is observed that both the networks are similar in terms of the measures listed in the table. For networks with over 1000 nodes, the average distance between the nodes of 2.86 and 2.87 are very low. The degree centrality of both networks is also low. The average degrees of the nodes in the two networks are of 24.62 and 42.14. With the low average degrees, we can assume that both networks are in a low-
scale unitary connection. The low average density in both networks implies that different people develop most of the projects, and there are more co-developers than core developers. From the table, it is also observed that both the networks have low average distances and high clustering coefficients. The combination of low average distance and high clustering coefficient denotes that this network is highly connected. In network analysis, this is called the “small-world” phenomenon. Small world phenomenon means that any two individuals in the network are likely to be connected through a short sequence of intermediate acquaintances [59]. With over a million possible links in the people network and three million in the project network, the connectedness values of 0.446 in the people network and 0.545 in project network show high extent of connectivity of the two networks. These characteristics provide basic information about the network structure. Further details are obtained by degree distribution and clustering in the following sections.

4.1.2. Degree Distribution of the Networks

The degree distributions are plotted in Figure 3 and Figure 4. Figure 3 contains the degree distributions of the nodes in the bipartite network whereas Figure 4 contains the degree distributions of the people and project networks. The points in Figure 3 represent the number of people working on different modules and the number of modules on which different people contribute to. On the other hand, the points in Figure 4 represent the number of projects linked to other projects through common contributors (left) and the number of people linked to other people (right). The X and Y axes in Figures 3 and 4 are the degree cardinality and the number of nodes in different degree cardinality respectively. It is observed that the degree distributions of the networks are linear on a log-log scale indicating a scale-free topology of all three networks. Such a scale-free topology has been observed in many biological, technical, and social networks. The community of Sourceforge [68] has also been shown to have a similar degree distribution [29].

In a scale-free network topology, there is a small set of nodes with a large number of links with other nodes, and a large number of nodes with small number of connections. The nodes with a large number of connections are called the “hubs”. The hubs in the project network are the key projects that provide the core functions of Drupal. The hubs in the people network are the small number of core developers who
communicate with and support a large number of other participants. In the bipartite network, the “hubs” can be either core developers or key projects.

A widely accepted model for generating a scale-free network is the preferential attachment model. According to this model, networks grow through the addition of nodes. New nodes preferentially attach to other nodes with high degree. Hence, the probability of attachment of a new node to existing nodes is proportional to the degrees of the existing nodes. We believe that the model explains the emergence of
scale-free networks in the open-source domain because the modules that have higher number of participants develop faster, thereby increasing the modules' utility, and hence attract even more participants.

4.1.3. Network Structure Analysis through Clustering

In this section, we analyze the structures of networks using clustering techniques. 3D plots are used to visualize the adjacency matrices corresponding to the weighted people and project networks. The plots are shown in Figure 5(a) and Figure 5(b) respectively. The z-axis corresponds to the weights of links connecting nodes on x and y axes. The weights on the links are used as the similarity measure for clustering purposes. This is because the larger weights on people networks indicate that people are working together on greater number of projects. Similarly, larger weights on the project networks indicate that the projects share greater number of participants. Hence, the larger the weights, closer the nodes are.

The adjacency matrices are clustered and the corresponding plots are shown in Figure 5. From these plots of the clustered networks, it is clear that there are few participants near the bottom-right corner who have connections with a lot of other people. These participants are the core developers who contribute to the entire project and oversee the work of other participants. These participants are also highly connected with each other. The other participants are weakly connected and have few links with each other. As discussed in the previous section, this kind of distribution is due to the scale-free nature of the network. Hence, the plots provide an indication of the different roles of the individuals. Similar characteristics are also observed in the project network.

