SciBrowser: Exploration and Analysis of the Complexity, Structure, and Activity Dynamics of Open Source Science Communities

by

Damodar P. Shenvi Wagle

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 9, 2011

Keywords: complexity, social network analysis, visualization, software engineering

Approved by

Levent Yilmaz, Associate Professor of Computer Science and Software Engineering
David Umphress, Associate Professor of Computer Science and Software Engineering
Dean Hendrix, Associate Professor of Computer Science and Software Engineering

Abstract

Open Biomedical Ontology (OBO) is a socio-technical community that is comprised of individuals dispersed geographically, but function as a coherent unit through the use of cyber-infrastructure. This study explores dynamics of open source science in such virtual socio-technical networks. Innovation within a socio-technical network can be defined as the approach to work that leads to the generation of novel and useful ideas and processes. Among the factors that influence innovation are structural properties such as centrality, density, clustering coefficient, and average path length of socio-technical networks, as well as effectiveness in collaboration. Hence, we explore virtual scientific communities from three main perspectives: network, collaboration, and activity. Structural network metrics measure the resilience of socio-technical networks. Collaboration analysis aims to discover interaction patterns among participants and between knowledge domains. Activity analysis facilitate discerning artifact submission and community growth patterns over time. To expedite analysis, a computational ethnography tool, called SciBrowser, is introduced. Using SciBrowser, we observe power law degree distributions, which indicate presence of scale-free network configurations. Such configurations provide an explanation for the resilience of research communities in cyberspace. A new metric, called activity strength, suggests that major contributors of the project are weak collaborators. As a result, their strong contribution factor is nullified by their weak collaboration intensity. Activity patterns of the observed projects suggest the presence of an adaptive renewal cycle, which is the epitome of behavior in innovation ecosystems.

Acknowledgments

I would like to thank Dr. Levent Yilmaz, Dr. David Umphress and Dr. Dean Hendrix for their direction, assistance, and guidance. In particular, Dr. Yilmaz's recommendations and suggestions have been invaluable for the project and for software improvement.

Special thanks should be given to my student colleague and lab-mates Michael Arnold, Guangyu Zao and Ozgur Ozmen who helped me in more ways than one. My thesis would be incomplete without the visualizations and data provided by Michael.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The open source research has turned into a business model because of its popularity and the open innovation revolution it brought in the software industry 15 years ago. Despite differences in research dynamics between software and biomedical industries, this model has entered the biomedical research [29]. Virtual collaboratories are enabling a new mode of collaboration among scientists distributed over the globe to share and co-develop knowledge over the cyber-infrastructure. The practice of science is becoming open and global as the access to knowledge, as well as its production, is becoming increasingly transparent. Service-oriented science [15] and e-Science [8] initiatives create scientific communities where shared domain knowledge is no longer exclusively documented in scientific literature or patents. Rather it is documented in software, simulations, and databases that represent an evolving collective knowledge-base that is governed and maintained by community members. Just like open source software communities, "SourceForge for science" style in scientific production and collaboration provides the requisite infrastructure that encompasses community membership services, catalogs, storage services, and workflow orchestration services.

As reported in [20], [13] and [38], open source communities promise a great deal of discovery and learning. Many researchers have examined open source science communities in the past. For example, Madey et al. [18] performed topological analysis of an entire development community on SourceForge.net [14], wherein they classified the members of the community based on their activity into 4-5 groups and then performed social network analysis on the networks. They study primarily the project-developer network. Based on this network, they derive project-project and developer-developer networks. In our study, we investigate the individual projects and study the collaboration network of the members

at the artifact level which makes our study more detailed. Moreover, we do not classify the members based on their roles in the database. The roles as mentioned in the database can be deceiving for the purpose of our study because they does not correlate with their respective member activities. Recent studies such as the human flesh search engine [13] and the network analysis of scientific workflows [38] also use social network analysis to explore their communities of practice and answer some of the vital questions about their research. For instance, in [38] authors aim to answer the questions "What is the current usage pattern of services in scientific workflows, and how can this knowledge be extracted to facilitate reuse?"

What differentiates our study is the 3 pronged approach towards analysis of the OBO community. Our aim is to study the network not only from its structural point of view (which is social network analysis) but also from the perspectives of collaboration and activity. From the structural perspective, we visualize different types of graphs: user-user, user-artifact-user, user-domain, and domain-domain. We study social network metrics: different centralities, density, clustering coefficient and average path length for these graphs. Centrality and density metrics assist in detecting whether the core periphery structure exists in the network. Clustering coefficient and average path length comment on the small world nature of the community. Both of these properties, if present in the network, can foster the innovation in the community. In structural analysis we also look for presence of power law in the degree distribution of the users, artifacts, and domains for different types of graphs mentioned above. Apart from degree distribution, we concentrate on the phenomenon of preferential attachment. To examine this phenomenon, we plot rate of change of degrees of actors over time.

Collaboration and activity analysis gives a new dimension to the study. Collaborative approach takes place at the user level, for which we developed a novel set of metrics (to be discussed in chapter 6) to identify influentiality and innovation potential of users based on their activity in the project. We also visualize the collaboration between users using

color coded maps and highlight the areas where collaboration is significantly high. Temporal analysis of activity is conducted at both the group and the domain level within a group, thereby relating the open source science project life cycle to the organizational life cycle. Two types of activities are identified for purpose of our study: artifact contribution and number of active users. We examine the implication of one activity over the other. Activity outbursts (high activity points) are also plotted to see how frequently activity crosses a threshold. Threshold is determined by the user by setting relevant parameters.

We introduce a tool, called SciBrowser, for comprehensive analysis of open source science communities. Among significant findings is a power law degree distribution for the User-User graph indicating the resilient nature of the community. We also observe power law distribution for the User-Artifact-User graph which indicates that only a few artifacts greatly influence responses from users. Most of the projects have their clustering coefficient value stabilizing at around 0.5 and their average path length value stabilizing at around 2 which is the indication of the small world network. Distribution of innovation metric "Activity Strength" on a log scale indicates that major contributors of the project are weak collaborators. Thus, their strong contribution factor is nullified by their weak collaboration intensity. As a result, we fail to get a power law distribution when we consider both contribution as well as collaboration as a part of "Activity Strength" metric. Activity patterns of the projects closely resemble the virtual organization life cycle. Thus, there is a possibility that the open source science projects possess specific organizational characteristics like division of labor, leadership, level of commitment, and coordination/control.

This thesis is structured as follows. Chapter 2 highlights the properties of complex adaptive systems and open source communities that prevail in OBO. It also explains relevant social network metrics that are related to innovation and creativity. Chapter 3 presents the conceptual framework of the SciBrowser tool in terms of major building blocks that constitute the tool. Chapter 4 introduces the SourceForge database schema and its extension

to facilitate calculation of metrics required for the project. Chapter 5 outlines the implementation of the SciBrowser tool from a software engineering perspective and discusses the software process in terms of different stages of the software development life cycle. Chapter 6 elaborates on data analysis using SciBrowser and describes different ways in which we analyze the community. We relate our observations to the innovation capacity of the group as well as the individual. In Chapter 7, we conclude by summarizing our findings and point out potential avenues of future research.

Chapter 2

Background

This chapter presents the theory behind open source communities and complex adaptive systems. It explains the characteristics of Open Biomedical Ontology (OBO) which makes it a complex system. Some light is thrown on the important aspects of visualizing science. At the end, various metrics used in the field of social network analysis are formulated and their relevance in our study is explained.

2.1 Open Source Communities

The key to understanding an organization is to understand its governance, because it not only offers an insight into an organization's conception of control, but also indicates how such communities can be sustained over time [30]. Corporate organizations use bureaucratic bases of authority, while other organizations such as socio-technical communities of practice use shared bases of authority. It is important to know how collaboration between individuals in social communities accomplishes important outcomes such as knowledge sharing - a determining factor in innovation.

One of the key aspects of open source communities is learning through knowledge sharing. Members voluntarily collaborate and contribute towards community formation and growth in the form of artifacts for either public or private benefit. Typically, members are geographically dispersed and rely on modern communication technologies such as the Internet as the means of communication and coordination. According to [30], a meritocratic governance system must be introduced in the open source communities in order to attract high quality contributions from the members. In return, the contributors can be rewarded with greater status, responsibility, or opportunity. Thus, communities tend to satisfy the

contributor's need for recognition. There are 4 stages of governance that an open source community goes through [30]. These stages are explained in relevance to the study of Debian Linux Community:

- De facto Governance
- Designing Governance
- Implementing Governance
- Stabilizing Governance

Leadership of such communities can either be decided based on Technical Contribution [30], or Organizational Building and Leadership [30]. According to the technical contribution approach, the greater the amount of technical contribution of a member, the higher is the probability that the member will become a leader. Based on the organizational building and leadership approach, the more a community member participates in online discussions, the higher is the probability that the member will become leader.

2.2 Complex Adaptive System

Complex systems research is an interdisciplinary field that seeks to explain how large number of relatively simple entities organize themselves, without the benefit of any central controller into collective whole that creates patterns, uses information and in some cases evolves and learns (called as Complex Adaptive System) [27]. For instance, ant colonies [27] are an example of complex adaptive systems. An ant colony consists of hundreds to millions of ants, with each ant being a simple creature performing simple tasks like foraging for food, responding to the chemical signals of other ants and fighting intruders. But as a group, they create complex structures like bridges (out of their own bodies) from one nest site to another via tree branches separated by great distances.

It can be seen from the examples in [27] that complex systems consist of many elements connected together. A possibility exists that the parts of complex systems are complex

systems themselves. But the individual element need not necessarily have a complex nature. They can be simple parts which adhere to simple rule sets. If a system consists of simple parts whose collective behavior is complex, then the resulting phenomenon is called as emergence [4]. Emergent patterns are not caused by single elements/agents working in isolation, but they emerge from the interactions that take place between the agents based on the simple rules which an agent operates on. In order to understand complex systems it is important to know their properties. Each complex system is sufficiently different from others, but at an abstract level they have some commonalities. Following are some of the common characteristics and mechanisms of complex systems:

- **Aggregation [16]** - Aggregation has two interpretations; first one is the way we model a system and second one is the behavior of the complex system. According to the first interpretation, we categorize the system into similar objects and each category becomes a class that is treated equivalently. The second interpretation is concerned about the emergent behavior as a result of aggregate interactions of agents. For example, an ant in an ant colony is a relatively simple agent, but the ant aggregate is highly adaptive and complex. Aggregates so formed can act as agents at higher levels called meta-agents. Meta-agents can also aggregate to yield meta-meta-agents which leads to an hierarchical structure commonly found in complex adaptive agents. Thus the second interpretation of Aggregation is a typical characteristic of complex adaptive systems.

- **Tagging [16]** - Tagging is the mechanism by which aggregates are formed in a system. Similar agents are identified by tags which enable the members to filter their interactions so that they can choose from the pool of agents, the agents that they need to interact with. Thus, tagging leads to aggregation of agents into meta-agents and organizations which is so common in complex adaptive systems.

- **Self Organization and Dynamicity** - It has been argued that complex activities are inevitably self-organizing [28]; that is, they cannot be fully externally or hierarchically controlled. A system can be considered a self-organizing complex system if its components dynamically interact to achieve a global goal or function [40] page 40. In the decentralized self-organizing systems there is no central authority who imposes the function. Rather the function is imposed through the autonomous interactions to produce the feedback that regulates the system. Internal structures of complex systems do not remain the same; it changes dynamically depending on the interactions that take place between the actors of the systems.

- **Non-Linearity [16]** - Generally, linearity means that we can get the whole by adding sum of the parts. Linear function consists of weighted sum of its parts as given in this function: 3x + 5y + z. But in complex adaptive systems, the whole is more that just the sum of its parts. Such systems demonstrate the non-linear properties like power law.

- **Flows [16]** - Flows can be thought of as the information transfer over a network of nodes and connectors. Agents form the nodes and their interaction forms the connectors of the network. There are two effects caused by the flows in the network: multiplier effect and recycling effect. Multiplier effect is caused when the resource is added at any node in the network. The resource is passed from node to node throughout the network. Recycling effect is caused when the resource is reused across the network. Recycling leads to increase in the resource amount as well as its quality.

- **Unpredictability and uncertainty [40] page 40** - Presence of this characteristic in the environment is another common feature of complex systems. As the systems are self organizing they use two mechanisms to cope with the uncertainty and volatility in the environment: adaptation and anticipation. In adaptation system uses learning techniques such as genetic algorithms or evolutionary computing to adapt or update

its behavior to changes in the environment. In anticipation system uses current state as well as current image of its future states to determine what the next state of the system is and accordingly updates the behavior.

- **Diversity/Disparity [16]** - Diversity of the complex system is reflected by the heterogeneity of the agents that are part of the system. Each agent may be following same set of simple rules but there are some properties that are different for each agent. For e.g. in standing ovation problem [26] each agent can have different personal traits and can behave according to those. Presence of such diverse agents causes the system to undergo cascade of adaptations when a certain type of agent is removed from the system.

2.3 Open Biomedical Ontology (OBO) as a Complex System

OBO Foundry [6] is a collaborative experiment in order to establish a common set of principles for ontology development and a standardized data acquisition system. Aim of OBO foundry is to support the community members who are developing and publishing ontologies in the biomedical field. The goal is to apply the scientific methods to the ontology development, so that, the data gathered through the biomedical research can be single, consistent, cumulatively expanding and algorithmically tractable whole. OBO Foundry is open and its contributors are the researchers who work together on a continuously evolving set of design principles that can foster interoperability of ontologies. There are more than 60 ontologies that are interested in the goal of OBO Foundry. OBO Foundry is a consortium that is comprised of multiple groups that focus on different domains. The groups used in this study are: Open Biomedical Investigations (OBI), Gene Ontology (GO), Open Biomedical Ontology (OBO), Chemical Entities of Biological Interest, Disease Ontology, Sequence Ontology and System Biology Ontology. Following are the 2 types of data that are taken into account for the purpose of this study.

- Trackers [25] facilitate submission of the artifacts that characterize open problems and feature requests. Each artifact tends to generate solution to the open problem in the form of comments or suggestions posted by other members of the community. Artifacts not only facilitate social interactions, but also act as the contributors to the knowledge base. Mostly, artifacts are submitted by the active members of the community that are engaged in knowledge creation.

- Patches [25] are the revisions submitted to the knowledge base by the core members of the community. Knowledge base evolves over time and sometimes branches out in new directions as it is explored. Exploratory branch may mature and merge with the main stable branch, or it may even terminate to become the discontinued development branch.

Just like world wide web [27], OBO can be thought of as a self organizing socio-technical system with little or no central control. OBO comprises of the individual users spread across geographically and performing simple tasks like submitting the artifacts, elaborating/commenting on the artifacts submitted by the other members. The only means of direct interaction between the members is through emails. The elaborations provide an indirect means of interaction. Through these simple actions OBO emerges as the complex system in terms of its dynamic structure, growth over time, patterns of artifact submissions, user collaborations and information flows across domains. Following are the properties of complex adaptive system that appear to be applicable to open source science communities:

- **Aggregation** - Aggregation in an open source science community symbolizes the collaboration between the agents. Members of the community are the agents who collaborate over the artifact contributions that they make. A single agent behavior is simple in terms of the contribution he/she makes, but the emergent behavior becomes evident when the users collaborate. Emergent behavior like power law, scale free network can be seen in such communities.

- **Tagging** - Members in open science communities are tagged or designated based on the technical contributions that they make [30]. Due to tagging the community gets segmented into different groups of users, and apparently members of the lower rank try to associate themselves with the members of the higher rank leading to a phenomenon called Preferential Attachment [1] which can be the cause of the power law and the scale free network.

- **Self Organization and Dynamicity** - In open science communities, central control does not exist; participation of members of the community is entirely voluntary. An impact of self organization is that the structure of the network changes dynamically with the addition of new members and resources over time. Users who greatly contribute to the community become central to the community, while those having scarce contribution remain on the periphery.

- **Non-Linearity** - Power law is an example of non-linear behavior.

- **Flows** - Knowledge mobility signifies the information transfer in the network. When an artifact is submitted by the member of the community, the knowledge gets passed on from node to node throughout the network (multiplier effect) due to collaborating users. At the same time, as the members start contributing to the artifacts in the form of comments the artifact is refined (recycling effect).

- **Diversity/Disparity** - Diversity is not necessarily a trait of the members of the community; it can be a trait of the knowledge that is injected into the network by the members. When injected knowledge is new, it gives rise to the response from the other members of the community which further enhances the knowledge. But after a while, the knowledge saturates and there are no more responses coming from the members of the community. It is at this point that we need diversity in terms of the knowledge contribution so that the responses keep coming from the members.

Having homogeneous knowledge in the network quenches the growth of the community, whereas presence of the novel ideas gives a new direction to the growth.

2.4 Visualizing Science

Visualizing science [34], over a period of time has come up as the useful tool in decision making and analysis. Researchers in this field are producing representations that are catching the attention of the program officers and the policymakers. But at the same time it is important to have a statistical basis for the visualizations so that we can differentiate between noise and a real change. Any visualization is based on the following keys:

- **Data** - Having data readily available to the researchers is the most essential requirement in visualization. University of Notre Dame maintains the Source Forge Data Archive, which is available for research. For any open source project on the Source Forge, data is updated monthly. Data can be easily downloaded as a csv (comma separated values) file by querying the database using the query portal. Queries fired on the database are SQL queries. Having such a framework and roadmap makes data collection and management easier.

- **Model** - Visualization should be based on the statistical models built on the data. This is a difference between data analysis and visualization. Data analysis is more of a building statistical models on the data that would induce sense into the data, where as visualization is the way data analysis is represented so that it becomes easy for the end user to understand the data analysis. For e.g., power law [1] becomes a statistical model/metric which is visualized using a line plot or a bar plot, because mare numbers will make is difficult for the user to interpret the power law.

- **Validation** - Underlying statistical models need to be validated, so that they convey the right information what an user intends to see. But it becomes a challenge to validate these models when we are exploring. Visualization when used as an exploration tool,

makes validation difficult, because one does not know what to expect ahead of time [34]. The following section on social network metrics talks more about the metrics that are already defined for any social network.

- **User Interaction** - For understanding what will be the right kind of visualization, we need to first understand user needs. Taking user needs into account while designing visualization tools, makes it easier to design useful visualization. The users in our case are technical people working on simulation models, so the visualizations were designed from their perspective. They needed to see the data in the form of graphs and plots so that they can tweak their simulation models based on the actual data that they see.

2.5 Social Network Metrics and Interpretation

The best way to visually represent a social network is in the form of network of nodes connected together; where the nodes are the actors of the network. In this section we will discuss the metrics explained by [36], that are useful in interpreting the social network with respect to creativity. These metrics have different interpretations for different types of graphs.

2.5.1 Centrality

Centrality indicates the prestige and influentiality associated with the actor. Creativity and centrality are related to each other according to following propositions.

> "In phase 1 a positive self reinforcing spiral exists between centrality and creativity such that an increase in one leads to an increase in the other, until centrality becomes constraining. In phase 2 the spiral becomes self-correcting such that an increase in centrality no longer leads to an increase in creativity." [31]

"As an individual becomes more central, his or her creativity should continue to increase at a decreasing rate, up to a point. Beyond this point, increases in centrality may constrain creativity." [31]

There are 3 types of centralities:

- **Degree Centrality**: At the individual level this metric determines the number of nodes connected to a given node. One can view this as a measure of activity, in the sense that a highly active actor will have links to most of the other actors [36]. Degree centrality for an actor is represented by the formula:

$$C_D(n_i) = \frac{d(n_i)}{g-1} \tag{2.1}$$

where $d(n_i)$ is the degree of the actor n_i and $(g-1)$ is the group size [36].

For a group, the degree centrality is found by subtracting the individual actor degree centrality from the maximum degree centrality value and summing them up; sum is then divided by maximum attainable value for the numerator. Generalized formula for the group degree centrality is:

$$C_D = \frac{\sum_{i=1}^{g}[C_D(n^*) - C_D(n_i)]}{(g-2)} \tag{2.2}$$

where $C_D(n^*)$ is the maximum individual degree centrality in the group and $(g-2)$ is the maximum attainable value for the numerator (star network) [39].

- **Closeness Centrality**: For an individual actor, closeness centrality determines how close that actor is from all other actors. It reflects the distance between the actor and all other actors in the network. The metric is computed as the average distance between an actor and other members of the network [31]. It not only takes direct links, but also indirect links required for an actor to communicate with all other actors. Closeness

centrality helps in knowledge mobility [9]. The actor closeness centrality is defined in [36] by the formula:

$$C_C(n_i) = \frac{(g-1)}{\sum_{i=1}^{g} d(n_i, n_j)} \tag{2.3}$$

where $d(n_i, n_j)$ is the shortest distance between node i and j. Closeness centrality for node i is the inverse of the sum of the distances of node i from all other nodes in the graph. So higher is the distance lower is the closeness and vice versa. Minimum value for the sum of distances from any node to all other nodes in a graph with g nodes is $(g-1)$. So the distance is normalized by $(g-1)$ in the equation above. Value of closeness centrality thus varies between *0* and *1*.

Closeness centrality for the group is found in the similar manner as in case of degree centrality. First subtract the individual actor closeness centrality from the maximum closeness centrality value, and add the difference; the sum is then divided by the maximum attainable value for the numerator. Following equation gives the formula:

$$C_C = \frac{\sum_{i=1}^{g} C_C(n^*) - C_C(n_i)}{[(g-2)(g-1)]/(2g-3)} \tag{2.4}$$

where $C_C(n^*)$ is the maximum individual closeness centrality in the group and $[(g-2)(g-1)]/(2g-3)$ is the maximum attainable value for the numerator (star network) [36].

- **Betweenness Centrality**: This metric highlights the actors that act as the mediators between two actors or groups of actors. They act as the communication medium between the two groups or actors, and hence have a high betweenness centrality value. Betweenness is a measure of how good an actor is at routing information. Following quote explains betweenness centrality in a better way.

