MINING AND MODELING THE OPEN SOURCE SOFTWARE COMMUNITY

JIN XU
Ph.D Defense
PRESENTATION OUTLINE

- Research Motivation
- Data Collection Methodologies
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
RESEARCH MOTIVATION

- What is Open Source Software?
  - Computer Software
  - Freely access, modify and redistribute
- OSS developers
  - Distributed/decentralized
  - Volunteers
  - Users participate in OSS development
- Examples of OSS
  - Linux, Apache, Sendmail, Perl, PHP, R, Python, Java
THE STATE OF ART

- Computer software engineering
- Success factors (Cronstone (2003), et al.)
  - Development process, work practice (Scacchi (2004), et al.)
- Economics
  - Innovation process (Von Hippel (2005), et al)
- Psychology
  - Motivations of individuals (Hars (2001), Bitzer (2005), etc.)
  - Motivations of organizations (Rossi (2005), et al)
- Organizational science
  - Modes of organization and performance (Dalle (2005), et al)
LIMITATIONS OF OSS STUDIES

- Qualitative studies
  - Collecting data is difficult
- Incomplete and inaccurate data
  - Samples collected using questionnaires
- Lack of studies on relationships among projects and developers
- Lack of simulation models
  - Missing historical data
  - Future prediction
OBJECTIVES

- Collect and mining OSS community data
  - Identify data resources
  - Build mining tools
- Study OSS communities
  - Projects
  - Developers
  - Relationships
- Build and validate simulation models
  - Developer activities
  - Project life cycles
PRESENTATION OUTLINE

- Research Motivation
- **Data Collection Methodologies**
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
OSS WEB REPOSITORIES

- OSS development — distributed and decentralized
  - Conflict of individual interests and project goals
  - Easy join and leave
  - Synchronization

- Web repositories
  - Version control system
  - Bug report and fix system
  - Discussion tools

- Two types of OSS web repositories
  - Individual vs. group
DATA SOURCE

- SourceForge.net
  - Over 140,000 projects and 1.5 million registered users
  - Project web servers, trackers, mailing lists, discussion boards, software management tools

- Data information
  - Classification, list of developers, statistics, forums, trackers
  - User id, name, participated projects

- Data limitations
  - Missing data, outliers, anonymous data
MINING PROCESS

- crawler
- web documents
- database
- Extraction
- raw data
- cleaning
- preprocessing
- Statistics
  - Mathematics
  - Data mining
- cleaned data
- report
- analysis
- summary

Ph.D defense, CSE Dept. Univ. of Notre Dame
WEB CRAWLER

URL accessing

Parser

Word extractor

Table extractor

Link extractor

Link exists?

database

yes

no

End

no

yes
PRESENTATION OUTLINE

- Research Motivation
- Data Collection Methodologies
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
SOCIAL NETWORK ANALYSIS

- Complex and Self-organizing System
  - Large numbers of locally interacting elements
- Dynamic Social Network
- Main components — developers and projects
  - Many developers participate on one project
  - A developer participates on more than one project
- Study of developer roles and their activities
OSS COMMUNITY

- **User Group**
  - Passive Users: no direct contribution
  - Active Users: report bugs, suggest features, etc.

- **Developer Group**
  - Peripheral Developers: *irregularly* contribute
  - Central Developers: *regularly* contribute
  - Core Developers: *extensively* contribute, manage CVS releases and coordinate peripheral developers and central developers.
  - Project Leaders: guide the vision and direction of the project.
OSS DEVELOPMENT COMMUNITY

- Project Leaders
- Core Developers
- Co-developers
- Active Users
OSS SOCIAL NETWORK

- Two Entities: developer & project
- Project-Developer Network (bipartite graph)
  - Node: project, developer
  - Edge: developers in a project are connected to that project
PROJECT-DEVELOPER NETWORK

Ph.D defense, CSE Dept. Univ. of Notre Dame
OSS SOCIAL NETWORK (cont.)