In addition to the highly skewed distribution of links between nodes, we also observe clusters of nodes appearing as blocks in the diagonal. The blocks indicate sets of participants or projects that are highly connected. The highly connected participants are similar to teams of individuals working together in traditional product realization. The difference, however, is that these clusters (teams) in open-source are self-organized as opposed to being pre-defined as in traditional product realization.
4.2. Analysis of Network Evolution

In Section 4.1, we discussed network structure of Drupal using the data from August 2009. Drupal was founded in 2000, rapidly growing from 12 developers and 23 projects to 1217 developers and 1907 projects. The network evolution analysis is aimed at understanding how the bipartite, people, and project networks grow over time. From the analysis of network evolution, our objective is to identify characteristics of the network that change over time and the characteristics that are invariant with time. The knowledge about the evolution of these networks can be used to direct community growth.

4.2.1. Generation of Snapshots of Networks at Different Intervals

During nine years of development, Drupal community has grown by a factor of about 100, both in terms of the number of participants and modules. In order to analyze the evolution of the community, six snapshots of the data are generated based on the time when people joined the community and time when projects were created. The snapshots are generated at intervals of one year, starting with year 4.
Different snapshots of data are not generated for the first three years because most of the evolution in the network took place between years 4 and 9 of the project. The number of participants (people) and modules (projects) at different snapshots are shown in Figure 6. It is observed that the growth is exponential. The exponential curve-fits for both are shown in the figure, which provide an important indicator of evolution of the Drupal community.

![Growth of the number of modules and participants](image)

**Figure 6 - Number of people and modules at different times**

4.2.2. *Evolution of the Network Characteristics*

The social network measures are used to analyze the characteristics of the six snapshots of the people and project networks. The results are shown in Figure 7. The trends in the evolution of the networks are analyzed using average degree, degree centrality, average density, clustering coefficient and connectedness.

In Figure 7(a), the average degree of the nodes in the networks is plotted at different time steps. The plots show three stages in the networks’ evolution - the first stage where the average degree increases, the second stage during which the average degree reduces, and finally the third stage where the mean degree remains relatively constant. The three stages are observed in both product network and the project network. We believe that this trend can be explained as follows. During the first phase, only the core developers contribute to the project. All the developers closely collaborate with each other. Any addition to the number of core developers results in the increase in the degree of all the nodes in the network. Hence, the average degree increases during this first phase. This phenomenon has also been discussed as an initial euphoria among the co-developers and core-developers [45]. During the second phase, as more developers join the network, the number of participants and modules (i.e., nodes in the network) increase at a rate faster than the number of links. Hence, the mean degree reduces. During the
third phase, the existing participants explore other modules and initiate collaborations with other participants.

Figure 7 - Average degree, degree centrality, average density, clustering coefficients, and connectedness of Project and People networks with respect to time

Figure 7(b) shows the degree centrality of the network as a function of time. The degree centrality of both the networks shows a steep decline during the initial phase, indicating the significant difference between the nodes with highest degree and the other nodes. During the initial years of an open-source project, core-developers play an important role in the development of the project. At that time, some project leaders start many projects while others start only a few. This causes the significant difference seen in the figure. During the later phases, there are more choices of projects for people to contribute to, which results in a gradually decreasing curve.

The average density of the networks is plotted in Figure 7(c). The trend of the average density is similar to the degree centrality. It reduces steeply during the initial phase and then reduces gradually during the later phases. During the initial phase, there is a closely connected group of core-developers
and project leaders, which means that the nodes are highly connected in both project and people networks. In order to maintain the same density of a network, the number of links must increase at a rate equal to the square of the number of nodes. However, we observe that rate of increase of the number of links is almost equal to the rate of increase in nodes. Hence, the density reduces as shown in Figure 7(c).

The clustering coefficients of the people and project networks are plotted in Figure 7(d). After reaching their peaks in a short time, both the curves undergo a steady decline. We believe that the reason for the downward trend is the exponential increase in participants in the Drupal network. Further, the core modules of Drupal were developed during the first phase, resulting in the high clustering of the people network. The closely related core modules result in higher initial clustering coefficients. Despite its decline, the clustering coefficients in both networks are still high compared to random networks. The low average node distance in both networks combined with the high clustering coefficient is a characteristic of scale-free and small-world networks.