> Suppose that in order for [actor] i to contact [actor] j, [actor] k must be used as an intermediate station. [Actor] k in such a network has a certain "responsibility" to [actors] i and j. If we count all the minimum paths which pass through [actor] k, then we have a measure of the "stress" which [actor] k must undergo during the activity of the network [36] (page 189)

For calculating betweenness centrality of a node, we count number of the shortest paths which pass through the node, out of the total number of paths possible in a graph. Actor betweenness centrality is defined in [36] according to the formula given below:

$$C_B(n_i) = \frac{\sum_{i=0}^{g} \frac{g_{jk}(n_i)}{g_{jk}}}{[(g-1)(g-2)]/2} \tag{2.5}$$

where $g_{jk}(n_i)$ is the number of paths between nodes j and k that pass through node n_i and g_{jk} is the total number of paths between j and k. The numerator is normalized by the denominator $[(g-1)(g-2)]/2$ which is maximum number of paths possible in a undirected graph with g nodes. Since we deal with undirected graphs above formula is appropriate for our case.

Group betweenness centrality is calculated in similar fashion as that of the degree and closeness centrality. According to [36], following equation defines group betweenness centrality:

$$C_B = \frac{\sum_{i=0}^{g} C_B(n^*) - C_B(n_i)}{[(g-1)^2(g-2)]} \tag{2.6}$$

where $C_B(n^*)$ is the maximum individual betweenness centrality in the group and $[(g-1)^2(g-2)]$ is the maximum attainable value for the numerator (star network)

2.5.2 Density

Density is the measure of degree of completeness and cohesiveness of the graph [36]. Any graph can have certain maximum number of edges. Density will measure the number of edges the graph actually consists of out of the total edges possible. For an undirected graph the maximum number of edges possible are $\frac{n(n-1)}{2}$, and if we denote the number of edges of the graph as $|E|$ then density is defined as:

$$D = \frac{|E|}{n(n-1)/2} \tag{2.7}$$

2.5.3 Clustering Coefficient

Clustering Coefficient (CC) measures the average fraction of an actor's collaborators who are also the collaborators with one another [35]. A clique is defined as a maximal complete subgraph of three or more nodes [36]. In the context of a social network, the clustering coefficient represents how cohesive the group is; a high clustering coefficient represents a tight circle, or subgroup. Similar to how circles of friends form in social settings, circles of preferred collaboration can form in socio-technical networks. When the change of this metric is shown over time it can show how well this node is integrating with the group.

2.5.4 Average Path Length

Average path length is the number of edges in the shortest path between two vertices, averaged over all the pairs of vertices [37]. It is the measure of efficiency with which information is transfered over the network from one node to the other; smaller the average path length higher being the efficiency. Small average path length gives rise to the phenomenon of the small world networks [37]. The issue of small world networks is of great importance for the network studies, as this property directly affects such crucial fields like information processing in different communication systems, disease or rumor transmission, network designing and optimization [2]. In relation to OBO the smaller average path length fosters

knowledge transfer, which can be a vital factor for innovation. The metric can be defined according to following equation. Consider an unweighed graph G with the set of vertices V. Let $d(v_1, v_2)$, where $v_1, v_2 \subset V$ denote the shortest distance between v_1 and v_2. Assume that $d(v_1, v_2) = 0$ if $v_1 = v_2$ or v_2 cannot be reached from v_1. Then, the average path length is:

$$APL = \frac{\sum_{i,j}^{n} d(v_i, v_j)}{[n * (n-1)]} \tag{2.8}$$

where n is the number of vertices in G.

2.5.5 Small World Phenomenon

According to [35] Clustering Coefficient (CC) and Average Path Length (APL) both define the small world network. In order to determine if a given network is the small world network or not, Watt's model compares CC and APL of actual network to that of the randomly generated graph of same size. Random graph has a low CC and APL. Small World Quotient (Q) is the measure of small world property of the network and is defined as:

$$Q = \frac{CC_{ratio}}{APL_{ratio}} \tag{2.9}$$

where CC_{ratio} is defined as:

$$CC_{ratio} = \frac{CC_{actual\_network}}{CC_{random\_network}} \tag{2.10}$$

and APL_{ratio} is defined as:

$$APL_{ratio} = \frac{APL_{actual\_network}}{APL_{random\_network}} \tag{2.11}$$

More closer the APL_{ratio} to 1.0 and more the CC_{ratio} exceeds 1.0, higher is the small world coefficient. In a bipartite (affiliation) network, members on the same team form a

fully linked clique. Clustering includes both the within-team clustering and the between-team clustering. If CC_{ratio} is approximately 1.0 then the clustering in the actual network is mainly the result of the within-team clustering. But as CC_{ratio} goes beyond 1.0 there is an increase in the between-team links. Also the cross-team links are mostly repeated which means that the member who has collaborated previously, likes to collaborate across the teams, with the same person they did previously. Thus, in bipartite networks the small world influences behavior through two mechanisms:

> "Structurally, the more a network becomes small worldly (formally, the more the small world quotient exceeds 1.0), the more links between clusters increase in frequency, which potentially enables the creative material within teams to be distributed throughout the global network." [35]

> "Relationally, the more a network becomes small worldly, the more links between clusters are made up of repeated ties and third-party ties, which potentially increases the level of cohesion in the global network." [35]

Thus, as the small world quotient increases, the level of connectivity between the different teams within the network increases through cohesive relations among the members of these teams. This can be considered as the reason for their successful collaboration and creativity.

Chapter 3

The Organizational Framework of the SciBrowser System

This chapter gives an overview of the application in terms of its important components, and how they communicate with each other. In addition, language specific libraries & frameworks used in the tool are also discussed. Broadly speaking, there are two parts involved in the construction of this tool: Schema Conversion Subsystem (Chapter 4) and SciBrowser Subsystem (Chapter 5).

3.1 Schema Conversion Subsystem

Figure 3.1 below shows the block diagram for Schema Conversion Subsystem. SourceForge.net [14] uses relational databases to store project management activity and statistics. There are over 100 relations (tables) in the data dumps provided to Notre Dame [33]. SourceForge.net cleanses the data about personal information and strips out all OSTG (Open Source Technology Group) specific and site functionality specific information. On a monthly basis, a complete dump of the databases (minus the data dropped for privacy and security reasons) is shared with Notre Dame. The Notre Dame researchers have built a data warehouse comprised of these monthly dumps, with each dump stored in a separate schema. Thus, each monthly dump is a snapshot of the status of all the SourceForge.net projects at that point in time. To help researchers determine what data is available, an ER-diagram and the definitions of tables and views in the data warehouse are provided. Data access is given to the academic and scholarly researchers through a query portal to extract the data. We query and extract the project specific data using the query portal and load it in the local MySQL database on the server in our lab. Our study requires us to aggregate the data, so that we have precalculated results which can further be used to calculate the

metrics we need for the purpose of our study. Thus, we have a set of python programs in the Schema Converter which convert the original schema to a new one. The new schema is called SciBrowser schema, and it contains the aggregate tables.

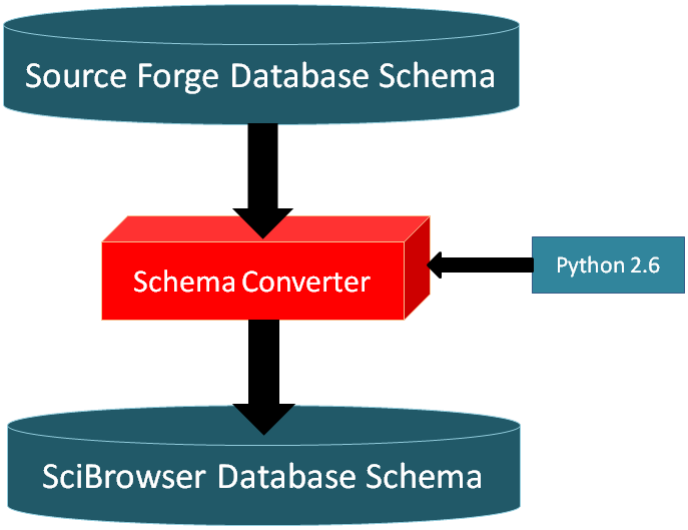


Figure 3.1: Schema Conversion Subsystem

3.2 SciBrowser Subsystem

SciBrowser subsystem is the GUI tool written entirely in python language, and is used analyze the projects on SourceForge.net [14]. The tool has three types of analysis as shown in the figure 3.2: Structural Analysis, Collaboration Analysis and Activity Analysis (explained in detail in Chapter 5). The database in the backend of the tool is the SciBrowser schema that we get after running the Schema Converter code against the SourceForge schema, as mentioned in the previous section.

As shown in figure 3.2, the tool uses couple of python libraries; given below is the brief description of each library:

- **matplotlib [19]**

  matplotlib is a library for making 2D plots of arrays in python. Although it has its origins in emulating the MATLAB [17] graphics commands, it is independent of

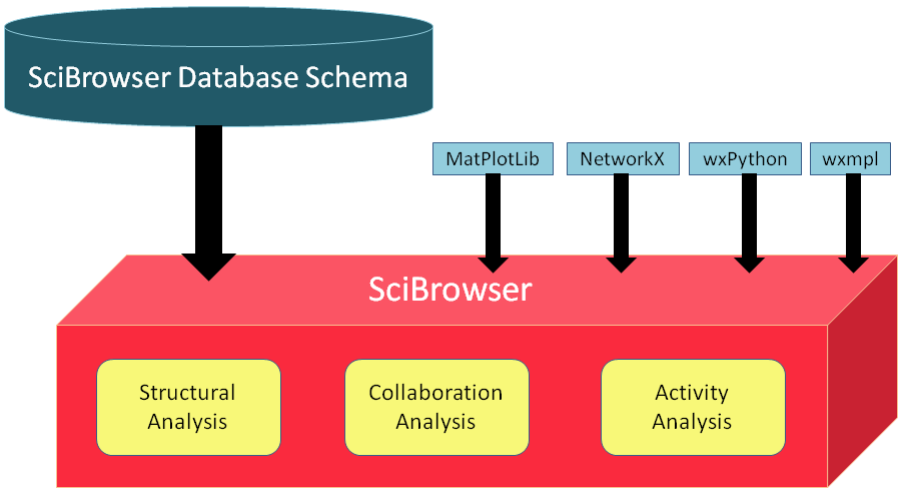


Figure 3.2: SciBrowser Subsystem

MATLAB, and can be used in a pythonic, object oriented way. Although matplotlib is written primarily in pure python, it makes heavy use of NumPy [7] and other extension code to provide good performance even for large arrays. Using matplotlib simple plots can be created with just a few commands.

- **NetworkX [3]**

  NetworkX is a python-based package for the creation, manipulation, and study of the structure, dynamics, and function of complex networks like social, biological, and infrastructure networks. It provides a standard API and/or graph implementation that is suitable for many applications such as social network analysis. The structure of the graph or network is encoded in the edges (connections, links, ties, arcs, bonds) between the nodes (vertices, sites, actors). Various types of graphs (directed, undirected) can be drawn using NetworkX; moreover weights can also be given to nodes and edges if needed.

- **wxPython [32]**

  wxPython is a GUI toolkit for the python programming language. It allows python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a python extension module (native code)

that wraps the popular wxWidgets - a cross platform GUI library written in C++. Like python and wxWidgets, wxPython is open source which means that it is free for anyone to use and the source code is available for anyone to look at and modify. Anyone can contribute fixes or enhancements to the project. wxPython is a cross-platform toolkit which makes it possible for the same program to run on multiple platforms without modification. Since the language is python, wxPython programs are simple, easy to write and easy to understand.

- **wxmpl [24]:**

  Embedding matplotlib in wxPython applications is straightforward, but the default plotting widget lacks the capabilities necessary for interactive use. WxMpl (wxPython+matplotlib) is a library of components that provide these missing features in the form of a better matplolib FigureCanvas.

Chapter 4

Schema Conversion Subsystem

This chapter gives a detailed description of the SourceForge schema that we harness to derive a new schema, called SciBrowser schema. Detailed description of the tables used in both of these schema's, is provided in the sections below.

4.1 Open Biomedical Ontologies Schema, SourceForge Research Data Archives

SourceForge.net [14] stores all its project related data in the relational databases. This data package is made available to researchers through a query portal provided by the University of Notre Dame [33]. The schema of our interest is the Artifact Data schema in the Source Forge database.

4.1.1 Original Schema

Figure 4.1 shows the original database schema provided by SourceForge for Artifact Data. Following is the description of the tables that interest our project.

- **groups**

  There are various communities that form the part of source forge open biomedical ontology (OBO), such as Gene Ontology (GO). These communities are designated as groups in the "groups" table of the SourceForge database schema. Each group has an identification number that uniquely identifies the group.

- **artifact_group_list**

  A group can be generic and might have various specific areas within it. These areas of specialization are called as domains. This table keeps the association between a

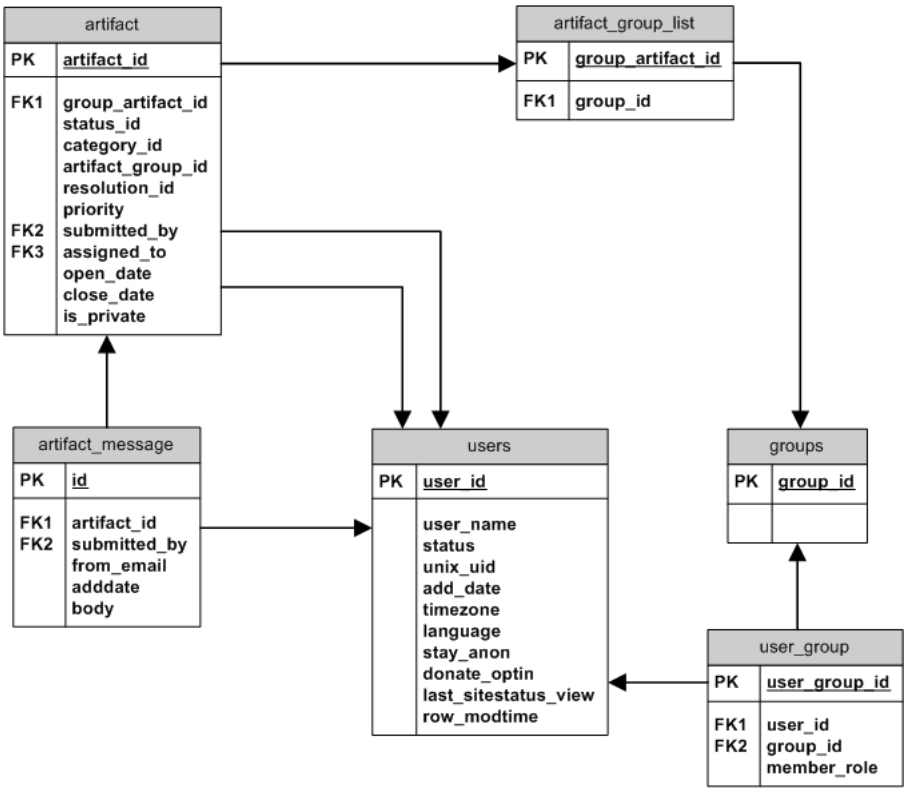


Figure 4.1: Open Biomedical Ontology Schema for Artifact Data

group and its domains. Each record in the table represents a domain, and is uniquely identified by an identification number. A group can have multiple domains, whereas a domain is always associated with a single group.

- **users**

  This table stores the list of all the members of the communities. The members submit the artifacts, comment on the artifacts and collaborate. Table lists all user information including their names and contact information.

- **artifact**

  Table "artifact" holds the artifacts submitted by the members of the community. Artifact is a generic term to represent numerous types of reports that are attached to a project. SourceForge automatically defines a number of default artifacts like bug reports, feature requests, support requests, and patches; however, projects can also define

their own type of artifacts. Each artifact record actually has two users associated with it; one user is the person who created or submitted the artifact record, and the other user is the person assigned to the artifact record. When a user contributes an artifact, the contribution is made towards a domain which is designated by "group_artifact_id". The same artifact cannot be associated with a single domain.

- **artifact_message**

  When a member submits an artifact other users are free to comment or elaborate on it. Each entry in the table "artifact_message" represents a single elaboration. Details like who submitted the elaboration, when it was submitted, which artifact it was submitted for and the body of that message is stored in the table. There can be multiple elaborations towards each artifact.

- **user_group**

  When a user gets registered with the community he/she gets affiliated with that group. Table "user_group" stores this affiliation information. Each user in a group has a member role which decides the rank of the user for that group. User can be core-developer, co-developer, admin, etc. In addition, an user can be the member of multiple groups, and different member roles can be assigned to the user; one for each group.

4.1.2 Extension

With original schema alone, the calculation of some of the complex metrics becomes computationally extensive; so we decided to include some additional tables that extend the original schema, and facilitate the calculations of these metrics. The new schema contains precalculated results stored in the tables. This helps in reducing the input-output calls made towards the database during calculation of the metrics. The extended schema is shown in figure 4.2.

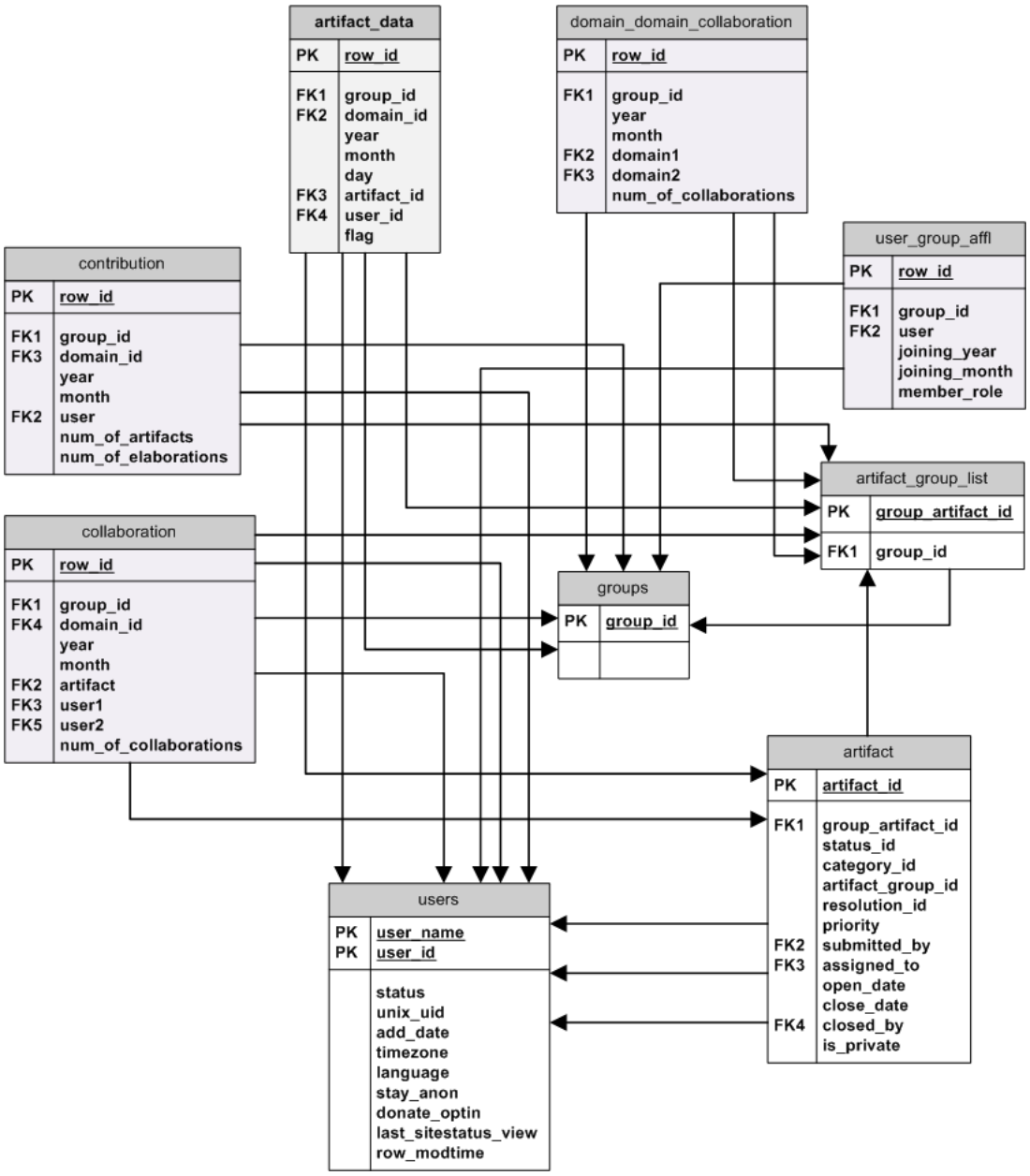


Figure 4.2: Extended Schema for Artifact Data

- **collaboration**

  The "collaboration" table stores the data about user-user collaboration that helps in calculating "Collaboration Intensity" factor of "Activity Strength" metric (explained in Chapter 6). This table stores edges between users for given group, domain, year, month and artifact which makes it very easy to slice and dice the data according to the various dimensions thus providing flexibility.

- **contribution**

  The "contribution" table stores the details about contribution of the individual users in terms of artifacts and elaborations and thus helps in calculating "Contribution" factor of the "Activity Strength" metric (explained in Chapter 6). The table stores number of artifacts and elaborations submitted by the user for a given group, domain, year and month. So that, it makes it very easy to slice and dice the data according to the various dimensions thus providing flexibility.

- **user_group_affl**

  According to SourceForge, the user is the member of the group with which he/she is registered. This relationship is misleading, because a user being registered with one or more groups does not mean that he/she would contribute only for those groups. It is upto user's discretion to submit an artifact to the domain of their choice. If we have this relationship, then we tend to neglect the contributions which were made by the user for the groups with which they are not officially registered. This prompted us to define a new table "user_group_affl", not available in the original artifact schema. The table gives the affiliation between users and groups based on newly defined relationship which differs from the relationship between user and group at the database level. Now, the user becomes the member of the group only if he/she contributes an artifact for the group.