- Developer Network (unipartite graph)
  - Node: developer
  - Edge: developers in a project are connected to each other
  - Impact of developers

- Project Network (unipartite graph)
  - Node: project
  - Edge: two projects are connected if there is a developer participates on both.
  - Impact of projects
OSS DEVELOPER NETWORK

Project 7789

Project 6783

Project 9013

Project 232

Project 4264

Ph.D defense, CSE Dept. Univ. of Notre Dame
DATA COLLECTION & EXTRACTION

- Data Source: SourceForge 2003 data dump
- Project Leaders & Core Developers
  - Identification stored in data dump
- Co-developers & Active Users
  - Forum: ask & answer questions
  - Artifact: bug report, patch submission, feature request, etc.
SUBSET OF THE SCHEMA
ANALYSIS

- Project Leaders (10%)
- Core Developers (5%)
- Co-developers (12%)
- Active Users (8%)
- Passive Users (65%)

Ph.D defense, CSE Dept. Univ. of Notre Dame
# MEMBER DISTRIBUTION

<table>
<thead>
<tr>
<th>Project Size</th>
<th>Project Number</th>
<th>Project Leaders</th>
<th>Core Developers</th>
<th>Co-developers</th>
<th>Active Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 88</td>
<td>64847</td>
<td>47.8%</td>
<td>20.6%</td>
<td>19.8%</td>
<td>11.8%</td>
</tr>
<tr>
<td>&gt; 88, ≤ 279</td>
<td>193</td>
<td>2.1%</td>
<td>5.7%</td>
<td>60.3%</td>
<td>31.7%</td>
</tr>
<tr>
<td>&gt; 279</td>
<td>70</td>
<td>0.9%</td>
<td>2.7%</td>
<td>55.8%</td>
<td>40.6%</td>
</tr>
</tbody>
</table>
PRESENTATION OUTLINE

- Research Motivation
- Data Collection Methodologies
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
TOPOLOGICAL PROPERTIES

- **Degree Distribution**
  - The total number of links connected to a node
  - Relative frequency of each degree value
  - Poisson distribution vs. Power law distribution

- **Diameter**
  - The maximum longest shortest-path
  - The average longest shortest path

- **Cluster**
  - A social network consists of connected nodes

- **Clustering Coefficient**
  - The ratio of the number of links to the total possible number of links among its neighbors
DEGREE DISTRIBUTION

Degree distribution

Index Distribution of Developer Nodes
## ANALYSIS (cont.)

<table>
<thead>
<tr>
<th>Property</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>58651</td>
<td>83118</td>
<td>139507</td>
<td>161691</td>
</tr>
<tr>
<td>Z1</td>
<td>~1</td>
<td>6</td>
<td>508</td>
<td>3241</td>
</tr>
<tr>
<td>Z2</td>
<td>~1</td>
<td>17</td>
<td>13398</td>
<td>31998</td>
</tr>
<tr>
<td>Diameter</td>
<td>Inf.</td>
<td>10.2</td>
<td>2.7</td>
<td>2.7</td>
</tr>
<tr>
<td>Clustering coefficient</td>
<td>0.8406</td>
<td>0.8078</td>
<td>0.8867</td>
<td>0.8297</td>
</tr>
<tr>
<td>Largest project cluster</td>
<td>737</td>
<td>15091</td>
<td>30794</td>
<td>40175</td>
</tr>
<tr>
<td>2nd Largest project cluster</td>
<td>197</td>
<td>34</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td># of project clusters</td>
<td>43826</td>
<td>34280</td>
<td>27983</td>
<td>21659</td>
</tr>
</tbody>
</table>
SUMMARY

- Identified and quantified co-developer and active user roles
- Different community distributions for large and small projects
- Small World Phenomenon
  - Small diameter & high clustering coefficient
- Scale Free
  - Power law distribution
- Effect of Co-developers & Active Users
PRESENTATION OUTLINE

- Research Motivation
- Data Collection Methodologies
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
COMMUNITY STRUCTURE

- Communities: In some networks, some nodes are grouped together by a high density of edges, while there are few edges between those groups.