4.2.3. Evolution of Scale-free Property

The degree distributions of the participants in the bipartite network for three snapshots of data are shown in Figure 8. It is observed that during each data snapshot, the form of the degree distribution remains the same – power law. The only difference is in terms of the parameters of the power law. The coefficient increases while the exponent decreases with time as shown in the figure. Faloutsos [69] has shown that the frequency, $f$ of an outdegree, $d$, is proportional to the out-degree to the power of a constant $k$.

From Figure 8, we obtain the following relationship between the frequency and degree:

$$f = \beta \times d^\alpha$$  \hspace{1cm} (4)

where $d$ is the degree and $f$ is the frequency of a degree, $\alpha$ and $\beta$ are constants. It is observed that there is an approximate exponential relationship between $\alpha$ and $\beta$ in bipartite network. This helps in eliminating one constant $\alpha$ or $\beta$ in the equation of frequency and degree. Hence, the evolution of the Drupal network is far from arbitrary and follows some common trends observed in complex networks [69]. Currently, we have obtained the equation of frequency and degree with only one constant. In future work, we will determine whether there is a relationship between the constant and other parameters of the network, such as the number of nodes or links of the network.
4.2.4. Analysis of Network Structure Evolution

In this section, we present the adjacency matrix for the snapshots of network at different times. In both people and project networks the row and column sequences, which stand for participants and projects are ordered for years 6-9. This method is effective for observing the evolution of the network structure. Figure 9 shows the adjacency matrices of the people network at different times as 3D plots. The participant IDs are organized in an increasing order based on the time when they joined Drupal project. Hence, the developer who joined Drupal four years ago has an ID greater than the participant who joined two years ago. Similarly, the projects’ IDs are also arranged in increasing order.

The heights of the bars in Figure 9 indicate the values in the adjacency matrix. In Figure 9(a), we observe that almost every participant has strong collaborations with others. In the second plot, more participants are added and the strengths of collaboration between newly added participants are weaker. The strengths of collaboration between existing participants continue to become stronger, as indicated by the change in heights of bars corresponding to existing participants. Finally, the plot corresponding to year 9 shows the presence of small groups of participants who collaborate with almost everyone in the
network, which signifies that these are participants who joined later but became core developers. Similar trends are observed in the projects network displayed in Figure 10. The block close to the origin refers to the core modules. The height corresponding to the core modules increases as more and more modules are added.

Figure 9 - Evolution of People network

Figure 10 - Evolution of Project network
5. A SIMULATION MODEL OF THE DRUPAL NETWORK EVOLUTION

The Drupal network presented in Section 4 shows power-law distributions indicating a scale-free topology. Extensive literature on network evolution suggests preferential attachment as the likely mechanism for the generation of scale-free networks [31, 70]. In a growing network, preferential attachment means that the probability of attachment of a node to an existing node is proportional to the degree of the existing node [31]. Variations to the preferential attachment mechanism have been proposed to account for specific deviations from a truly scale-free network [52, 71, 72]. The attachment rules are defined based on the attachment kernel, defined as the probability that a new node links to an existing node with degree k. For example, Dorogovtsev and co-authors [73] present a linear attachment kernel where the probability of a new node linking to an existing node, $i$, is given by $P_i \propto (K_i + C)$ where $C$ is a constant and $K_i$ is the degree of node $i$. Krapivsky and Redner [74] present a comprehensive analysis of the models for growing networks.

Our hypothesis is that preferential attachment mechanism underlies the growth of the Drupal network also. In order to validate the hypothesis and to predict the underlying dynamics of the network growth, we present a computational model whose input is the local behavior of the nodes (people) and the output is the structure of the networks at various time-steps. The local behaviors in the network include activities such as new individuals joining the network, creation of new projects, and participation of existing individuals on existing projects. Existing literature on preferential attachment is focused on 1-mode networks. In this paper, we extend the model to a bipartite network. Based on specified local behaviors of individuals in the network, the global structures of the entire network and process of network growth can be modeled. Using the model, the effect of different local behaviors on the global network structures can be studied.