- **domain_domain_collaboration**

  In order to construct the domain-domain graph, "domain_domain_collaboration" table is built so that edges between different domains can be retrieved efficiently from the database. Moreover, this also helps in calculation of certain structural metrics like "Preferential Attachment" and "Degree Distribution" (to be discussed in later sections).

- **artifact_data**

  This table is a combination (equi-join) of "artifact" and "artifact_message" tables from original schema, with the only exception of the date field. The dates in the original schema tables are stored in UNIX epoch time. But in our case, we need the dates in the form of month and year. As a result, the dates in the artifact table are converted to month, year format and stored in this table. In order to differentiate the users who submit the artifact from the users who elaborate on it, we have a column called flag. Setting the flag as 'S' indicates that the user submitted the artifact; where as, setting it as 'E' indicates that user elaborated on the artifact.

As we can see from figure 4.2, the extended schema forms a star in which tables "collaboration", "contribution", "domain_domain_collaboration" and "artifact_data" form the FACTS that are surrounded by the DIMENSIONS: "group", "artifact", "artifact_group_list" and "users"- the tables from original schema.

Chapter 5

Implementation of SciBrowser Tool

This chapter gives the details about implementation of the SciBrowser tool. It explains entire software engineering approach followed in terms of various stages of software process. The Chapter is divided into 4 sections namely: Requirement Analysis, Design, Verification and Validation. Each of these sections resemble a specific stage of software process that we follow.

5.1 Requirement Analysis

5.1.1 Purpose

Primary purpose of the project is to analyze the Open Biomedical Ontology (OBO) network from the perspective of Complex Adaptive Systems (CAS). Analysis includes demonstrating the properties of a CAS, such as power law, scale free network, preferential attachment etc. Our objective is to search for these properties and find out if they do or do not exist. These properties can be observed in different types of graphs mentioned in [25], where the interpretation of the properties vary accordingly. The properties targeted are mainly the structural attributes of a social network. Apart from the structural aspects, the tool also focuses on the behavioral properties of the the social network that are inclusive of the collaboration as well as the activity taking place in the network. It visualizes as well as quantifies the collaborations between members of the network, and also displays the activity distributions over time.

Following is the list of requirements:

- Visualize the Degree Distribution, at the Group/ Domain level, and for a given time frame.
- Visualize the Preferential Attachment phenomenon, at the Group/ Domain/ User/ Artifact level, and for a given time frame.
- Visualize the Activity Strength Distribution of users, at the Group/ Domain level, and for a given time frame.
- Visualize the Artifact Contribution Distribution of users, at the Group/ Domain level, and for a given time frame.
- Visualize the Collaboration Intensity Distribution of users, at the Group/ Domain level, and for a given time frame.
- Visualize the Activity Distribution, which includes visualizing the Active User distribution and the Artifact Contribution distribution at the Group/ Domain level, and for a given time frame.
- Visualize both, the cumulative and the non-cumulative forms of activity.
- Month-wise aggregation of the activity and the preferential attachment plots, within a given time frame.
- Comparisons of the domains in terms of their activity.
- The plots displayed in the image panel should be interactive in the sense that the user should be able to zoom in and zoom out in order to see the fine details of the plot.

Options for each metric are mentioned in tables 5.1, 5.2 and 5.3.

All the metrics in the discussion are calculated for the selected group and domain during a certain time frame, and in some cases for certain artifacts or users as well. All these parameters are required as the input for the calculation of the metrics. The user should be

Type of Metric	Metric	User Options
User User	Preferential Attachment	Type of Plot, Group, Domain, From Year, From Month, To Year, To Month, User, Time Interval
	Degree Distribution	Type of Plot, Group, Domain, Year, Month
User Artifact User	Preferential Attachment	Type of Plot, Group, Domain, From Year, From Month, To Year, To Month, Artifact, Time Interval
	Artifact Degree Distribution	Type of Plot, Group, Domain, Year, Month
	User Degree Distribution	Type of Plot, Group, Domain, Year, Month
User Domain	Preferential Attachment	Type of Plot, Group, From Year, From Month, To Year, To Month, Domain, Time Interval
	Domain Degree Distribution	Type of Plot, Group, Year, Month
	User Degree Distribution	Type of Plot, Group, Year, Month
Domain Domain	Preferential Attachment	Type of Plot, Group, From Year, From Month, To Year, To Month, Domain, Time Interval
	Degree Distribution	Type of Plot, Group, Year, Month

Table 5.1: Structural Analysis Metrics

Metric	User Options
Artifact Contribution Distribution	Type of Plot, Group, Domain, Year, Month
Activity Strength Distribution	Type of Plot, Group, Domain, Year, Month
User Collaboration Map	Type of Plot, Group, Domain, Year, Month
Collaboration Intensity Distribution	Type of Plot, Group, Domain, Year, Month

Table 5.2: Collaboration Analysis Metrics

Type of Metric	Metric	User Options
Cumulative/Non-Cumulative	Active User Distribution	Type of Plot, Group, Domain, From Year, From Month, To Year, To Month, Time Interval
	Contribution Distribution	Type of Plot, Group, Domain, From Year, From Month, To Year, To Month, Time Interval
	Domain Active User Comparison	Type of Plot, Group, Domain List, From Year, From Month, To Year, To Month, Artifact, Time Interval
	Domain Contribution Comparison	Type of Plot, Group, Domain List, From Year, From Month, To Year, To Month, Artifact, Time Interval

Table 5.3: Activity Analysis Metrics

able to select the parameters required to calculate the metrics, from the drop down choices provided on the panel and see the result.

5.2 Design

Figure 5.1, shows the comprehensive class diagram of the SciBrowser application. Details like method signature and variable names have been eliminated from the class diagram due to space constraint. To start with, we focus on the various packages this class diagram is made up of; they are listed as follows.

- **Data:**

  Data package is composed of the classes that hold the data to be displayed on the user interface in the form of the options on the control panel (*ControlPanelOptions*), and the metadata which is used in order to calculate the metrics. The class *ControlPanelOptions*, holds the data to be displayed on the control panel for all 3 pages. The class *Data*, holds the different types of method binding data. For example, each metric has a specific method used for its calculation; thus, there is a binding between metric name and its method name in the form of a dictionary– a data structure in python.

The class, *LoadParameterData* is used to load the parameter values in the control panel, directly from the database. The options for different groups or domains, used in our analysis are loaded using this class. Finally, *Parameter* class is used to hold the parameters selected by the user, which are then used in the metric calculation.

- **Metric Interfaces:**

  The package *MetricInterfaces*, holds the common interface used to calculate all three types of metrics (to be discussed in detail in following sections). *IMetrics* is the interface for metrics, *IToolkit* is the interface for all the toolkits used in calculation of metrics and *IQueryConstructor* is the interface all the query constructors used to generate queries. Although, python does not have concept of Interfaces we still enforce this concept by creating the classes and having unimplemented methods inside it. If in a class, just the method signatures are defined with "pass" keyword in the body of the method, we call it as "Interface".

- **Connection:**

  The *Connection* package has 2 classes: *DatabaseConfig* and *DBConnect*. The class *DatabaseConfig*, holds all the parameters such as host, port number, database name, username and password that are required to connect to any database. All these parameters are static or class variables. The class is not singleton i.e. we can create multiple instances of the class. Thus, if we want to get/set any of these parameters (using getters and setters), then we just need to create an instance of the class, and use getter/setter methods to alter these parameters. The class *DBConnect*, returns the connection object to be used to query the database during metric calculation. Again, connection object is a static variable which makes it possible to have single point of reference for connection.

- **Threading:**

  The class *Thread* in *threading*, module is used to generate a background thread for

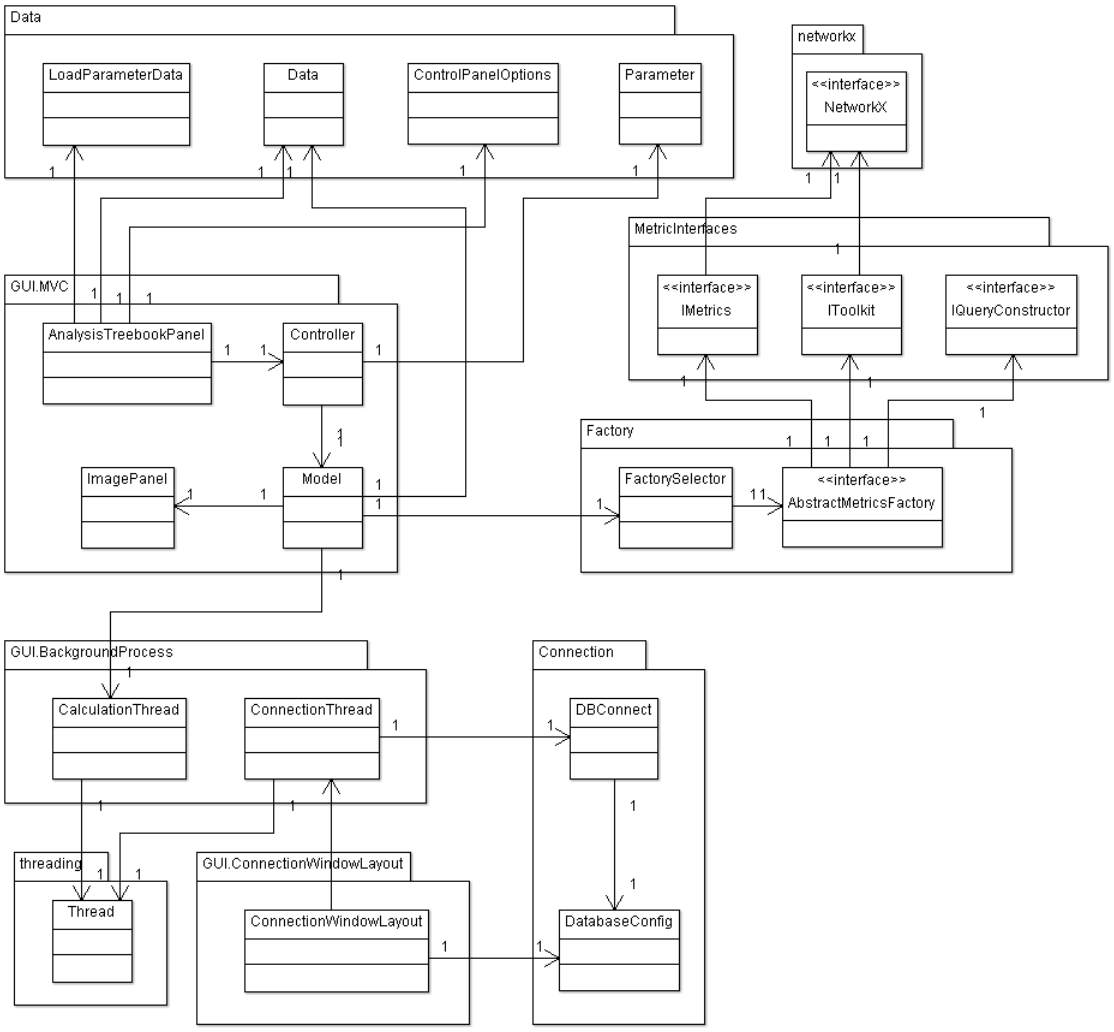


Figure 5.1: Comprehensive Class Diagram

metric calculation or connection to the database. This makes it possible to have a progress bar displayed for the user, while calculation or connection process is going on in the background.

- **GUI:**

  Detailed view of the GUI class diagram can be seen in the figure 5.11. Figure 5.1, shows only the important parts of the *GUI* module. *GUI* package consists of number of sub packages. The package *GUI.MVC*, is the dedicated package for the implementation

of Model View Controller [21] design pattern. Both, *ImagePanel* and *AnalysisTreebookPanel* (which is a part of control panel) act as the views which call the *Controller* instance. The *Controller* instantiates *Parameter* class, and stores all the parameter values in that class. The instance of the class *Parameter*, is passed to the *Model* for metric calculation. The *Model* calls the instance of class *CalculationThread* to calculate the metric, and then updates the *ImagePanel*. *ImagePanel* registers with the *Model* initially when the application starts. Thus, the *Model* is aware about the view it needs to update. *GUI.BackgroundProcess* package works in association with *threading* module, in order to generate background threads for metric calculation and connection to the database.

- **Factory:**

  Factory package comes into play when the class *Model* (in the *GUI.MVC* package) needs to calculate a metric. *Factory* returns right kind of object for the metric to be calculated. The package *Factory*, is explained in details in Metric Selection section below.

5.2.1 Helper Tables

The database schema provided by SourceForge, is alone insufficient to efficiently calculate the metrics. As a solution to this problem, we came up with a new schema which has the data in the relevant format. The new schema expedites the calculation of metrics in terms of the time complexity, by reducing the number of input-output calls made to the database. The new schema forms the basis for the calculation of all the metrics we need. Tables in the new schema are designated as the Helper Tables, as they assist a great deal in efficient calculation of metrics. The conversion from old schema to new schema is done through a set of python programs.

5.2.2 Metrics

Metrics are classified into three types: Structural Analysis, Collaboration Analysis and Activity Analysis. Although they are significantly different modules, they have same dependency structure which we call as 3 level dependency structure. The module that sits on the top of the structure is the high level module, and is named as *Metric* module. Metric module is represented by an interface which defines the methods that calculate metrics. The actual implementation of the metric method is done by the classes that implement the interface. In addition to that, the concrete class can also have its own methods to calculate some of the specific metrics.

Module at the second level is called as *Toolkit* module, and is represented by the interface that defines the common methods that are needed by the *Metric* module. So, it is needless to say that the *Metric* module depends on the *Toolkit* module. Any concrete class that implements the *Toolkit* interface can define its own methods.

Module at the third and the bottommost level is called *Query Constructor* module, and is also represented by the interface which defines the common methods required by the *Toolkit* module. *Toolkit* module communicates with the database to acquire data and *Query Constructor* module assists *Toolkit* module in data acquisition, by providing the relevant SQL queries that need to be fired on the database for fetching the required data.

At each level there are interfaces, and higher level concrete classes are dependent on the interfaces defined at the level below them. For instance, concrete classes in *Metric* module implement the higher level *Metric* interface and also depend on the *Toolkit* interface defined in the level below them. Similar is the relationship between *Toolkit* and *Query Constructor* modules. This implementation is inspired from the Dependency Inversion Principle [22] (one of the 5 principles of SOLID code). The principle states that details should depend on abstraction but abstractions should not depend on the details. In our case the concrete classes in the higher level module depend on the abstractions in the lower level module.

As a result, every level has to be represented by an interface. This approach has following advantages

- There is flexibility of modifying the lower level modules without modifying the higher level modules, because higher level modules depend on the abstraction rather than the concrete implementations. As long as the interface requirements are met there is not need to change the higher level modules which depend on the interface.

- The software can be easily extended with new code with out modifying the existing code. According to the Open Closed Principle (OCP) [23] (yet another principle of SOLID coding), the software should be built in such a way that the entities (Classes, Modules, Functions, etc.) should be open to extension but closed for modification. This can be achieved via Dependency Inversion Principle. Suppose, in future we need to extend the software for a new type of metric; then a new method can be added in the existing class which implements the *Metric* interface, and corresponding *Toolkit* and *Query Constructor* methods can be added to the classes in their respective modules if required. Thus existing code doesn't have to be tempered with. Whereas in absence of this architecture, every time the higher level class depends upon a new lower level class, changes have to be made in the higher level class.

Each category of metrics is discussed below in detail along with their class diagrams.

Structural Analysis

Structural analysis module is shown in figure 5.2. As there are 4 different types of graphs, there can be 4 different interpretations of a metric. For instance, the method *popularityGrowthRate()* showing preferential attachment has four different interpretations for 4 different types of graphs. Thus, the *Metric* module has an interface *IGraphMetrics* and the classes that implement the interface will implement the metric in their own manner. In addition, the concrete classes also define their own metrics.

In *Toolkit* module (second level), the class *GraphToolkit* implements the interface *IGraph-Toolkit* which has signatures of common methods used by all the classes in the *Metric* module. There are additional methods needed by UserArtifactUserGraphToolkit and UserDomain-GraphToolkit classes which are defined in the corresponding classes These classes inherit from *GraphToolkit* class.

The third level is called as *Query Constructor* level. It takes care of all the SQL query generation required by the toolkit level. *Toolkit* level communicates with the database to query for the data and the queries that are needed are provided by the *Query Constructor* level. Interface *IGraphQueryConstructor* gives the methods that have different implementations for different types of graphs. We have 4 different classes (one for each type of graph) that implement this interface. Method specific to a certain type of graph are defined in the corresponding class.

Collaboration Analysis

Collaboration analysis module is shown in figure 5.3. These metrics do not depend on the type of graph. In fact, currently there is single implementation of collaboration metrics as can be seen from the *Metric* module in figure 5.3. But we are open to a different interpretation as well as implementation of these metrics and as a result, we have created an interface *ICollaborationMetrics* having signatures of these metrics.

The *Toolkit* class in the Toolkit module implements *ICollaborationMetricsToolkit* interface and inherits from *CollaborationMetricsToolkit* class. The reason behind this implementation is that, there are certain methods such as getting user list or domain list from the database which are standard methods having just single implementation. These are used through out the project. Thus, they are defined in the concrete class on which the higher level classes depend. Although this goes against the Dependency Inversion Principle it makes sense, because ultimately we send instance of *Toolkit* class as a argument to the methods in

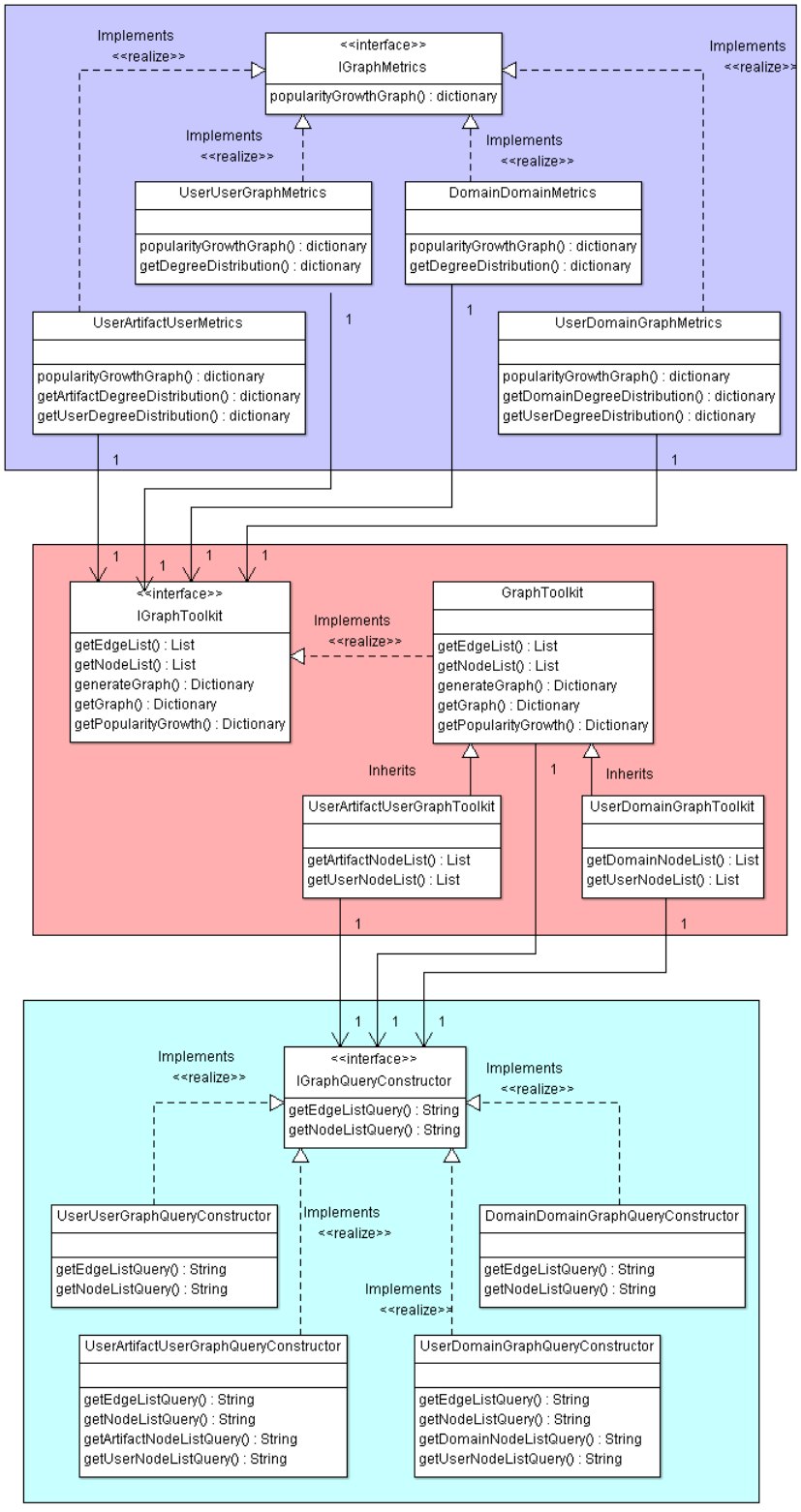


Figure 5.2: Structural Analysis Module

the higher level class. This instance is of type both *ICollaborationMetricsToolkit* and *CollaborationMetricsToolkit*, and hence, it has access to both the implemented methods and the base class methods. If we define a new class that implements *ICollaborationMetricsToolkit*, then we do not have to redefine the standard methods in *CollaborationMetricsToolkit* again in the new class. The new class can get them by inheriting *CollaborationMetricsToolkit* class. Same is the situation with the Query Constructor module.