- Communities: tightly-connected groups (Newman & Girvan, 2004)

- Importance:
  - Nodes within a community — common features
  - Nodes between communities — hubs
INFLUENCE IN OSS NETWORKS

- Identify projects which might have related subjects, similar programming environment, or common developers.
- Study projects interactions during their growth.
- Get information about the communication path and knowledge flow within or between communities. Such information can help us adjust and improve the robustness of communications in OSS
HIERARCHICAL CLUSTERING

- Tree data structure
- Measured strength on each pair of vertices
  - Shortest path
  - The inverse
  - Total number of paths
- Join a pair with the strongest strength
- Weakness: inaccurate clustering (Newman & Girvan, 2004)
BETWEENNESS

- Edge betweenness

\[ C_B(v) = \sum \frac{N_{ij}(e)}{N_{ij}} \]

- The Girvan & Newman method
  - Top-down, remove edges with the highest betweenness
  - \( O(m^2n) \) time on a network with \( m \) edges and \( n \) vertices
  - \( O(n^3) \) on a sparse graph
GREEDY ALGORITHM

- Modularity Q: the fraction of edges within a community minus the expected value of the fraction of edges within a community in a random network.
- The best community structure is where Q is the largest

\[
Q = \sum_i (e_{ii} - a_i^2)
\]

\[
a_i = \frac{k_i}{\sum e_{ij}}
\]

\(e_{ij}\) is the fraction of edges that connect nodes in group \(i\) to nodes in group \(j\).
\(a_i\) is the fraction of edges that connect to node \(i\)
the total edges in the network
\(k_i\) is the number of edges connecting to group
DATA STRUCTURE

- A matrix contains $\Delta Q_{ij}$. Each row of the matrix is stored as a balanced binary tree for $O(\log(n))$ search and insertion and as a max-heap for constant time to find the largest element.
- A max heap contains the largest element of each row.
- An array contains $a_i$. 
GREEDY ALGORITHM

- The initial values

\[ \Delta Q_{ij} = \begin{cases} 
\frac{1}{\sum e_{ij}} - \frac{K_i K_j}{\sum e_{ij}^2} & \text{if } i, j \text{ connected} \\
0 & \text{otherwise} 
\end{cases} \]

\[ a_i = \frac{k_i}{\sum e_{ij}} \]

- Select \( \Delta Q_{ij} \) from the max-heap and communities are grouped

- Update values

\[ \Delta Q_{ik} = \begin{cases} 
\Delta Q_{ik} + \Delta Q_{jk} & \text{if group } k \text{ is connected to } i, j \\
\Delta Q_{ik} - 2a_j a_k & \text{if group } k \text{ is connected to } i, \text{ not } j \\
\Delta Q_{jk} - 2a_i a_k & \text{if group } k \text{ is connected to } j, \text{ not } i 
\end{cases} \]

\[ a'_j = a_j + a_i \]

- \( O(md \log(n)) \)
PROBLEM SETUP

- The largest component of the project network in Jan. 2003.
- Exclude project 1 which is the SourceForge software project because it links to 10,000 other projects.
- The project network consists of 27,834 nodes and 173,644 edges.
RESULT: THE VALUE OF Q
ANALYSIS

- Max Q = 0.2227
- 611 communities
- The largest community consists of 3467 projects
- Many small communities with size less than 10
- The 10 largest groups include 64.8% of the whole projects
- The important communication paths are those connections between communities
- Common developers are keys to transfer information between two project groups

Ph.D defense, CSE Dept. Univ. of Notre Dame
OSS ASSORTATIVE MIXING

- Several explanations exist for the community structure.
- Mutual acquaintance: two nodes with a common neighbor are more likely to link to each other.
- Homophily: two nodes with the same attributes are more likely to link to each other.
  - Measure: assortative mixing.
ASSORTATIVE MIXING

- Assortative coefficient

\[ r = \frac{\sum_i e_{ii} - \sum_i a_i b_i}{1 - \sum_i a_i b_i} \]