The model is based on the following assumptions: i) only one new participant joins the network in each time-step, ii) the probability that an existing participant creates a new project is based on a constant project/people ratio, iii) the number of links added between existing participants and existing people is constant in each time-step. The inputs of the model are the numbers of people ($M$), projects ($N$), and the links between people and projects ($L$). The model is initialized by creating a small set of people and projects and linking them as a random initial network. The initial network is grown through the gradual
addition of nodes (people and projects) and links (the participation of individuals on projects) in discrete time-steps. During each time-step, the network grows in three ways:

a) **Participation of new individuals**: One new individual is added to the model during each time-step. The new individual either initiates a new project with probability \( p_a \), or contributes to an existing project with probability \( 1-p_a \). If the new participant creates a new project, a link between the new participant and the new project are created. Otherwise, the new participant preferentially links to an existing project based on projects’ degrees (i.e., the number of people working on the project). We utilize the model proposed by Dorogovtsev and co-authors [73] where the probability of linking to a project is equal to the degree plus a constant \( c_0 \), normalized over all the projects. Hence, the probability of a new individual \( i \) linking to a project \( j \) (with degree \( k_j \)) is:

\[
P_j = \frac{k_j + c_0}{\sum_{r=1}^{n_t}(k_r + c_0)}
\]

where \( n_t \) is the number of projects at time-step \( t \).

b) **Creation of new projects by existing participants**: The probability that a new project is created by existing participants \( (p_b) \) is determined by the ratio of total number of people and projects \( (m/n) \). This is a key parameter used to ensure that the targeted number of people and projects are achieved at the end of the process. It is assumed that the choice of the individual who creates a new project is again based on preferential attachment. Hence, the number of projects created by an individual is proportional to the number of projects that he/she is currently working on (i.e., the individual’s degree) plus a constant \( c_1 \). The probability that a new project \( j \) is created by an individual \( i \) (with degree \( k_i \)) is thus:

\[
P_i = \frac{k_i + c_1}{\sum_{r=1}^{m_t}(k_r + c_1)}
\]

where \( m_t \) is the total number of participants at time-step \( t \).

c) **Participation of existing participants on existing projects**: Existing participants may also choose to work on projects created by other participants. In the bipartite network, this can be viewed as the addition of links between the existing participant nodes and project nodes. The number of links to be added is determined based on the total number of links \( (L) \) to be obtained at the end of the process. As a simplifying assumption, equal numbers of links are added throughout the process. Again, the
links are added between projects and participants using the preferential attachment mechanism. An existing project is selected using the probability given in expression (5) and an existing participant is selected based on the probability given in expression (6). The process is repeated for the number of links to be added during a time-step.

The inputs to the model are \( N = 1907, M = 1217, L = 4614 \). The output of the model is bipartite network of people and projects generated over predefined time-steps. The objective is to simulate the global network structure simply based on the local behaviors. The preliminary focus is on evaluating the degree distribution of the global network structure. It is observed that the overall degree distribution follows a scale-free degree distribution. The parameters \( p_a, c_0, \) and \( c_1 \) are calibrated to bring the degree distribution of the bipartite network close to those in the Drupal network. The values of the parameters that resulted in the degree distribution that is closest to the real network are: \( p_a = 0.15, c_0 = 0.55 \) and \( c_1 = 0.75 \).

![Figure 11 – Degree distribution of the nodes obtained from the model, compared with the degree distribution from Drupal](image)

Figure 11 shows the degree distribution of the bipartite network generated from the model along with the degree distributions from the original Drupal data (for year 9). It is observed that there is agreement between the degree distributions generated from the model and those from the original data. The slope and intercept parameters of the two linear-fits are within 4% of the original data. This agreement is a positive indicator of the validity of the hypothesis that local behaviors based on preferential attachment underlie the growth of network. A comparison of the joint degree distribution for the bipartite graph is shown in Figure 12. The X and Y axes are the degrees of people and projects respectively. The Z-axis is
the probability that a person and a project with corresponding degrees are linked to each other. The figure shows that the model closely replicates the joint degree distribution also.