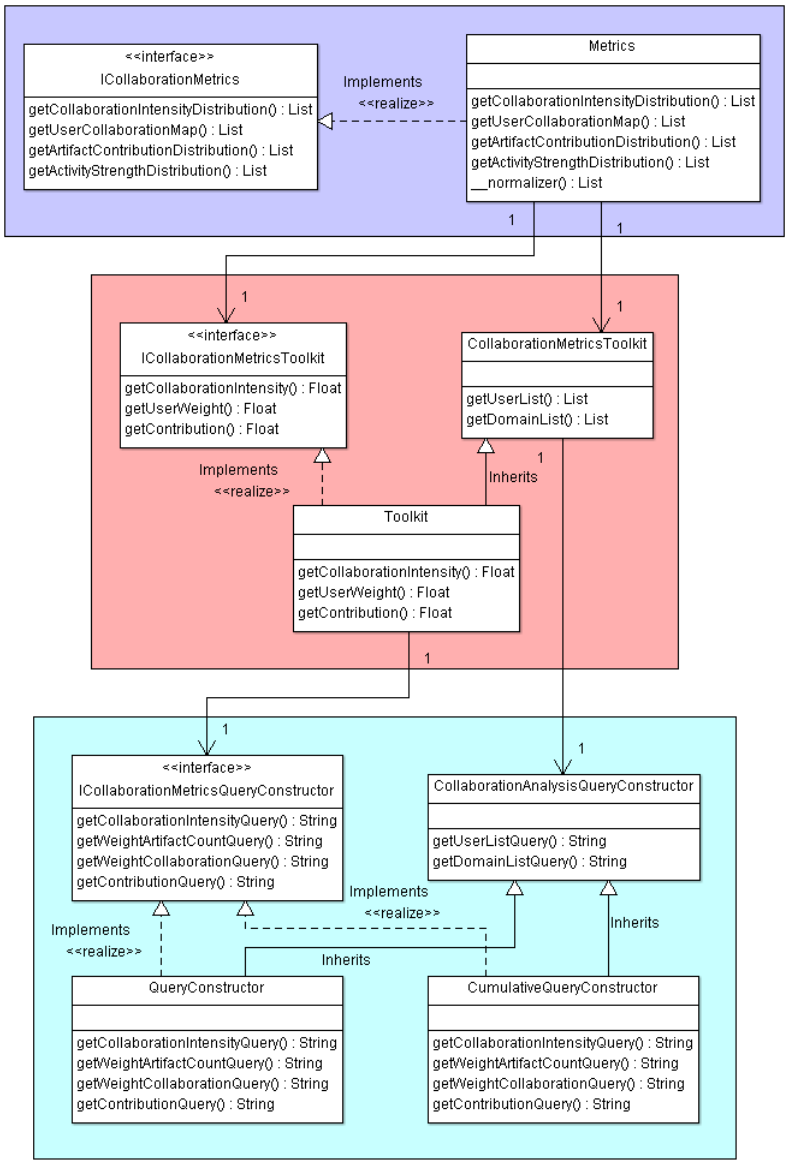


Figure 5.3: Collaboration Analysis Module

Activity Analysis

In this project we are targeting two types of activities Cumulative and Non-Cumulative. As a result, we have two implementations of each metric method defined in the interface *IActivityMetrics*. As far as *Toolkit* module is concerned, we just have a single implementation of the *IActivityMetricsToolkit* interface and for *Query Constructor* module there are two query builder classes as shown in figure 5.4.

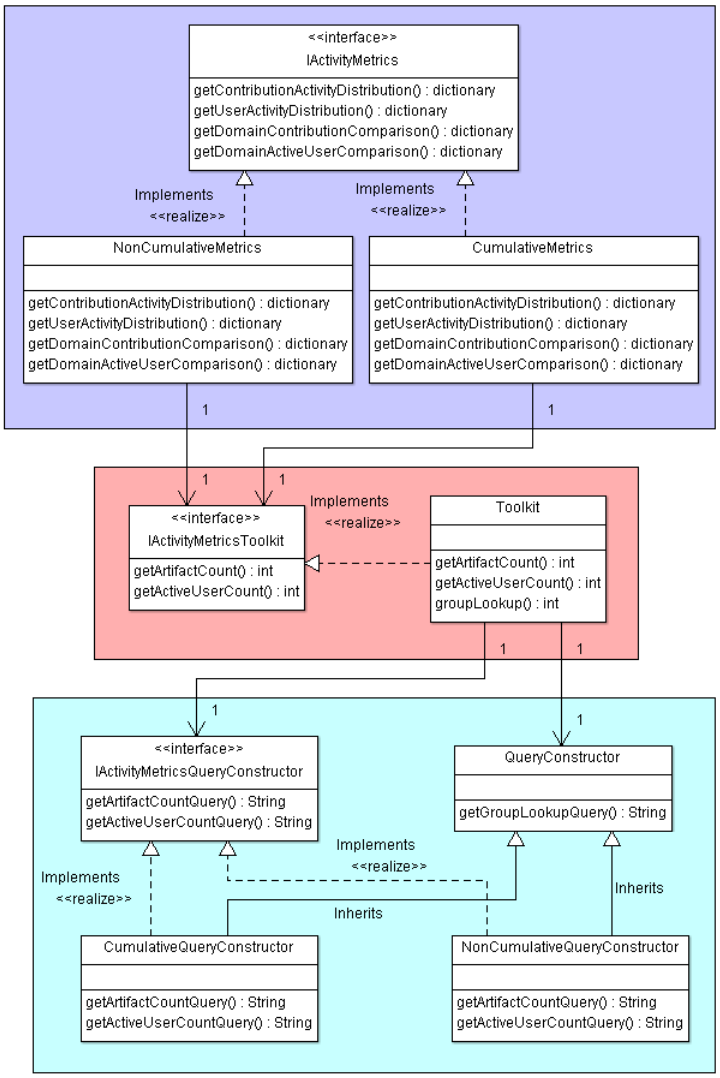


Figure 5.4: Activity Analysis Module

5.2.3 Metric Selection

Above sections are dedicated to metric calculation that happens in the backend. This section concentrates on how the intended metric is chosen for the calculation, based on the user input. User selects a bunch of parameters on the control panel that decide which metrics should be drawn on the screen for the user. As we have seen in the above sections that we have three categories of metrics. For calculating a metric be it any category, we need to select and instantiate three classes because of three level dependency structure. For instance, if we have to calculate Degree Distribution for User-Artifact-User graph then the classes that need to be instantiated are *UserArtifactUserMetrics* (from *Metric* Module), *UserArtifactUserGraphToolkit* (from *Toolkit* module) and *UserArtifactUserGraphQueryConstructor* (from *Query Constructor* module).

Thus, metric selection can be achieved by applying Factory Method pattern [12], as shown in figure 5.5. Three factories are built, one for each type of metric: structural, collaboration and activity. Each factory class has three methods with the same signature for producing the instance of a concrete class in *Metrics* module, *Toolkit* module and *Query Constructor* module for the given metric. This structure closely resembles Abstract Factory pattern [12], if we just have an abstract class or interface for the three factories that are created. Thus, we have Abstract Metric Factory as the interface for the three factories. This also requires us to have a common interface for each type of metric, toolkit and query constructor classes. In order to select the appropriate factory class based on the category of metric, we have a wrapper class called Factory Selector around the three factory classes.

5.2.4 GUI

When designing graphical user interfaces (GUI), we should use a solid design pattern or model so that the GUI would be stern and smooth in all its transitions. If a GUI contains several views or screens, or if it contains complex controls, it would not be wise to create the GUI on the fly without doing any prior design or without having any design base model.

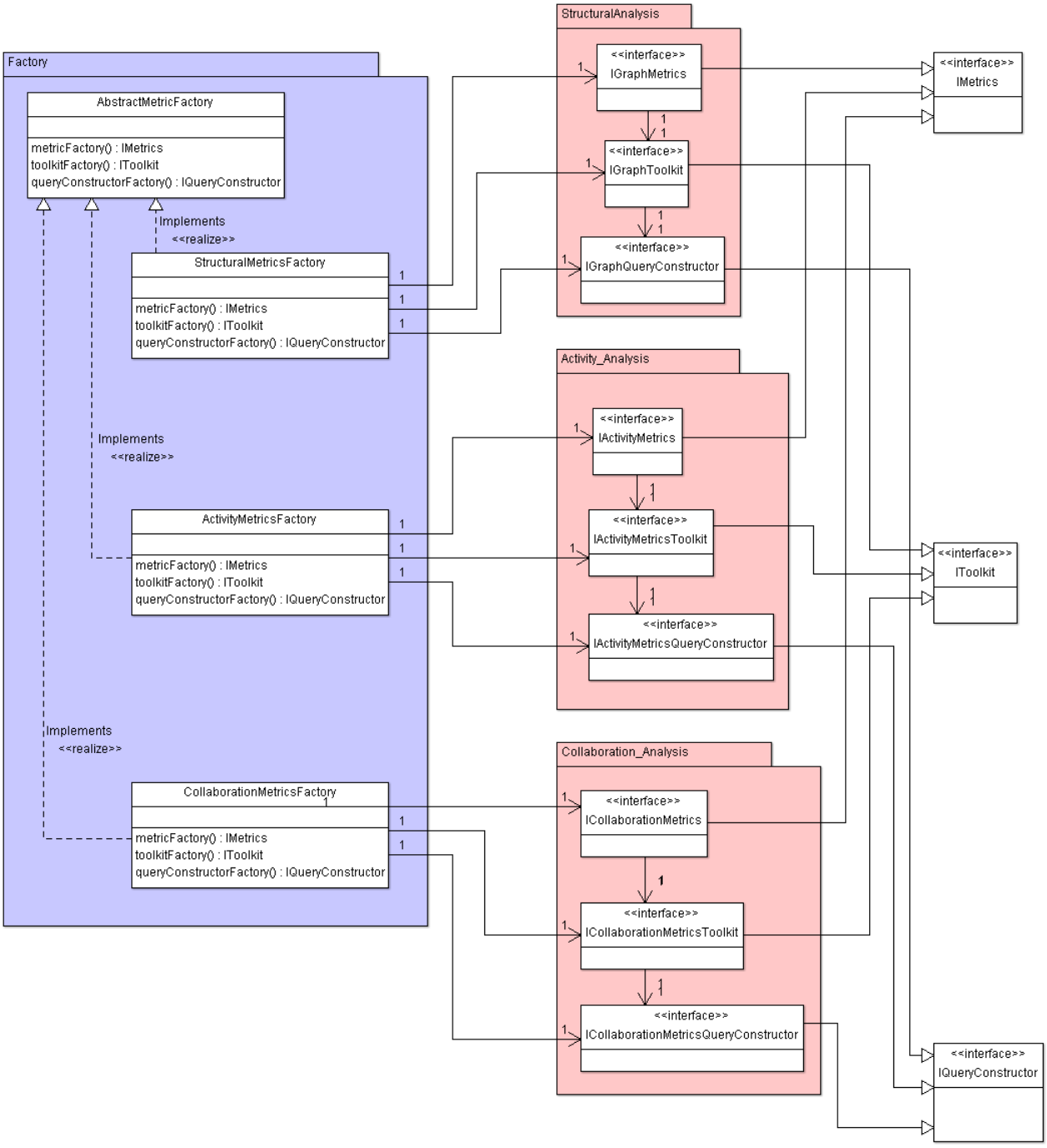


Figure 5.5: Metric Factory

MVC (Model View Controller) [21] is such design pattern that is used to model complex user interface. The MVC metaphor imposes a separation of behavior between the actual model of the application domain, the views used for displaying the state of the model, and

the editing or control of the model and views. Figure 5.6 shows the interaction between different modules of MVC.

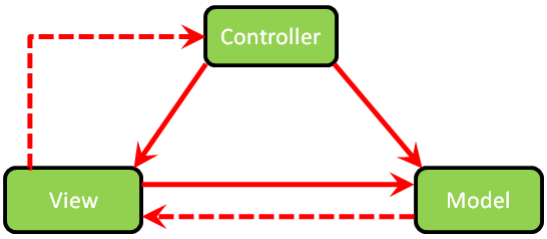


Figure 5.6: Model-View-Controller

Model manages the information and domain logic which in our case is the calculation of metrics. Our application uses a backend database that stores all the raw data, and calculation of metrics is built upon the data that is queried from the database. So the data access layer is assumed to be encapsulated by the model. When model changes its state it notifies all the views about the state change. View is the visual representation of the model and is comprised of screens and widgets used within the application. The view is shown in figure 5.7. We have just one view (image panel) to be updated. Controller responds to the user inputs such as button clicks, data entry or menu selection. Its acts as the link between user and application. Once request is received by controller it instructs the model to perform certain actions by making calls on model objects.

GUI for the tool is structured as shown in the figure 5.7. GUI window is divided into two segments called image panel and control panel. Image panel displays the graphs and the plots where as control panel displays the options to be selected by the user in order to get the desired plot. Control panel is a notebook having 3 pages namely Structural Analysis, Collaboration Analysis and Activity Analysis. Figure 5.8, 5.9 and 5.10 shows the sample graphs for three diferrent types of analysis. Each page corresponds to a specific type of analysis mentioned under “Metrics” subsection under “Design”. A tree structure of available metrics under the given analysis, is displayed on the left side where as the options associated with each of the metric are displayed on the right side. When user selects the appropriate

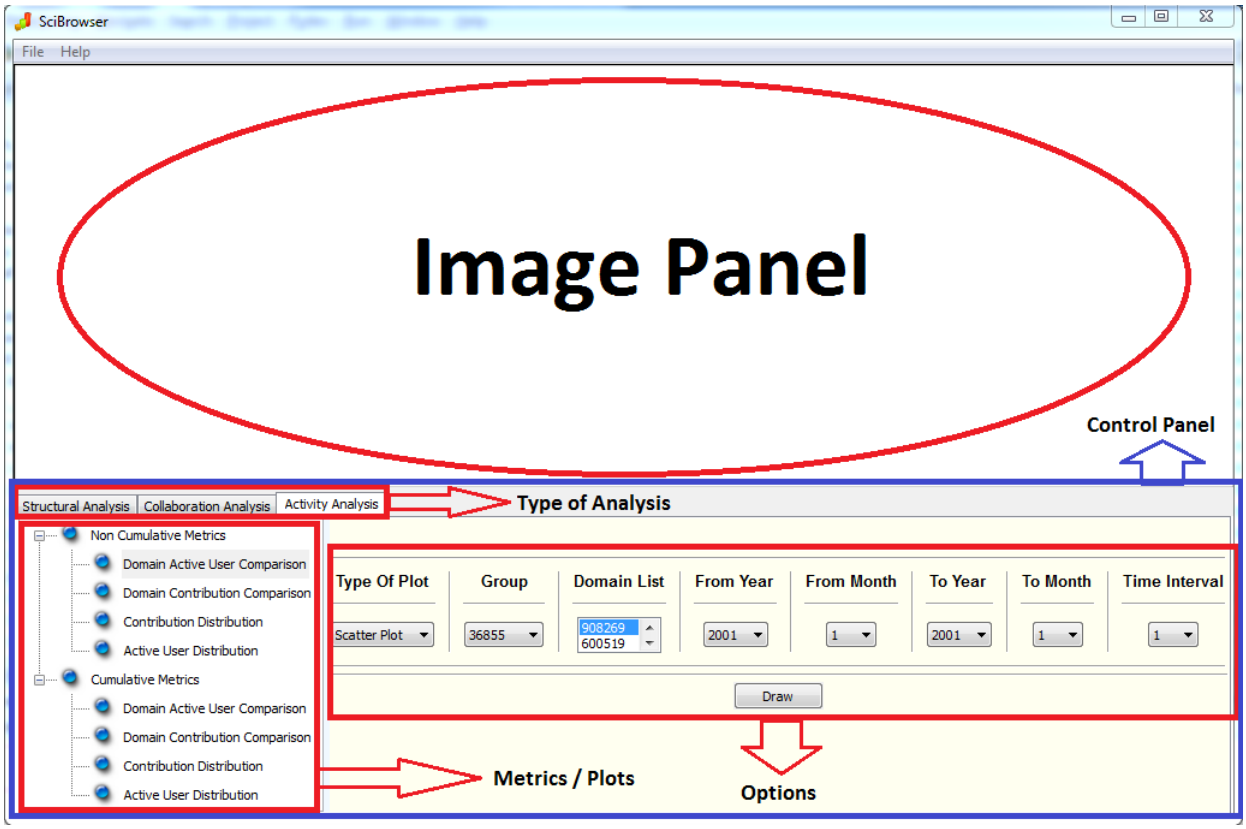


Figure 5.7: GUI Snapshot

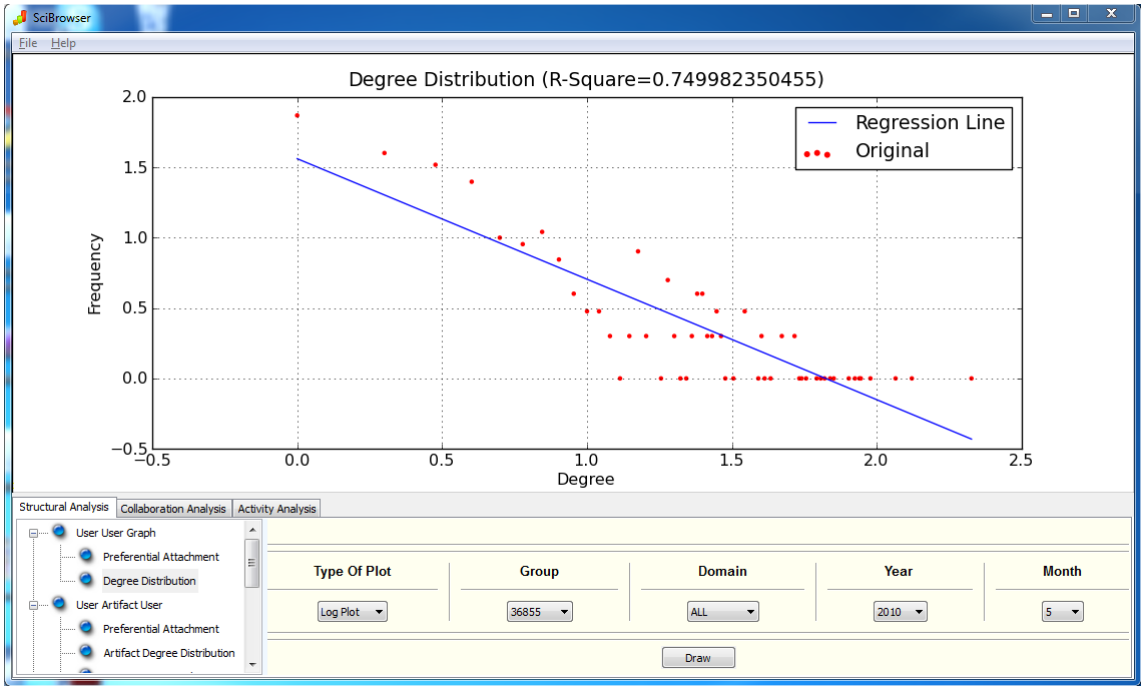


Figure 5.8: Structural Analysis Tab

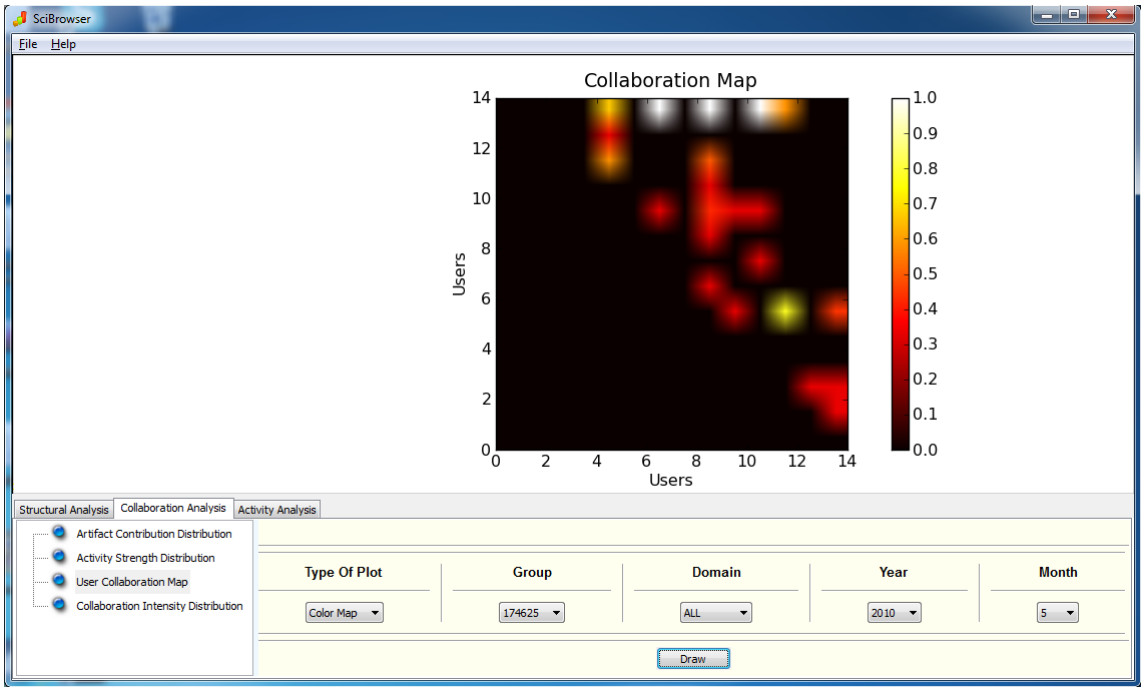


Figure 5.9: Collaboration Analysis Tab

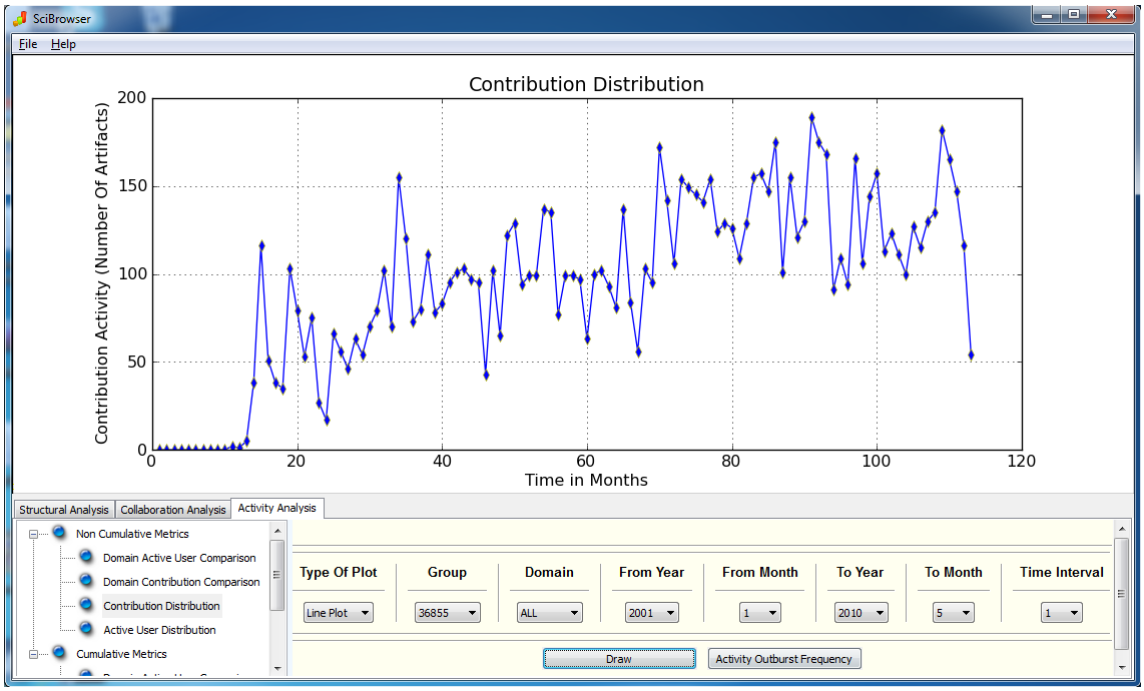


Figure 5.10: Activity Analysis Tab

set of options and clicks "Draw" button, user request goes to the controller which in turn sets the parameters in the *Parameter* class and instructs the model to calculate the desired

metric. Model runs the calculation as a background thread, and shows a progress bar to the user indicating that the calculation is taking place. When calculation is done model notifies the image panel and image panel updates itself with the new data from the model. Class diagram for GUI package (figure 5.1) is shown below in figure 5.11