![Figure 12 – Joint Degree Distributions (JDD) of Drupal and simulated networks](image)

The degree distributions of people in the Drupal network and the simulated network for years 7 and 8 are shown in Figure 13. It is observed that the model replicates these degree distributions for the intermediate time-steps relatively well, indicating that the model captures the dynamic behavior of the network growth also. Note that the deviation reduces with the increase in time. In addition to the degree distributions, there are various other indicators of networks, some of which are discussed in Section 3.3.1. These indicators are compared to the people network in Table 2. It is observed that the simulated networks are close to the original networks in terms of indicators such as the connectedness and clustering coefficients. On the other hand, some of the indicators such as the average degree show noticeable differences. The difference between the Drupal and the simulated networks become smaller as the number of years increase. This can be attributed to the fact that the data from the last year are used as the inputs to the model. The difference in the mean degree in early years is due to the non-linear increase in the number of links in the real network (it is assumed linear in the model).

From these comparisons, it is clear that the extension of preferential attachment to bipartite networks can be used to model the growth of these networks, and that preferential attachment may be a significant underlying mechanism in Drupal. This is a step towards validating the hypothesis. While the greatly simplified model presented in this section closely reproduces the degree distribution of the resulting networks, the assumptions such as the linear increase in number of people and projects over time result in the deviations observed in Table 2. Hence, there may be additional mechanisms that play a role in the...
growth of open-source communities. This highlights the need for further investigations in the underlying mechanisms. Future extensions of the model will involve simulating the growth of community based on different attachment kernels, relaxing the assumptions, and utilizing the information about increase in numbers of participants, projects, and links to model the network growth.

![Figure 13 – Degree distribution of People in Drupal and simulated networks](image)

| Table 2 - Comparison of Drupal people-network with the simulated people-network |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Years | Years | Mean Degree | Degree Centrality | Average Density | Clustering Coeff. | Connectedness |
| 4 | Drupal | 24.0 | 14.05% | 0.480 | 0.842 | 0.742 |
| | Simulated | 10.8 | 17.18% | 0.138 | 0.990 | 0.386 |
| 5 | Drupal | 32.7 | 5.61% | 0.303 | 0.855 | 0.696 |
| | Simulated | 12.0 | 14.54% | 0.105 | 0.934 | 0.526 |
| 6 | Drupal | 32.8 | 3.41% | 0.115 | 0.816 | 0.568 |
| | Simulated | 14.6 | 13.47% | 0.062 | 0.783 | 0.637 |
| 7 | Drupal | 30.39 | 2.27% | 0.055 | 0.787 | 0.508 |
| | Simulated | 17.9 | 7.44% | 0.034 | 0.752 | 0.616 |
| 8 | Drupal | 24.85 | 1.57% | 0.028 | 0.754 | 0.394 |
| | Simulated | 19.9 | 4.02% | 0.017 | 0.714 | 0.615 |
| 9 | Drupal | 24.62 | 1.26% | 0.020 | 0.740 | 0.446 |
| | Simulated | 21.9 | 2.54% | 0.009 | 0.679 | 0.640 |

6. CLOSING THOUGHTS

The structure of an organization not only affects the efficiency through the flow of information between individuals, but also has an impact on the final product. Hence, it is important from a design engineering standpoint to gain an understanding of the structure and evolution of open-source communities, and, ideally, to design mechanisms to direct their evolution.

The first contribution of this paper is an approach for analyzing the structure and evolution of open-source communities and its application to a specific OSS community (Section 4). The objective of this analysis is to extract patterns in community structures and trends in their evolution. An understanding of
such patterns and trends, and their impact on the product evolution can potentially be used for guiding open-source projects. Despite the differences between software and hardware development, the underlying mechanisms of community formation and evolution in open-source software and hardware are similar. The analysis is carried out using an existing OSS development community because of the availability of data from successful projects. It is observed from the analysis that the specific community structure also displays a scale-free topology, which has been observed in a large number of other social, technical and biological networks. It is emphasized that the conclusions about the community are valid for the Drupal community. Whether the patterns observed in this community represent all open-source communities is still an open question that requires further investigation.