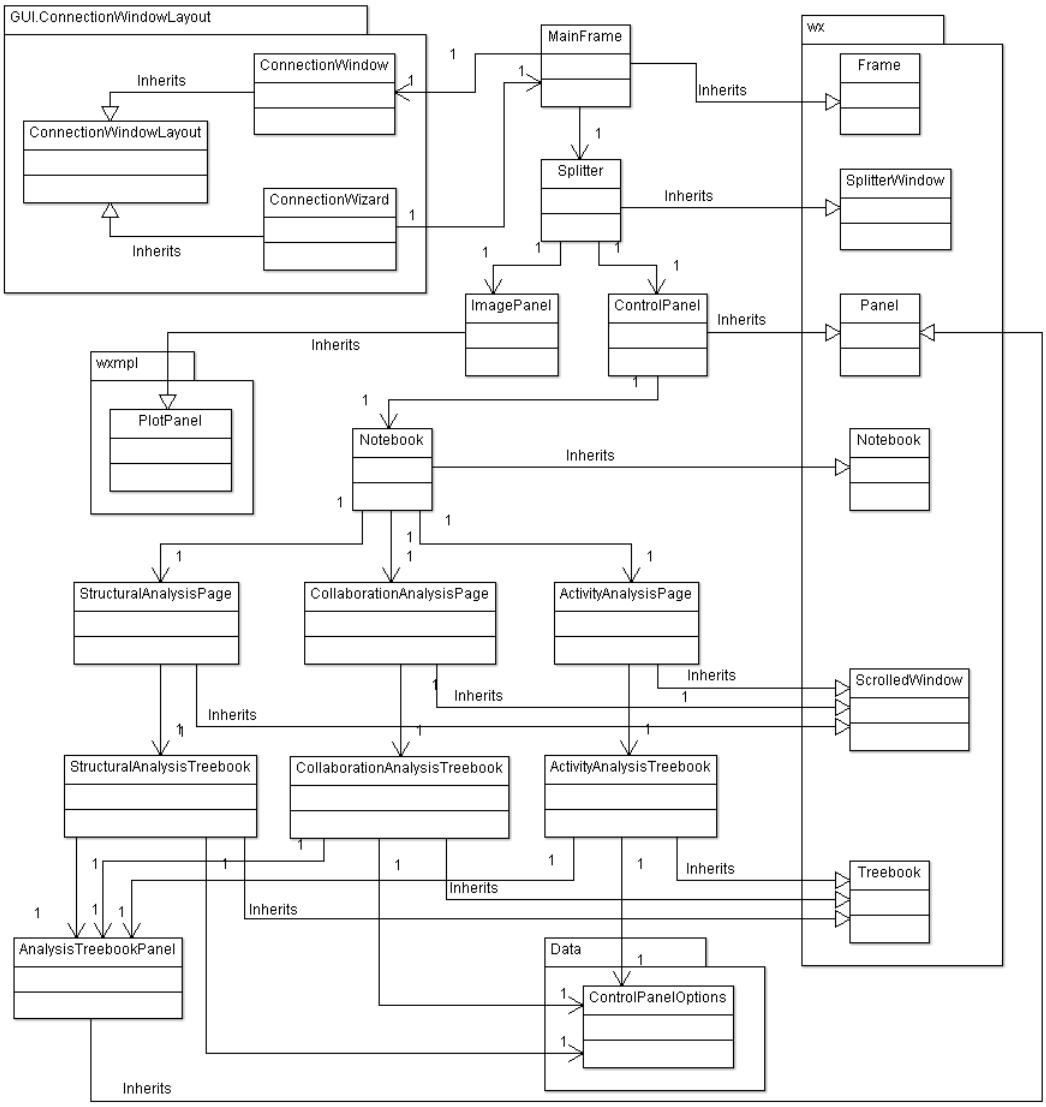


Figure 5.11: GUI Class Diagram

artifact_id	group_artifact_id	submitted_by	open_date
3	2	4	1107306000
4	1	2	1109725200
2	3	1	1136163600
1	1	2	1159750800
5	3	3	1178067600
6	4	1	1183338000

Table 5.4: artifact

5.3 Verification/Testing

Metrics calculated had to be tested for correctness before they were actually transformed into plots. Figure 5.12 shows components involved in testing. Each of these components have been explained below.

5.3.1 Test Database

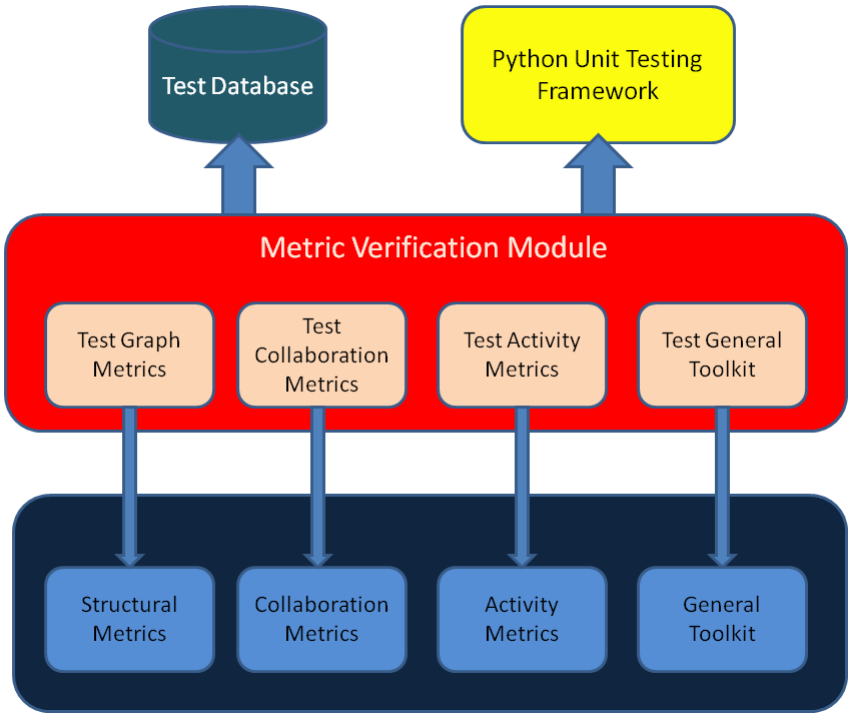


Figure 5.12: Testing Framework

user_id	user_name
1	user1
2	user2
3	user3
4	user4
5	user5
6	user6
7	user7

Table 5.5: users

group_id	unix_group_name
1	group1
2	group2

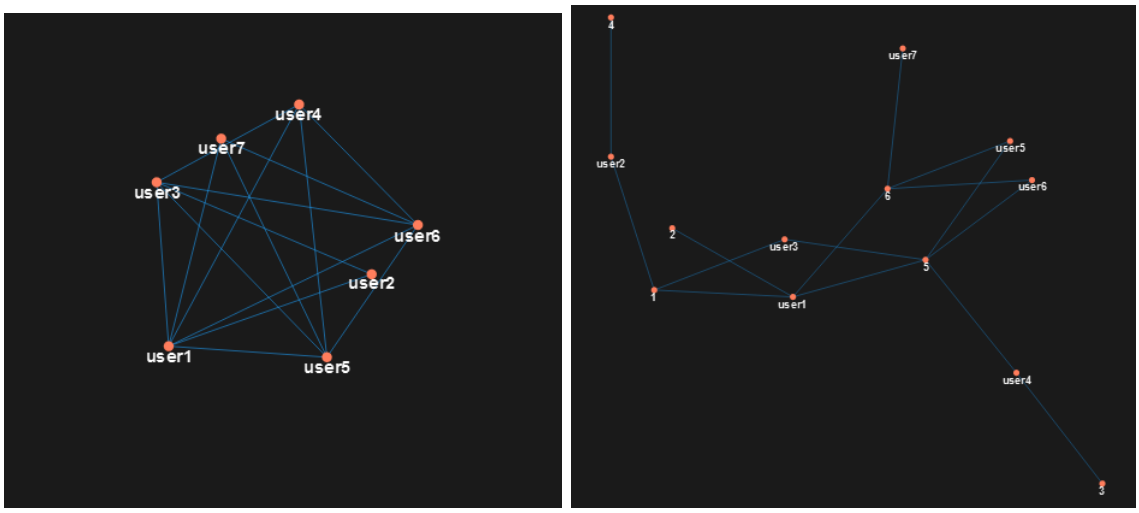
Table 5.6: groups

group_id	user_id	member_role	admin_flags
1	1	0	Y
1	2	100	N
1	4	101	N
1	5	100	N
2	5	0	Y
2	6	101	N
2	7	100	N
2	1	100	N

Table 5.7: user_group

group_artifact_id	group_id
1	1
2	2
3	2
4	1

Table 5.8: artifact_group_list



(a) User User Network (b) User Artifact User Network

Figure 5.13: Test Network

Test database "obo_test" is created in order to test the metrics. Schema of test database is a replica of original OBO database and consists of test data, loaded using SQL scripts. The test data is generated using a hypothetical network shown in figure 5.13. The hypothetical network is constructed by considering the boundary cases as well as intermediate test cases. First the network is sliced into time frames in such a way that time frames take care of all the test cases. The test data is shown in the tables below.

5.3.2 Metric Verification Module

Metric Verification Module consists of a Test Module for each category of metrics. There are three types of Test Modules corresponding to three types of metrics as mentioned in design section. Each Test Module consists of unit test cases in python that are tested using python's unit test framework PyUnit. Each unit test splits the network into a certain time frame using given parameters. Test cases and their results for some of the metrics are shown in tables 5.10, 5.11 and 5.12.

id	artifact_id	submitted_by	adddate
1	1	1	1160010000
2	1	3	1165280400
3	5	1	1178326800
4	5	5	1178499600
5	5	4	1181005200
6	5	6	1188954000
7	5	3	1188954001
8	6	5	1186275600
9	6	6	1189126800
10	6	7	1199494800
11	6	5	1186275601

Table 5.9: artifact_message

Test Case	Description	Result
group = 1, domain = 1, year = 2005, month = 1	Boundary Condition, No User Exist	{ }, Raise Exception "No Data Available for selected set of Parameters."
group = 1, domain = 1, year = 2005, month = 3	Boundary Condition, User exists but does not have a degree.	{"user2":0}
group = 1, domain = 1, year = 2006, month = 10	General test case where user exists and has a degree	{"user2":1,"user1":1}
group = 2, domain = 0, year = 2008, month = 1	Full Graph Condition for group_id = 2	{"user1":4, "user5":4, "user3":4, "user4":4, "user6":4}
group = 0, domain = 0, year = 2008, month = 1	Full graph condition for all groups i.e. cumulative	{"user1":6, "user5":5, "user3":5, "user4":4, "user2":2, "user6":5, "user7":3}

Table 5.10: Test Cases for Degree Distribution

Test Case	Description	Result
group=1, domain=1, year=2005, month=2	Boundary Contribution, No User Exist	{ } Raise Exception “No Data Available for selected set of Parameters.”
group=1, domain=1, year=2005, month=3	Single User Condition	{“user2”:0.5}
group=1, domain=4, year=2008, month=1	Full graph for group_id = 1	{“user1”:0.833, “user2”:0, “user3”:0, “user5”:0.5, “user6”:0.333, “user7”:0.333}
group=0, domain=0, year=2008, month=1	Full graph condition for all groups i.e. cumulative	{“user1”:0.9117, “user2”:0.6176, “user3”:0.75, “user4”:0.5441, “user5”:0.4705, “user6”:0.3823, “user7”:0.23529}

Table 5.11: Test Cases for Activity Strength Distribution

Test Case	Description	Result
group=1, domain=1, fromYear=2005, fromMonth=3, toYear=2005, toMonth=7, timeInterval=1	-	{1:1, 2:0, 3:0, 4:0, 5:0}
group=1, domain=1, fromYear=2006, fromMonth=9, toYear=2006, toMonth=12, timeInterval=1	-	{1:0, 2:1, 3:0, 4:0}
group=1, domain=1, fromYear=2005, fromMonth=9, toYear=2005, toMonth=8, timeInterval=1	Start Time comes before End Time	Raise Exception “Start Time Cannot be less than End Time”
group=1, domain=1, fromYear=2006, fromMonth=9, toYear=2006, toMonth=10, timeInterval=2	Time Interval is greater than the difference between Start and End time	Raise Exception “No Data Available for selected set of Parameters.”
group=0, domain=0, fromYear=2006, fromMonth=8, toYear=2007, toMonth=7, timeInterval=3	Time Interval greater than 1. Aggregating the data for every 3 months.	{1:1, 2:0, 3:0, 4:2}

Table 5.12: Test Cases For Contribution Activity Distribution

5.4 Validation

In this section we demonstrate the validity of the SciBrowser by showing that it successfully completes all the requirements specified in the Requirement Analysis section.

5.4.1 Structural Analysis

There are four different types of graphs [25] considered for this study. Structural analysis page broadly consists of two metrics: degree distribution and preferential attachment. As these metrics have different interpretations for different types of graphs they have been grouped according to the graph types. Degree distribution is a plot of the degree of a node v/s number of members having that degree. Besides line and bar, this plot can be viewed on logarithmic scale as it helps in accurate representation of the power law if it exists. Preferential attachment brings out the phenomenon of "rich gets richer". It represented by the plot of rate of change of degree with respect to time; thus, a constant increase in the rate of change of the degree indicates the presence of preferential attachment.

5.4.2 Collaboration Analysis

The metric, activity strength is introduced in this analysis. This metric considers both collaboration and contribution in its calculation; thus, it reflects the innovation potential of the users/members of the community. The metric is plotted on time axis in order to determine if there exists a similar distribution like power law. In addition, there is User-User collaboration map in order to view the interactions between the users. The areas that have high interactions are highlighted with lighter shades; whereas, the areas with low interactions are assigned the darker shades.

5.4.3 Activity Analysis

Under Activity Analysis page, we have metrics that display monthly distributions of number of artifacts submitted and number of active users in the network. We also compare the domains based on the activity using "Domain Active User Comparison" and "Domain Contribution Comparison" plots. In order to depict the frequency at which activity reaches/crosses a threshold level, the distribution called "Waiting Time Distribution" is used where the threshold can be set by the user.

Chapter 6

Social Network Analysis Using SciBrowser

As mentioned in the last chapter, SciBrowser analyzes the OBO community from three perspectives: structural, collaboration and activity. Structural analysis focuses on the social network analysis metrics like degree, centrality, density, clustering coefficient and average path length. Collaboration analysis focuses on collaboration between the users and innovation potential of individual user, using some novel metrics like "Activity Strength". Activity analysis concentrates on visualizing temporal distributions of activity to see how frequently the activity goes beyond a threshold which is a point of high activity. This chapter pinpoints the results of analysis performed by the tool. Each type of analysis, along with the results and their interpretations, is the highlight of this chapter.

6.1 Structural Analysis

Structural analysis is based on different types of graphs which we introduced in [25]. Following is the brief description of each type of graph that we used in our study and its implications on social network metrics. In each type of graph, node size is directly proportional to the degree of the node, and the strength of the connection between the nodes is indicated by the thickness of the line connecting the nodes. Definition of the connection strength varies according to the type of graph.

- **User-Artifact-User Graph:**

  In this network graph, the nodes are users and artifacts. The network depicts the contributions made by the users towards their group in the form of artifact submissions and elaborations. An artifact is created by exactly one user, while multiple users can

elaborate on it; thus, a given artifact has at least one connecting edge with the user who submits it, and there can be multiple edges based on these user elaborations. This graph shows the responses from other members of the network, and from these responses we can deduce how influential a given artifact will become. Figure 6.1 depicts the User-Artifact-User graph for the OBO hub group (group id: 125463 (ChEBI)). The nodes which are named as numbers indicate artifacts, whereas rest of the nodes named as alpha-numeric specify engineers and scientists (e.g., community members). The greater the number of elaborations an user makes towards an artifact, the stronger the connection strength between the user and the artifact.

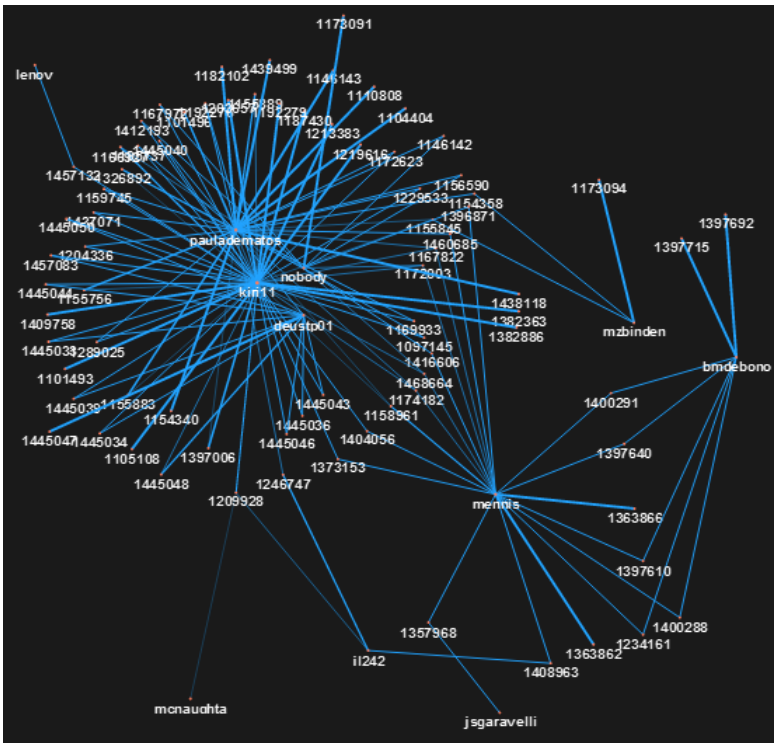


Figure 6.1: User Artifact User Network

- **User-User Graph:** The User-User graph is derived from the User-Artifact-User graph. We assume a transitive relationship between artifacts and users; if both member *A* and member *B* are connected to artifact *1*, we posit that member *A* is connected to member *B*. This allows us to simplify the graph and show only how users change over

time, and prevent information overload due to the artifacts. This graph can be termed as a collaboration graph as it displays collaboration between users. The greater the number of times users collaborate (indicated by number of comments shared between users), the greater the strength of connection between them. Figure 6.2 depicts the user-user graph for the OBO hub group (group id: 125463 (ChEBI)).