The second contribution is a predictive model that simulates the evolution of the community (Section 5). The model is solely based on the predicted local behaviors of the individuals to generate global community structures. The local behaviors are modeled based on the hypothesis that the observed scale-free network topology is a result of preferential attachment of individuals to high-degree projects. A variation of the basic preferential attachment model for bipartite networks serves as a good starting mechanism for modeling the evolutionary dynamics of the network. The generated networks closely approximate the actual Drupal networks in terms of basic network characteristics such as degree distributions, connectedness, and clustering coefficient. The model is applicable, and can be calibrated, for other networks with similar degree distributions.

We acknowledge that open-source processes are complex in nature. Each project has its own peculiar characteristics. Hence, just like any other model, we envision various levels of abstraction of the model – from very general to very specific. The general models are good for a broad range of projects but fail to capture the specific details of a project. The appropriate granularity of a model depends on the goals of the modeling activity. The specific models contain specific knowledge that can be used for detailed modeling of a specific project. In this paper, the focus is not on modeling a particular project in complete detail but to extract general underlying mechanisms that drive the bottom-up evolution of these communities. It is these general underlying mechanisms that can be used to guide a wide range of such projects. These mechanisms can range from providing incentives to the participants to change their behavior to supplementing the self-organized teams with traditional structured teams.
There are various avenues for further extension of this work. The first important aspect is the analysis of other projects to determine similarities and differences between different open-source projects. As more data about open-source hardware becomes available, the similarities and differences between open-source software and hardware can be explored. The model can be further improved by refining the local behaviors of individuals. The information about the amount of contributions from each participant is available in many projects. This information can be used to further refine the model. The use of other network growth models such as rate equation method, probabilistic, and generation function techniques [74] can be investigated. As discussed in Section 2, the co-evolution of community structure and product structure is an important avenue for further research.

As a final comment, Social network analysis (SNA) is a fast growing field with a focus on analyzing structures of communities and their impact on processes on these networks such as transmission of information. Design research community can benefit from the tools developed in SNA to understand and account for the organizational (community) structures within product development, particularly in emerging bottom-up self-organized approaches such as open-source.

7. ACKNOWLEDGEMENT

The authors gratefully acknowledge the financial support from the National Science Foundation through the CAREER grant # 0954447.

8. REFERENCES

5. Rally Fighter, Product Brochure, [cited February 05, 2010]; Web Link: http://www.local-motors.com/assets/rally-fighter-brochure.pdf,

10. Anderson, C., February 2010, In the Next Industrial Revolution, Atoms are the New Bits, Wired Magazine.


List of Figures
Figure 1 - Illustration of the data-gathering step
Figure 2 - Example of a bipartite network and the two derived networks
Figure 3 - Degree distribution of two types of nodes in the bipartite network
Figure 4 - Degree distribution of projects and people networks
Figure 5 - Adjacency matrix plots of a) people network and b) projects network before and after clustering
Figure 6 - Number of people and modules at different times
Figure 7 - Average degree, degree centrality, average density, clustering coefficients, and connectedness of Project and People networks with respect to time
Figure 8 - Evolution of the power-law coefficients for degrees of people in the bipartite network
Figure 9 - Evolution of People network
Figure 10 - Evolution of Project network
Figure 11 – Degree distribution of the nodes obtained from the model, compared with the degree distribution from Drupal
Figure 12 – Joint Degree Distributions (JDD) of Drupal and simulated networks
Figure 13 – Degree distribution of People in Drupal and simulated networks

List of Tables
Table 1 - Characteristics of the People and Project networks
Table 2 - Comparison of Drupal people-network with the simulated people-network