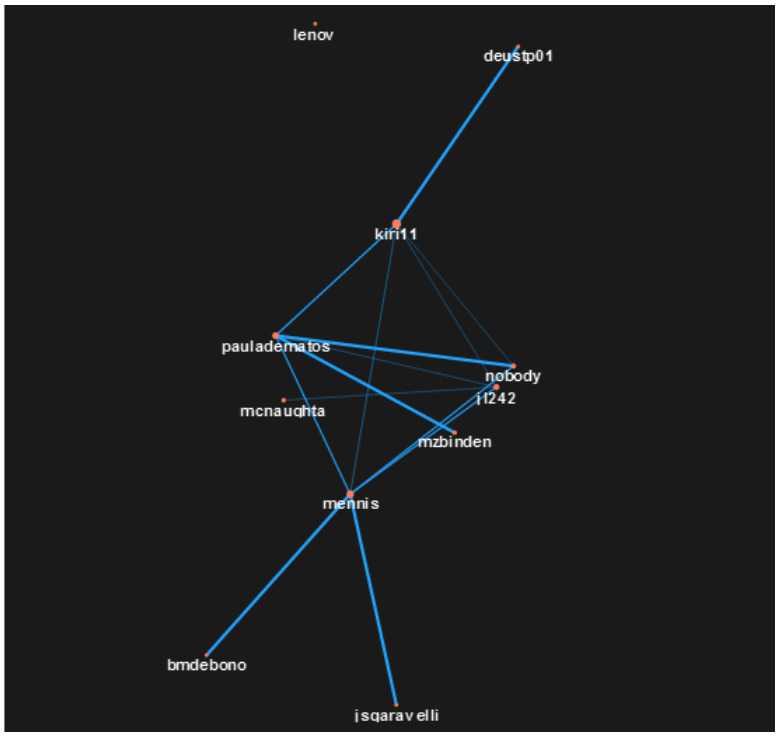


Figure 6.2: User User Network

- **Artifact-Artifact Graph:** The Artifact-Artifact graph is also derived from the User-Artifact-User graph, by connecting together the artifacts that are linked to the same member. The greater the number of members linked to the pair of artifacts, the stronger the link between the two artifacts. This graph depicts the information flow at the artifact level. Figure 6.3 shows the artifact-artifact graph for the OBO hub group (group id: 125463 (ChEBI)).

- **User-Domain Graph:** A group has one topic of focus, but these topics can be broken further into subfields in a similar manner to a phylogenetic tree. These communities

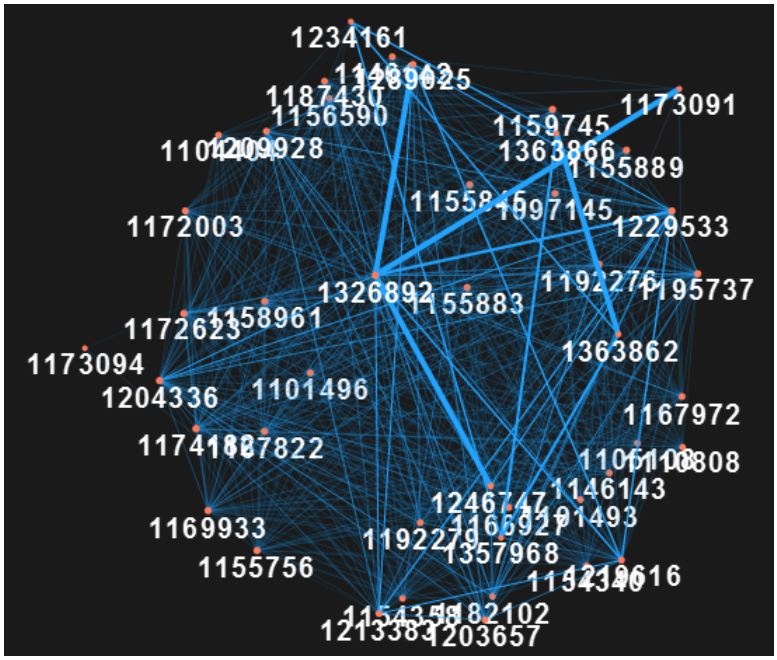


Figure 6.3: Artifact Artifact Network

are laid out in a similar fashion. If we take any particular group, this group will be composed of one or more subject area's or domains. The subject areas are the focus of the User-Domain graph. Figure 6.4 depicts the User-Domain graph for the OBO hub group (group id: 36855 (Gene Ontology)). Each group in OBO has subgroups (i.e., domains) that focus on specific subject areas. Each artifact is submitted for a specific domain. By abstracting the artifacts of the User-Artifact-User graph onto domains they belong, we derive a low-resolution and abstract network representation that denotes distribution of members onto subject areas. The nodes named as numbers indicate domains, whereas the rest of the nodes named as alpha-numeric specify engineers and scientists (e.g., community members).

- **Domain-Domain:** The Domain-Domain graph is derived from the User-Domain graph. We assume a transitive relationship between users and domains similar to what we have in the User-Artifact-User graph above; if member *A* is connected both to domain *1* and domain *2*, we posit domain *1* is connected to domain *2*. Furthermore, the greater the number of members common to a pair of domains, the stronger

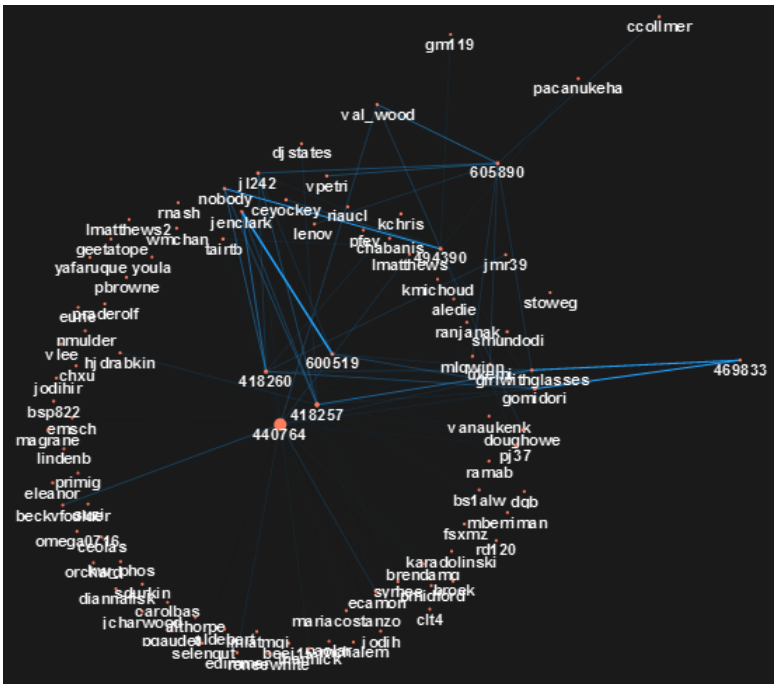


Figure 6.4: User-Domain Network

the relationship between the domains. Members of the groups act as the medium of knowledge transfer between domains (in Domain-Domain graph) or artifacts (in Artifact-Artifact graph). This rationale gives us an idea of how well the knowledge transfer takes place between domains– the concept which fosters innovation. Figure 6.5 depicts the domain-domain graph for the OBO hub group (group id: 36855 (Gene Ontology)).

It was observed that despite network diversity, most of the real web-like systems share three prominent structural features: small average path length (APL), high clustering and scale-free (SF) degree distribution [2][37]. Although the SciBrowser tool is used to visualize degree distribution and preferential attachment, we did not limit our study to these metrics; rather, we have done comprehensive social network analysis of the OBO community.

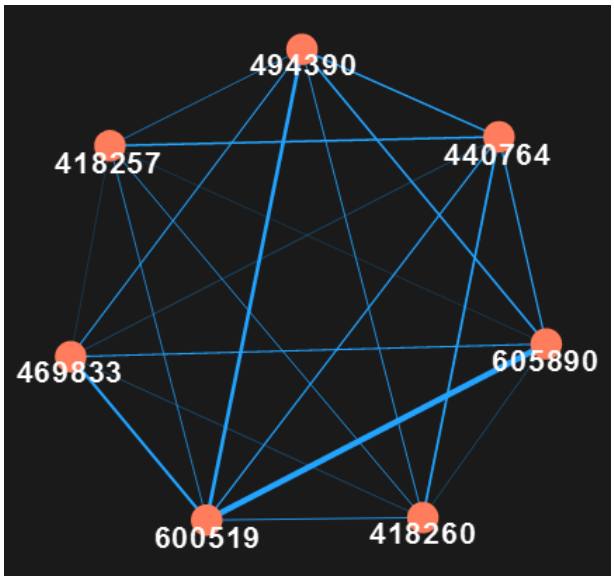


Figure 6.5: Domain Domain Network

6.1.1 Centrality

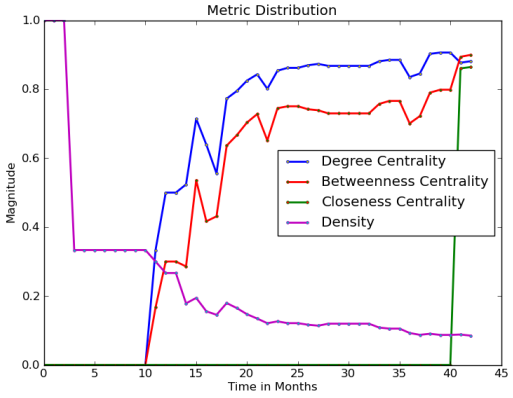
For the User-Artifact-User graph, the artifact with high degree-centrality will have ties to most of the users which means large number of users contribute to the artifact. For the User-User network, a highly central user will collaborate with most of the users, and hence reflect high collaboration intensity. For the User-Domain and the Domain-Domain networks, the central domain indicates that many users contribute to the domain which makes it an active and important domain. High closeness centrality in an User-Artifact-User graph indicates that the artifact is easily accessible to most of the users. In the User-User graph the user having high closeness centrality can reach all other users easily which facilitates communication. In Domain-Domain network high closeness centrality indicates that knowledge diffusion from one domain to the other will be smooth.

Figure 6.6 shows the monthly distributions of different types of centralities and density. Some of the projects like Open Biomedical Ontologies (76834), Disease Ontology (79168), and Systems Biology Ontology (174625) have the closeness centrality value as 0 because the graphs of these projects are disconnected. Sequence Ontology (72703) has values of all centralities in the range 0.85 - 0.95 which is very high. This network has the structure close to

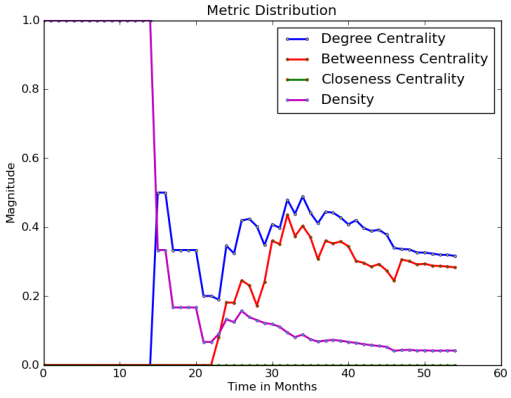
a star network which makes it evident that the network has a small core and a large periphery. Presence of the core members keeps the community active as they heavily contribute to the community. Presence of peripheral members helps keep constant flow of novel ideas into the community, as the peripheral members are the links between the community and the outside world.

6.1.2 Small World Phenomenon

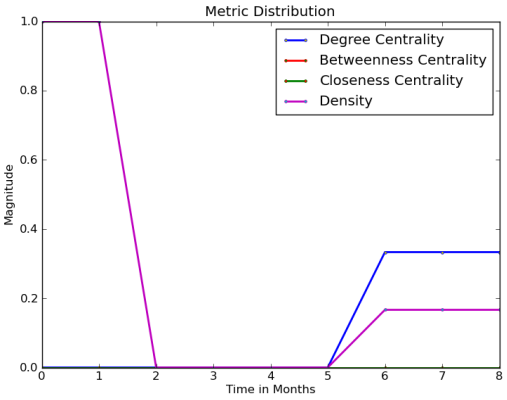
Clustering Coefficients (CC) and Average Path Lengths (APL) define the small world phenomenon for networks [35]. Month-wise distributions of CC and APL for the User-User graphs of various projects are shown in figure 6.7. CC indicates how complete the subgraph is for the user in discussion. If the neighbors of a user are fully connected it means that the CC for the user is 1. Whereas, if the user's neighboring network is fully disconnected then the CC for that user is 0. Thus, if the CC is high or very close to 1, then most or all of the neighbors of the user will be connected to each other, creating a uniformity in the knowledge level of the individual users in the clique. Simply, if everyone in the group has the same knowledge as every other individual, then there's no diversity– one of the important factors in fostering innovation and creativity. Also having a 0 value for CC means that there is no communication between the users that are connected to the user in discussion. This is not advisable, as the knowledge mobility is suppressed. Thus, it is preferable to have a value of CC between 0 and 1, as it indicates the presence of highly connected subgroups (within the project) which are loosely connected to each other. As seen in Figure 6.7, most of the groups have their CC value between 0 and 1 which is an indication of the existence or the possibility of existence of creativity in the groups. APL is defined as the average number of nodes it takes for any node to get to any other node in the graph. The smaller the APL, the faster the information diffusion in the graph which favors the condition for the small world phenomenon.



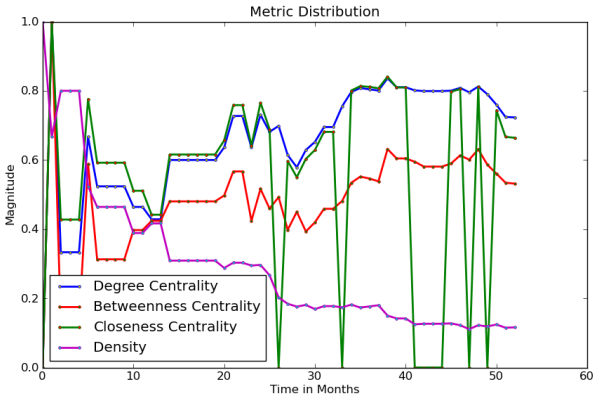
(a) Sequence Ontology (72703)



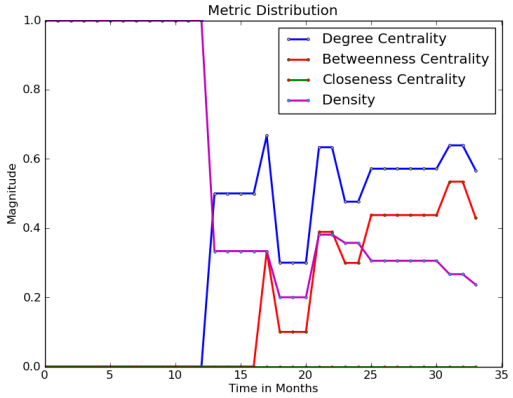
(b) Open Biomedical Ontologies (76834)



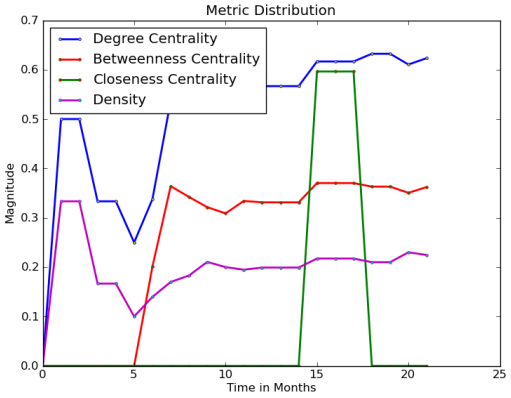
(c) Disease Ontology (79168)



(d) ChEBI (125463)



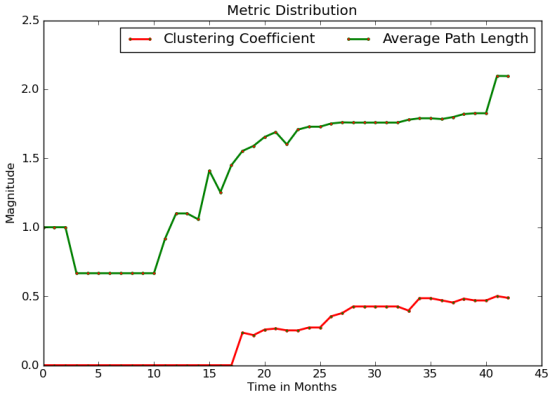
(e) Systems Biology Ontology (174625)



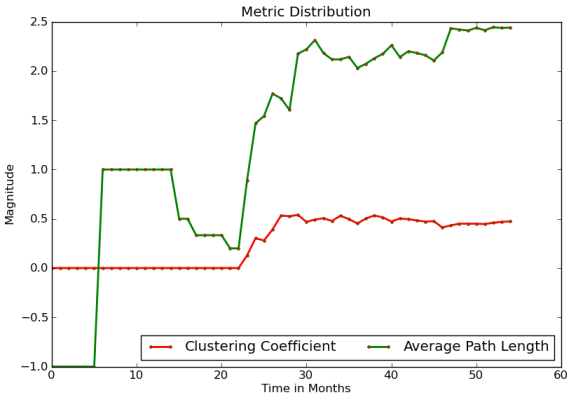
(f) OBI (177891)

Figure 6.6: Centrality and Density Distributions

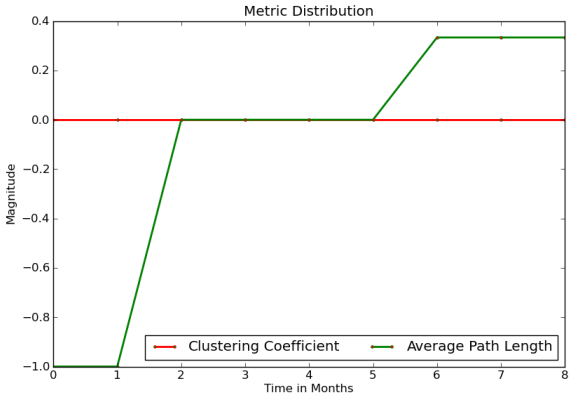
It is worth noting that CC fluctuates initially for most of the groups which indicates that the group is constantly restructuring itself with new users joining the group and innovating. But some of these groups (Open Biomedical Ontologies, OBI, Disease Ontology), stabilize their CC values without much fluctuation which means that there is no more restructuring



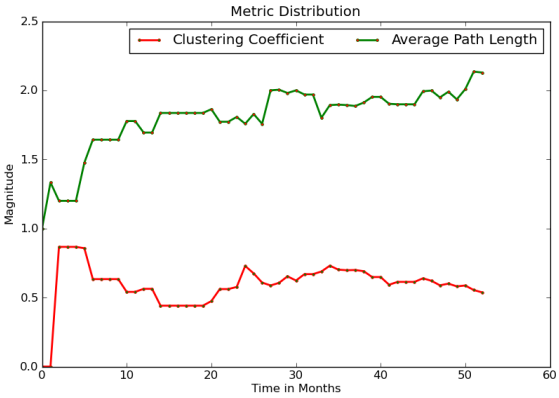
(a) Sequence Ontology (72703)



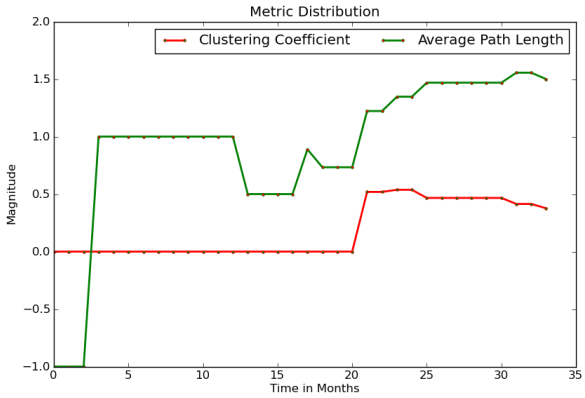
(b) Open Biomedical Ontologies (76834)



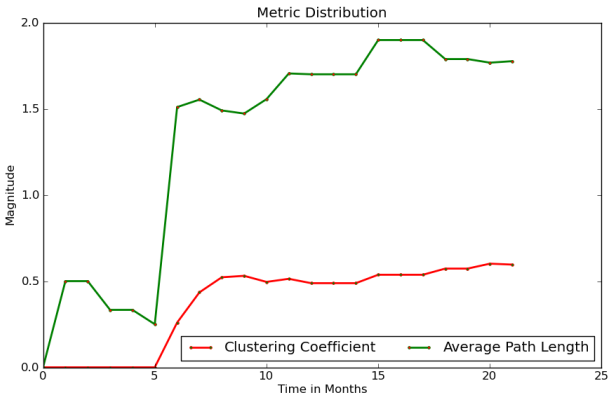
(c) Disease Ontology (79168)



(d) ChEBI (125463)



(e) Systems Biology Ontology (174625)



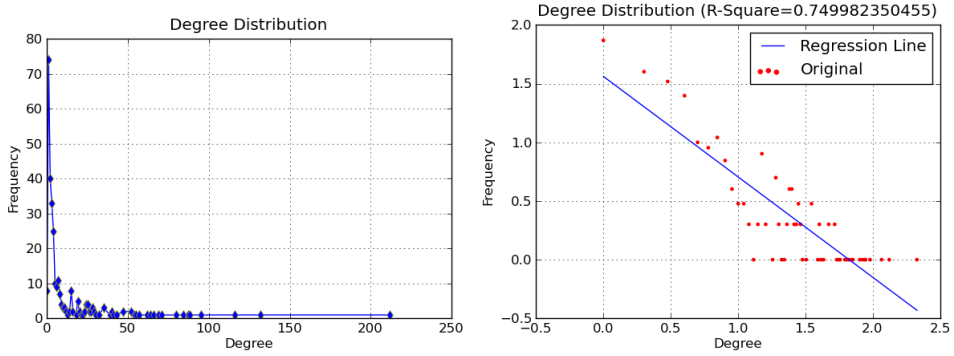
(f) OBI (177891)

Figure 6.7: Clustering Coefficient and Average Path Length Monthly Distribution

in the project and no more innovation. Other groups (Sequence Ontology, ChEBI, Systems Biology Ontology) show slight variation in CC, but not enough for innovation and creativity.

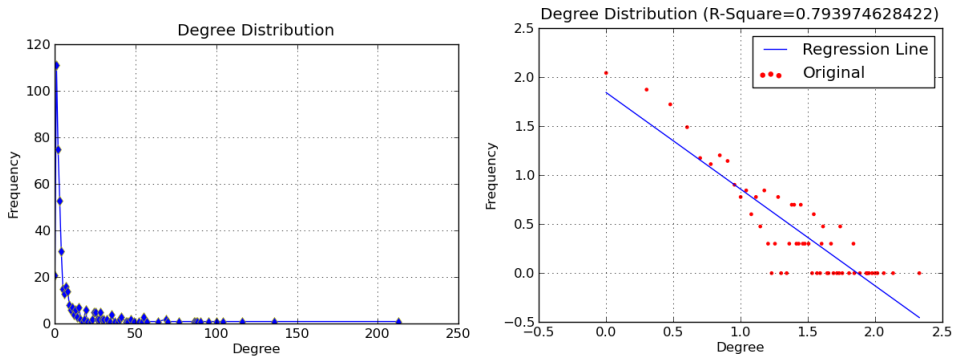
6.1.3 Degree Distribution

Figure 6.8 and 6.9 show the degree distributions for the User-User network of Gene Ontology and entire OBO community respectively. The line plot in both cases indicates that we have a power law which is shown by the log plot next to it. Power law has different interpretations in different types of graphs. In the User-User network power law indicates that only few users have a high degree where as most of the users have low degree. As can be see from figures 6.8 and 6.9, the distributions are progressively broadening in time developing heavy tails. This implies that the distribution has high variance i.e. if we randomly pick a user then he is likely to have a degree value which is far from average. In User-Artifact-User network we focus on the degree distributions of the artifacts and we get power law as shown in figure 6.10. This indicates that there are only few artifacts that attract large number of users but majority of the artifacts does not impact users of the network. Power law demonstrates the scale-free property of the network which makes the network robust and resilient; if we randomly remove nodes from the network, it does not fail. This is one of the reasons why the self organized communities like WWW (world wide web) flourish even though the members of the community join and leave voluntarily. We ignore the actors having zero degree by classifying them as outliers, as log (0) is not defined.



(a) Line Plot (b) Log Plot

Figure 6.8: Degree Distribution of User-User network for Gene Ontology (Group Id-36855)



(a) Line Plot (b) Log Plot

Figure 6.9: Degree Distribution of User-User network for OBO (All Groups Included)

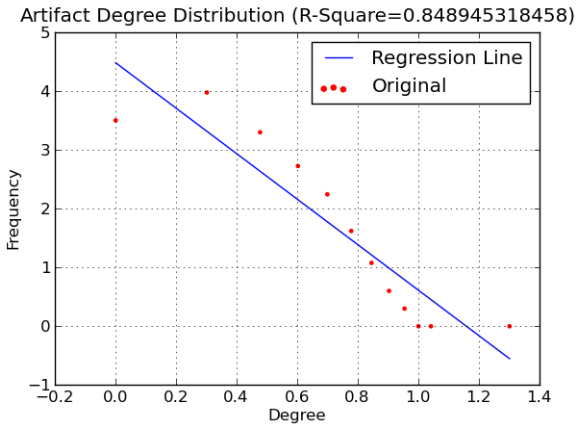
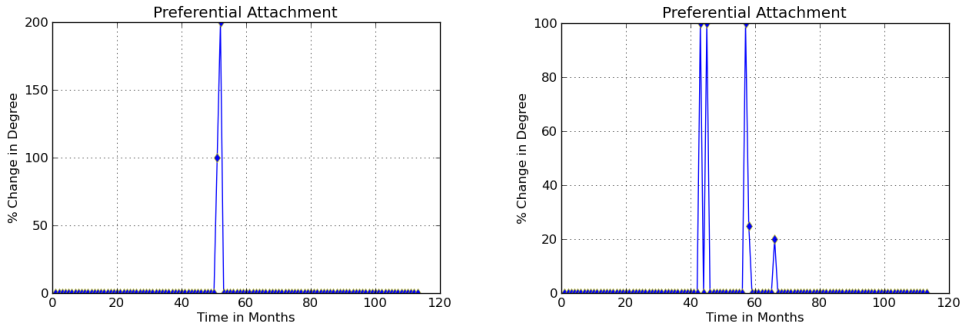


Figure 6.10: Artifact Degree Distribution of User-Artifact-User network for OBO (Comprehensive)

6.1.4 Preferential Attachment

According to [5], the reasons of having scale-free power law distribution for many large networks are: (i) networks expand continuously due to addition of new nodes to the network, and (ii) new nodes attach preferentially to the nodes that are well connected. This indicates that the nodes that are well connected will attract new nodes, and continue to grow until a certain limit [31]. In order to visualize preferential attachment, we plot the change in degree of the actor over period of time. Actor can be a user (in User-User network), artifact (in the User-Artifact-User network) or domain (in Domain-Domain network). Ideally, what we can expect from the visualization for the actor who displays preferential attachment is a linear

rise in the rate of increase in degree of the actor indicating that the actor is becoming more and more connected; but later the change will fall and approach zero which means that the actor's degree becomes saturated there after. In User-Artifact-User network, this exactly resembles the life cycle of any artifact as shown in figure 6.11(a). Initially when the artifact is new it influences response from the users; but later the responses diminish as there is no novelty left in the artifact. But there is a exception to this revelation which is shown in figure 6.11(b). Here the artifact just after its submission receives response which leads to its degree change going to 100%, and over next few months the artifact becomes dormant as there are no users contributing towards the artifact; but there is a sudden increase in the degree of the artifact after that. We tracked this artifact at the database level and found that the reason for the increase in the degree of the artifact is a certain contribution made by a user, which influenced a series of responses from the existing as well as new users. There is a possibility that this contribution was a novel one or an important addition to the existing artifact. Thus, the contribution could have been either a radical innovation or an incremental one.

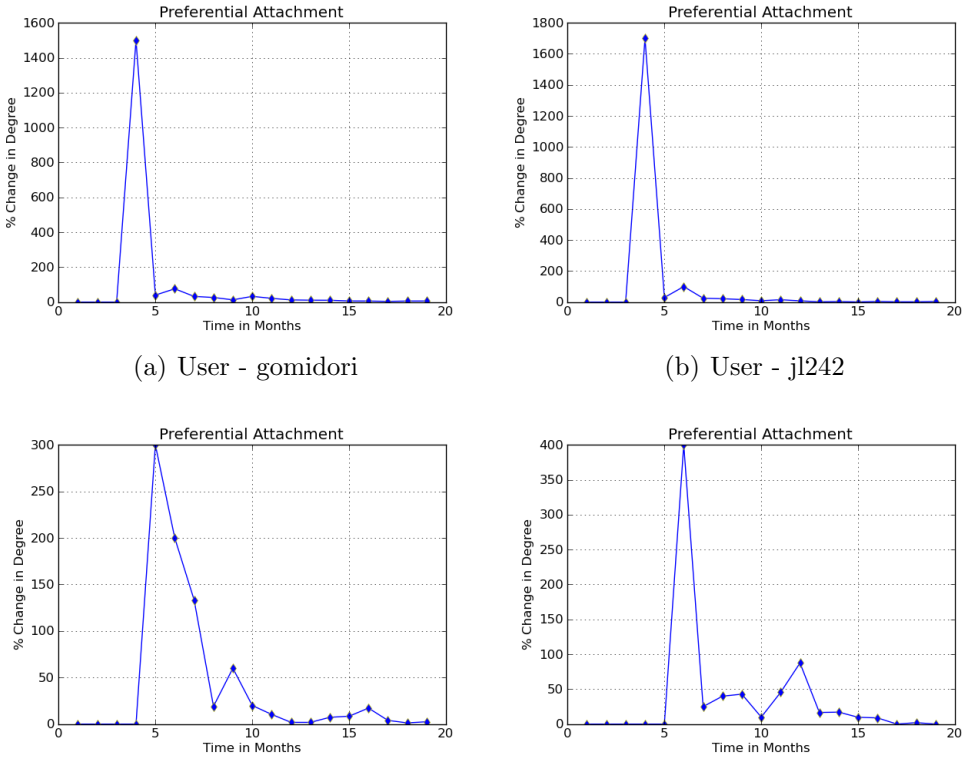


(a) Artifact - 1167822 (b) Artifact - 994121

Figure 6.11: Preferential Attachment Graph for Artifact

In case of users, the preferential attachment depicts the journey of the user from the periphery to the core of the network. Figure 6.12 shows the degree change plots for some of the users of the Gene Ontology project. It is evident from these plots that they all follow the same pattern in terms of degree change. Initially when users join the community they

are highly active as indicated by their sharp rise in the rate of increase in degree. But over period of time their degree starts increasing at a lesser rate and eventually stays constant. Based on the plots, we can say that as the user becomes more and more central, the rate of increase in the degree declines. Thus for a user, preferential attachment only exists initially for a certain time.



(a) User - gomidori

(b) User - jl242

(c) User - val_wood

(d) User - ramab

Figure 6.12: Preferential Attachment Graph for Users

6.2 Collaboration Analysis

Collaboration Analysis focuses on the collaboration aspect of the users. When an artifact is submitted, the users discuss the artifact by commenting on it. The influentiality of the artifact becomes evident from the number of comments it gets in the form of responses. But this analysis is targeted towards determining the influentiality of the user and not the artifact. Another objective to successfully visualize the collaboration between users. Activity strength

helps identify the productivity of the user by considering both contribution and collaboration aspects associated with the user, while collaboration map helps visualize the collaboration between users at any given point in time.

6.2.1 Activity Strength

This metric has been drawn from the study conducted on the impact of Co-Authorship teams [20], where it was used to identify the productivity and influentiality of the authors. In OBO the users submit the artifacts, and these artifacts are elaborated by other users in the form of comments. Thus, the artifacts become the means by which the users can effectively collaborate. In order to calculate this metric for a user, we need both the number of artifacts submitted (A) and the collaboration intensity (CI) for the user.

$$CI(i) = \sum_{j} w_{ij} \tag{6.1}$$

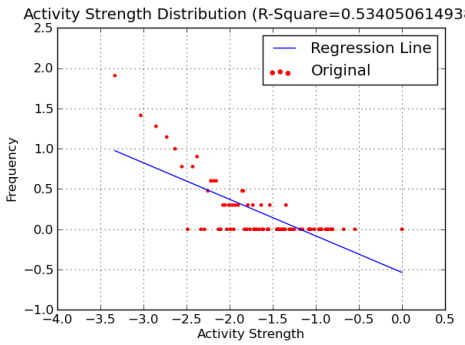
Equation 6.1 represents the Collaboration Intensity and it takes into account all the users j connected to user i. Connection between user i and j has a weight associated with it that is represented by w_{ij} where
$w_{ij} = \frac{\sum_a N_c}{N_a}$
takes the weight between user i and j over all artifacts. N_c represents the number of collaborations that take place between user i and j over artifact a and N_a is total number of artifacts over which user i and j collaborate.

$$S_a(i) = W_a\,(A_i) + W_{ci}\,(CI_i) \tag{6.2}$$

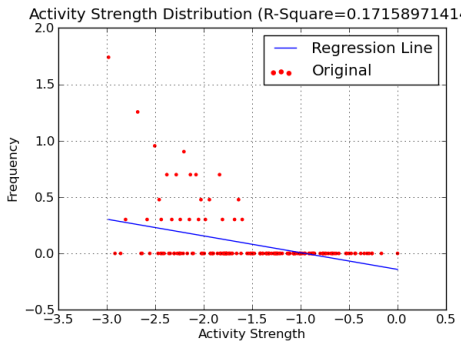
Equation 6.2 represents the activity strength with W_a representing the weight for artifact submission (A), while W_{ci} represents the weight for collaboration intensity (CI) such that $W_a + W_{ci} = 1$. If $W_a > W_{ci}$ then it indicates that the Activity Strength gives higher weight to artifact submissions than collaboration intensity and vice versa. Furthermore, A_i and CI_i

are normalized by dividing them with maximum value for A and CI respectively, such that $0 \leq A_i, CI_i \leq 1$. This also makes sure that $0 \leq S_a(i) \leq 1$.

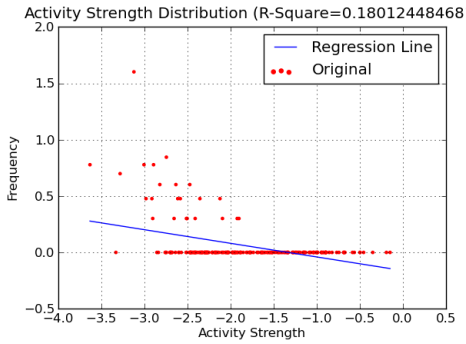
Activity Strength is used to assess the productivity, influentiality and innovation potential of the user. We expect to get a scale free distribution for this metric just like degree distribution. It means that there exist a few influential users around which the community is built, and all the other users connect to these influential users. Figure 6.13 shows activity strength plots that we get from the SourceForge data. If we put W_{ci} i.e. the weight for collaboration intensity (CI) as 0 in the equation 6.2 there by completely ignoring collaboration factor, then we get the plot as shown in figure 6.13(a). If the artifact submission factor is completely ignored in equation 6.2 by putting W_a to 0, then we get the plot shown in figure 6.13(b). With equal weights given to both these factors we get the plot shown in figure 6.13(c). As we can see, plot 6.13(a) displays a power law; whereas, plots 6.13(b)



(a) $W_{ci} = 0$



(b) $W_a = 0$



(c) $W_a = 0.5$, $W_{ci} = 0.5$

Figure 6.13: Activity Strength Log Plots for Gene Ontology (36855)

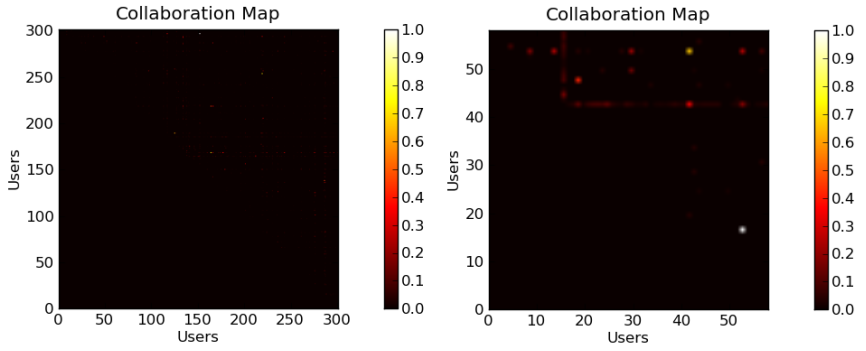
and 6.13(c) are scattered and do not indicate any specific pattern. The conclusion we can derive from these plots is that as far as artifact submission is concerned there are a few core users who make contributions towards the project 36855 (Gene Ontology). But these users are not major collaborators which is the reason why we fail to get a power law when we consider both contribution and collaboration. Most of the members of the community are collaborators and their collaboration intensity is along same lines which is the reason why we do not get a power law when we consider just the collaborations.

6.2.2 Collaboration Map

Collaboration map is used to visualize the collaboration patterns in a group or domain, using python matplotlib color map. The map is laid out as a 2 dimensional matrix with the users on both X and Y axis of the map. Each cell of the matrix represents the collaboration between the user on X axis and the corresponding user on Y axis. Diagonal cells are ignored as they represent the same user on both X and Y axis. The map is symmetrical with diagonal acting as the axis of symmetry. Figure 6.14 shows the collaboration patterns between the users of different groups in OBO. Color scheme used in the matrix is shown in a color bar adjacent to the color map. The darker the cell is, the lesser the collaboration intensity between the users associated with the cell. As the color approaches yellow or white, that indicates increase in the collaboration between associated users on X and Y axis. Lower half of the color map is colored black in order to indicate that the graph we are using is undirectional and is symmetrical across the diagonal. Thus, the collaboration between *User-X* and *User-Y* is same as the collaboration between *User-Y* and *User-X*. In case of bi-directional graphs entire color map can be used. Collaboration between two users is calculated using the formula given in equation 6.3 shown below.

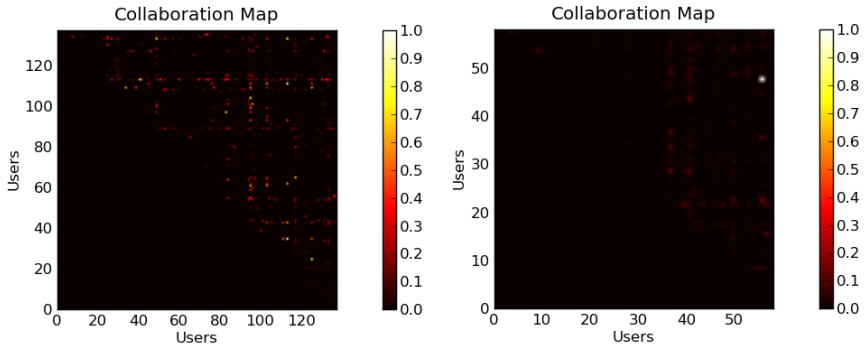
$$C_{User-X,User-Y} = \frac{N_c}{N_a} \tag{6.3}$$

In equation 6.3, N_c is the number of collaborations that took place between *User-X* and *User-Y*; while N_a is the number of artifacts over which the collaborations took place between *User-X* and *User-Y*. Collaboration is normalized using the maximum value. Thus, we get collaboration value between 0 and 1.



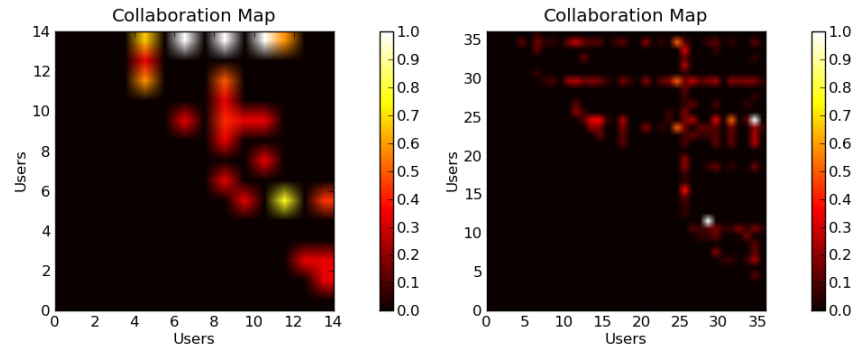
(a) Gene Ontology (36855)

(b) Sequence Ontology (72703)



(c) Open Biomedical Ontologies (76834)

(d) ChEBI (125463)



(e) Systems Biology Ontology (174625)

(f) OBI (177891)

Figure 6.14: User Collaboration Maps for various projects under OBO

6.3 Activity Analysis

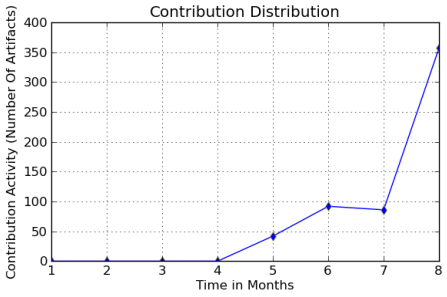
The activity patterns are seen against the time so that it reflects different stages the community has gone through. Activity not only shows the project life cycle stages but also explains the innovation taking place in the project. We define two types of activities: Artifact Submission and Active User Distribution. These activities can be viewed using SciBrowser for a certain group (project) or domain from a certain start time (year/month) to a certain end time (year/month). The smallest unit of time is one month i.e., by default the results will be shown as monthly distributions. Also, the time can be aggregated to see the total activity for that period; for e.g., a monthly activity can be aggregated to view it every n months, where n can be anything in the set [2, 3, 6, 9, 12]. For the plots in figures 6.15, 6.16, 6.17 and 6.18 n is set to 9.

The activity metrics indicate the stage of the community growth at any given point in time. A typical project life cycle of an organization is based on the sales or profits (dependent variable) over time. According to [10] for an open source software project it is the number of downloads the users do, that decides life cycle of the project. OBO being an open science project, we plot the number of contributions and the number of active members over time, in order to see how the project fits into the organizational life cycle model.

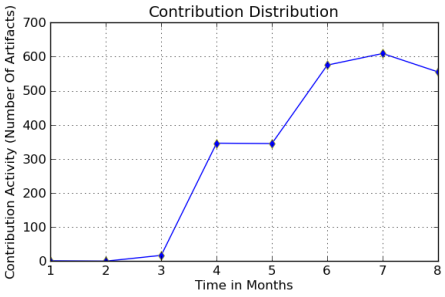
6.3.1 Contribution Distribution

"Contribution Distribution" is the plot of the number of artifacts submitted over period of time. This metric can be visualized using line or bar plot, with the time on the x axis and the magnitude of submission on y axis. Figure 6.15 shows the contribution pattern for different communities under OBO foundry. Each point in the graph accounts for the aggregate contribution of 9 months. This is done in order to achieve a smoother curve and eliminate noise. The figure shows all the stages of typical project cycle. Figure 6.15(a) shows the "Introduction and Growth" phase for the community. Figure 6.15(b) shows the community which is in its "Maturity" phase and figure 6.15(c) shows the "Decline" phase

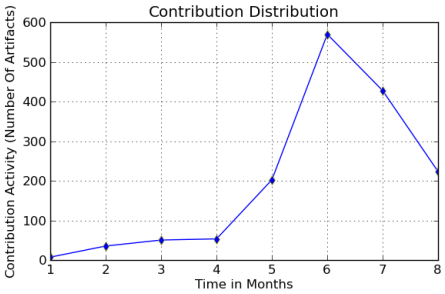
for the respective community. It is not necessary that all the projects follow the same cycle that is mentioned above. It is the most typically observed life cycle of a project. Figure



(a) Introduction & Growth



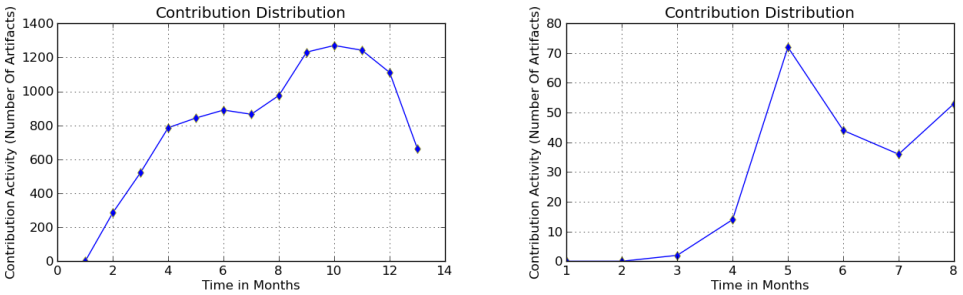
(b) Maturity



(c) Decline

Figure 6.15: Typical Contribution Activity Patterns across projects OBI (177891), Open Biomedical Ontologies (76834) and ChEBI (125463)

6.16 below shows the alternate life cycles that are followed by the projects. It is seen from figure 6.16(a), that the community starts of with a steady growth and appears to become mature after that, but then there is a sudden rise in the contributions coming from the members of the community. Later the number of contributions starts dropping. According to the project life cycle model [10], community can either start declining or reviving after it reaches its maturity. In this case we witness a revival which can be due to an important breakthrough in the existing domain in the project or due to the introduction of a new domain. It can also be simply due to a new discovery by a group of motivated researchers. Revival can make a project enter the growth phase again; this trend is evident from figure 6.16(b), where the community starts reviving after it started to decline. Thus, revival brings with it the innovation which tends to put the project back on track.



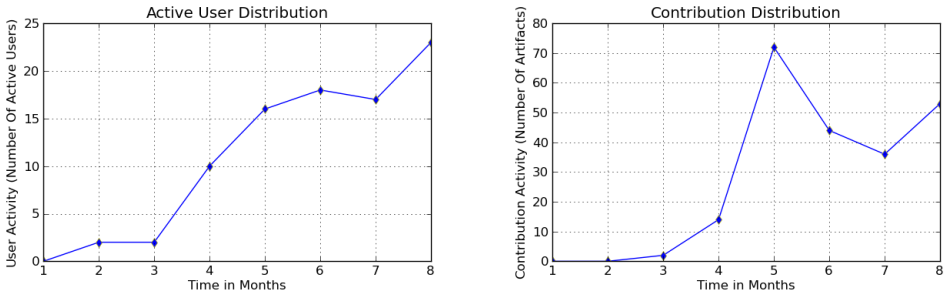
(a) Revival After Maturity (b) Revival After Decline

Figure 6.16: Alternate Contribution Activity Patterns across projects Gene Ontology (36855) and Sequence Ontology (72703)

6.3.2 Active User Distribution

We define active users as the users that contribute towards the community. "Active User Distribution" is the distribution of the active users over the period of time. The plot does not necessarily indicate the state of the project, but it gives us the idea about overall active population in a project at any given point in time. If we compare the plots of distribution of active user with the plot of distribution of artifacts contributed, we get an idea about how the changes in the active user concentration have affected the changes in the artifact submission. Figure 6.17 and figure 6.18 compares these two plots for two projects. As can be seen from figure 6.17(a) an initial increase in the number of active users can be correlated to increase in the artifact submission (from x=1 to x=5 in figure 6.17(b)). This indicates that the increase in the active population leads to influx of new ideas in the project and hence innovation. Further, as the growth in the number of active users stagnates (from x=5 to x=7), the artifact submission falls down and is revived as the active users grow (from x=7 to x=8).

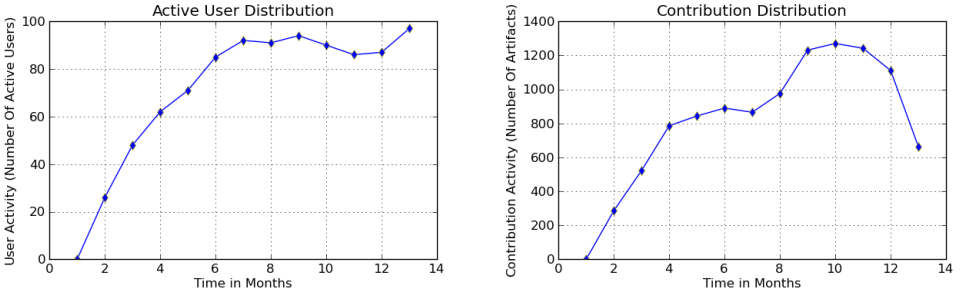
The above case might not happen at all the time; figure 6.18 shows an exception to the above revelation. Initially as the number of active users increase (from x=1 to x=4) , an increase in the number of artifacts can be seen in figure 6.18(b). But the growth in the number of artifacts stagnates (from x=4 to x=7) while the number of active users still increases indicating that although the number of active users increase, their contribution



(a) Active User Distribution (b) Contribution Distribution

Figure 6.17: Comparisons between Contribution and Active User Distribution Patterns for Sequence Ontology (72703)

is not enough for innovation. Later, as the number of users reach a saturation (in figure 6.18(a) from x=7 to x=11), there is a sharp increase in the number of contributions. This indicates that there was some significant contribution during this time which led to a high activity coming out of the project, although there was no increase in the number of active users at all and this again is a sign of innovation. Thus, in some projects only a few users are the active contributors while the others are dormant or inactive, yet there is a significant activity going on in the project. It also shows that it is the quality of the artifact which decides what activity will follow the current one; not necessarily the active population.



(a) Active User Distribution (b) Contribution Distribution

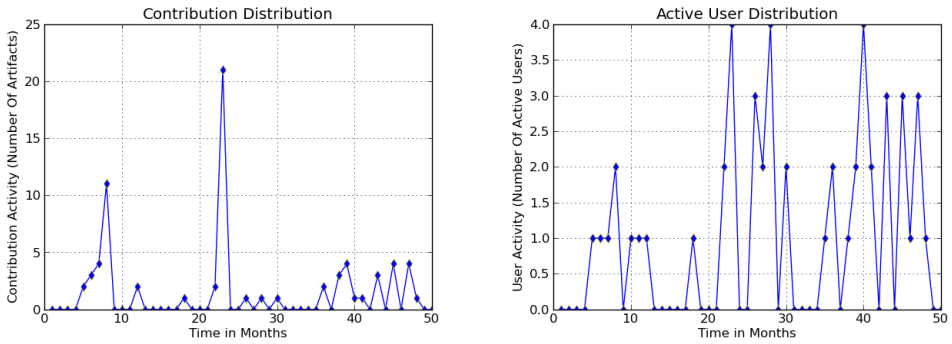
Figure 6.18: Comparisons between Contribution and Active User Distribution Patterns for Gene Ontology (36855)

6.3.3 Activity Outburst Frequency Distribution

The activity for any project can be one of the above two types the number of contributions over time and the number of active users over time. It can be seen from figure 6.19 that the activity of a project does not remain constant or ever increasing over time but goes though ups and downs. Outburst in an activity is defined as the activity that crosses a certain threshold. This threshold is defined by equation 6.4 given below.

$$ActivityOutburst_{Threshold} = Average_{Activity}(1 + \sigma) \tag{6.4}$$

where σ is the user defined variable. The value of σ has to be chosen carefully. Choosing too small value for σ can greatly increase the number of outbursts; whereas, choosing too large value for σ can greatly reduce the number of outbursts. So it is advisable to choose the value of σ based on the average value of the activity. "Activity Outburst Frequency Distribution"

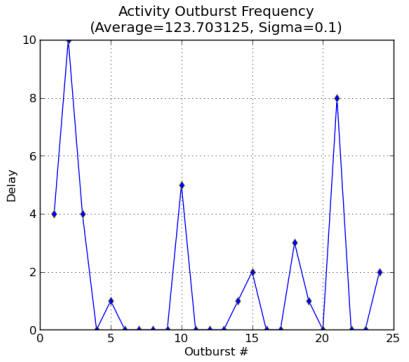


(a) Distribution of Number of Artifacts Submitted (b) Distribution of Number Active Users

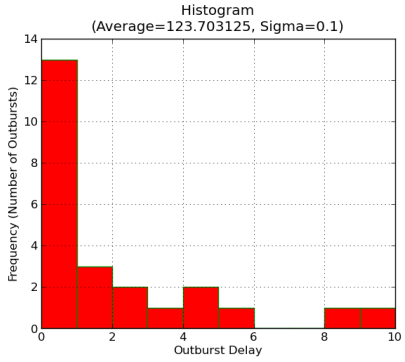
Figure 6.19: Activity Plots for Systems Biology Ontology (174625)

for any activity can be defined as the frequency of occurrence of the outbursts in the activity. It is a plot with *x-axis* indicating the outburst number; whereas, *y-axis* indicating the delay in the occurrence of the corresponding outburst on *x-axis*. This metric has been derived from the agent based civil violence model [11] created by Joshua M. Epstein. Figures 6.20 and 6.21 show "Activity Outburst Frequency Distribution" on the left and its corresponding

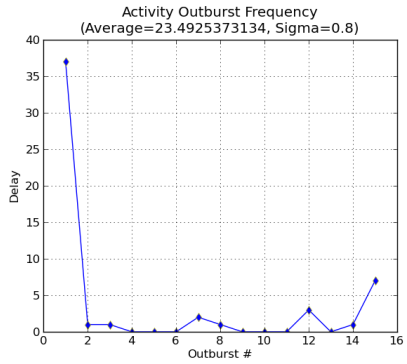
histogram on the right, for different projects under obo. The histogram groups the outbursts according to the delay caused for the outbursts. The plots also show the average value of



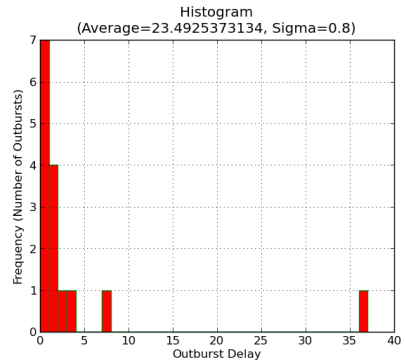
(a) Outburst Frequency Distribution for Gene Ontology



(b) Outburst Frequency Histogram for Gene Ontology



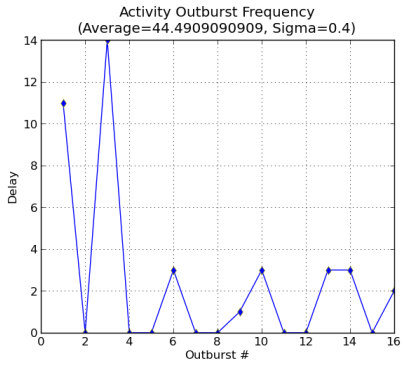
(c) Outburst Frequency Distribution for ChEBI



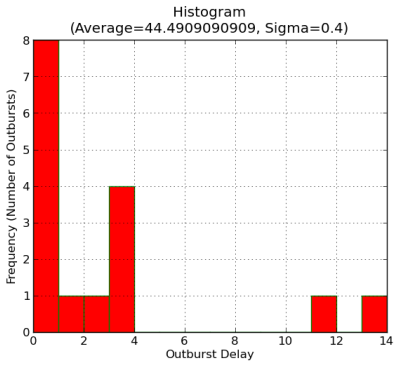
(d) Outburst Frequency Histogram for ChEBI

Figure 6.20: Activity Outburst Frequency Distribution For Gene Ontology and ChEBI

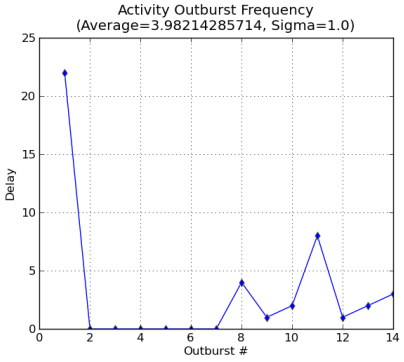
the activity and the value of σ chosen. Value of σ is selected, based on the average value of the activity and by considering the number of outbursts we get. It can be seen that a specific pattern comes across from these plots. According to this pattern, every activity outburst that occurs after a significant delay is followed by a series of quick outbursts. These outbursts might then be followed by an outburst that occurs after a significant delay. This means that a high activity seems to trigger a series of high activities that are probably related to the high activity with significant delay. Thus, in most of the histograms it can be seen that the majority of the outbursts occur frequently i.e. they have a small delay period.



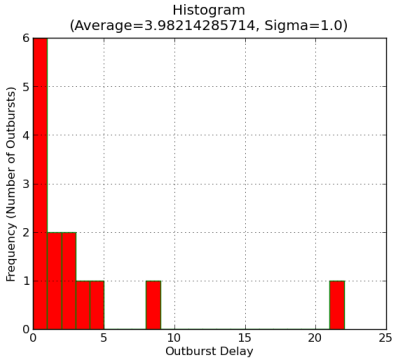
(a) Outburst Frequency Distribution for Open Biomedical Ontologies



(b) Outburst Frequency Histogram for Open Biomedical Ontologies



(c) Outburst Frequency Distribution for Sequence Ontology



(d) Outburst Frequency Histogram for Sequence Ontology

Figure 6.21: Activity Outburst Frequency Distribution For Open Biomedical Ontologies and Sequence Ontology

Chapter 7

Conclusion

In this thesis, we introduced SciBrowser, which is a computational ethnography tool, to explore open source science communities that reside in SourceForge. We demonstrate the applicability of the SciBrowser to the analysis of Open Biomedical Ontology (OBO), which is an open source science network in the field of biomedical science. To demonstrate the utility of SciBrowser and apply it the analysis of open source science networks, we present a three dimensional analysis approach: structural, collaboration and activity analysis

Under structural analysis, we examine traditional social network metrics such as centrality and density. We observe high values of centrality for the Sequence Ontology (72703) project, indicating that it has a structural topology resembling the star network. This suggests the possibility of the presence of core-periphery pattern. Clustering Coefficients (CC) and Average Path Lengths (APL) measures are also plotted over time in order to determine the presence of small world property in different projects. Values of CC for most of the projects are around 0.5. This observation suggests that there exist highly connected components which are loosely coupled with each other. For most of the networks, the value of average path length is around 2, which is small compared to a random network. This facilitates efficient knowledge transfer and innovation diffusion from one part of the network to the other. Furthermore, most of the projects that we examined stop substantially restructuring themselves eventually, as indicated by stabilized values of CC. Due to the lack of restructuring, it becomes evident that the communities may experience challenges in innovation. The SciBrowser tool also plots degree distributions and visualizes preferential attachment. A power law degree distribution is observed for the User-User graph of the Gene Ontology (group_id: 36855) domain, indicating the resilient nature of the community.

Novel metrics such as "Activity Strength" are introduced for collaboration analysis. These metrics are used to measure the productivity and innovation potential of users. Confining innovation to artifact submission generates a power law, indicating the presence of core members submitting most of the artifacts, while peripheral members commenting on them. When the artifact submission and collaboration factors are combined as a proxy metric for innovation, the power law is disturbed and seizes to exist. This may indicate that major contributors of the project are weak collaborators and their strong contribution factor is nullified by their weak collaboration intensity. We also visualize the collaboration among users of the community using a color coded map. The visualization indicates that only few users pair effectively collaborate, while most of the other user pairs have mediocre amount of collaboration between them.

Under activity analysis, the tool plots two types of activities: artifact submissions and active user distributions, over a period of time. It is observed that open source science communities examined in this study closely follow the organization project life cycle. Thus, there is a possibility that open source science projects possess specific organizational characteristics such as division of labor, leadership, level of commitment, and coordination/control. In addition, we also discovered certain unconventional activity patterns, in which the project picks up pace after the decline phase. Knowing which stage the project is in can provide potential insight to the administrators of a project, so that they can take certain decisions at the proper time to revive the project. Active user distribution is also discussed in association with artifact submission pattern. It is observed that when the number of users in the project increase, it leads to an increase in the artifact submission. The rational behind this can be the innovation which the new users bring into the project. We observe that an activity outburst occurring after a significant delay is usually followed by one or more, frequently occurring outbursts. This implies that the first outburst (occurring after a significant delay) might trigger one or more outbursts that follow it. Or, it might also imply that initially

when the community is growing, there are less outbursts, but once the community has found its direction and conflicts are resolved the outbursts occur more frequently.

Primarily the SciBrowser tool is used by the simulation team in Simulation and Modeling lab at Auburn University to validate their agent-based simulation models. But in a broader sense, the tool is targeted towards researchers that explore open source communities in SourceForge. Although our study pertains to specific community on the Sourceforge, the application of the SciBrowser is not limited to the open biomedical ontology (OBO) community. The tool is versatile in terms of its usefulness, as it can also be applied to open source software projects. At an abstract level, the structure of Sourceforge communities is similar, and the database schema used by all the projects is the same. These similarities make SciBrowser ideal for those who are interested in analyzing the collaboration between the users of a community and tracking the activity taking place within a project that resides in SourceForge.

Our future plans pertaining to the SciBrowser involve re-engineering the tool toward a comprehensive analysis tool, including options for network visualization, metric observation, and plot generation. Currently, we can visualize metrics and generate plots, but network visualization in the form of a graph with nodes and edges is lacking. Such a feature would give researchers the ability to observe the structural growth of a community and help develop hypotheses about its dynamics. Also, further work includes integrating data mining features, allowing the development of social network specific mining algorithms. Data mining techniques such as association rule mining can be used to establish association or relation between the changes in the structural metrics as well as temporal activity patterns. The current version of the tool lacks the feature that would explicitly link the structural attributes such as change in degree to the innovation metrics at the user level. Tool accounts for the activity at the group level and domain level, but not yet at the user level.

Bibliography

[1] F. Colaiori L. S. Buriol D. Donato S. Leonardi A. Capocci, V. D. P. Servedio and G. Caldarelli. Preferential attachment in the growth of social networks: The internet encyclopedia wikipedia. September 2006.

[2] Piotr Fronczak Agata Fronczak and Janusz A. Hoyst. Average path length in random networks. November 2004.

[3] Pieter Swart Aric Hagberg, Dan Schult. Networkx, 2010.

[4] Yaneer Bar-Yam. Dynamics of Complex Systems. Westview Press, 1997.

[5] Albert-Laszlo Barabasi* and Reka Albert. Emergence of scaling in random networks. Science, 286:509–512, October 1999.

[6] Open Biological and Biomedical Ontology Foundry. Obo:foundry, July 2010.

[7] Scipy Community. Numpy, 2008.

[8] Paul A. David and Michael J. Spence. Towards institutional infrastructures for e-science: The scope of the challenge. Oxford Internet Institute, Research Report No. 2, September 2003.

[9] Charles Dhanaraj and Arvind Parkhe. Orchestrating innovation networks. Academy of Management Review, 31(3):659–669, July 2006.

[10] Jr. Donald E. Wynn. Organizational structure of open source projects: A life cycle approach. Proceedings of 7th Annual Conference of the Southern Association for Information Systems, pages 285 – 290, 2003.

[11] Joshua M. Epstein. Modeling civil violence: An agent-based computational approach. Proceedings of the National Academy of Sciences of the United States of America, 99:7243 – 7250, May 2002.

[12] Ralph Johnson John Vlissides Erich Gamma, Richard Helm. Design Patterns. Elements of Reusable Object-Oriented Software.

[13] James A. Hendler Qingpeng Zhang Zhuo Feng Yanqing Gao Hui Wang Fei-Yue Wang, Daniel Zeng and Guanpi Lai. A study of the human flesh search engine: Crowd-powered expansion of online knowledge.

[14] Source Forge. Sourceforge.net research data, 2010.

[15] Ian Foster. Service-oriented science. Science, 308(5723):814 – 817, May 2005.

[16] John H. Holland. Hidden Order: How Adaptation Builds Complexity. Addison-Wesley Publishing Company Inc., 1995.

[17] The MathWorks Inc. Matlab, 2010.

[18] Scott Christley Gregory Madey Jin Xu, Yongqin Gao. A topological analysis of the open source software development community. 2005.

[19] Michael Droettboom John Hunter, Darren Dale. Matplotlib, 2008.

[20] Weimao Ke Katy Borner, Luca Dall'Asta and Alessandro Vespignani. Studying the emerging global brain:analyzing and visualizing the impact of co-authorship teams. Wiley Periodicals - Complexity, 10(4):57–67, 2005.

[21] Glenn E. Krasner and Stephen T. Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. 1988.

[22] Robert C. Martin. The dependency inversion principle. May 1996.

[23] Robert C. Martin. The open-closed principle. January 1996.

[24] Ken McIvor. wxmpl, 2009.

[25] Damodar Shenviwagle Michael Arnold and Levent Yilmaz. Scibrowser: A computational ethnography tool to explore open source science communities. March 2010.

[26] John H. Miller and Scott E. Page. The standing ovation problem. April 2004.

[27] Melanie Mitchell. Complexity: A Guided Tour. Oxford Univ Press, 2009.

[28] Susan A. Mohrman and Caroline S. Wagner. The dynamics of knowledge creation: Phase one assessment of the role and contribution of the department of energy's nanoscale science research centers. November 2008.

[29] Bernard Munos. Can open-source r&d reinvigorate drug research? Nature Reviews Drug Discovery, August 2006.

[30] Siobhan Omahony and Fabrizio Ferraro. The emergence of governance in an open source community. April 2007.

[31] Jill E. Perry-Smith and Christina E. Shalley. The social side of creativity : A static and dynamic social network perspective. Academy of Management Review, 28(1):89–106, January 2003.

[32] Noel Rappin and Robin Dunn. wxPython in Action. Manning Publication Co., 2006.

[33] SRDA. Sourceforge.net research data, September 2008.

[34] Georgia Tech Susan Cozzens and NSF Julia Lane. A deeper look at the visualization of scientific discovery in the federal context. September 2008.

[35] Brian Uzzi and Jarrett Spiro. Collaboration and creativity: The small world problem. American Journal of Sociology, 111(2):447 – 504, September 2005.

[36] S. Wasserman and K. Faust. Social network analysis: Methods and applications. Cambridge Univ Pr, 1994.

[37] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. Nature, 393:440–442, June 1998.

[38] Jia Zhang Wei Tan and Ian Foster. Network analysis of scientific workflows: A gateway to reuse.

[39] Wikipedia. Centrality, April 2010.

[40] Levent Yilmaz and Tuncer Oren. Agent-Directed Simulation And Systems Engineering. Wiley-VCH, 2009.