- \( e_{ij} \) is the fraction of edges in a network with type \( i \) to nodes of type \( j \)
- \( b_j \) is the fraction of edges of each type with nodes of type \( i \)
## ASSORTATIVE MIXING FOR OSS

<table>
<thead>
<tr>
<th>Category</th>
<th>Assortative mixing coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>0.1009</td>
</tr>
<tr>
<td>Operating System</td>
<td>0.1078</td>
</tr>
<tr>
<td>User interface</td>
<td>0.0893</td>
</tr>
<tr>
<td>Development status</td>
<td>0.0553</td>
</tr>
<tr>
<td>Programming language</td>
<td>0.1541</td>
</tr>
<tr>
<td>Intended audience</td>
<td>0.0449</td>
</tr>
</tbody>
</table>
SUMMARY

- Community structure exists among SourceForge project network
- Key communication paths are identified among groups
- Homophily exists in the OSS network
- Identify factors which are important in OSS grouping
PRESENTATION OUTLINE

- Research Motivation
- Data Collection Methodologies
- Social Network Theory
- OSS Network Topology
- OSS Community Structure
- Simulation and Validation
DOCKING

- Docking
  - Verify simulation correctness
  - Discover pros & cons of toolkits

- Four Models of OSS
  - random graphs
  - preferential attachment
  - preferential attachment with constant fitness
  - preferential attachment with dynamic fitness

- Agent-based Simulation
  - Swarm
  - Repast
DEVELOPER ACTIVITIES

New developer
- create
- join
- add a new project
- add a link

Existing developer
- abandon
- idle
- remove a link
- remove a project
VALIDATION USING DIAMETERS
CONCLUSIONS

- Addressed the challenge of data collection
- Classified roles of OSS developers
- Analyzed network properties of OSS community
- Identified community structures
- Explained the formation of community structure
- Presented simulation models and the docking process
FUTURE WORK

- Add weights on vertices and edges
  - More realistic representation
- Apply more social network property analysis
  - Closeness
- Include more OSS websites
  - Linux
  - Perl
  - Apache
  - Savannah
CONTRIBUTIONS

- Presented a general solution for mining data
- Found features of OSS community
- Provided useful topological information on the underlying structure and evolution of OSS community
- Provided useful information on the communication and the diffusion of information between projects
- Demonstrated the use of docking for validation of agent-based simulation models
PUBLICATIONS

- Renee Tynan, Gregory Madey, Scott Christley, Vincent Freeh and Jin Xu, "Task characteristics, communication, and outcome in open source software development", under submission
POTENTIAL PUBLICATIONS

- Chapter 2
  - Journal of Knowledge and Information Systems
- Chapter 4
  - Journal of Organizational Computing
- Chapter 5
  - Journal of Artificial Societies and Social Simulation
ACKNOWLEDGEMENT

- My advisor – Greg Madey
- My colleagues – Scott Christley, Yongqin Gao, Yingping Huang, Jeff Goett, Ryan Kennedy
- Professor Renee Tynan
- My committee members – Dr. Bowyer, Dr. Flynn, Dr. Cosimano, and outside chair – Dr. Seabaugh
- NSF, CISE/IIS-Digital Science and Technology
THANK YOU
PREVIOUS RESEARCH

- **Roles Classification**
PREVIOUS RESEARCH (cont.)

- Social Network Properties
**NETWORK PROPERTIES**

- **Diameter**

\[ d = \frac{\log(N/z_1)}{\log(z_2/z_1)} + 1 \]

- \( N \) – the total number of nodes
- \( z_1 \) – the average number of neighbors 1 link away
- \( z_2 \) – the average number of neighbors 2 links away

- **Clustering coefficient**

\[ c = \frac{1}{1 + \frac{(\mu_1 - \mu_2)(v_1 - v_2)^2}{\mu_2 v_1 (2v_1 - 3v_2 + v_3)}} \]

- \( \mu_n = \sum_k k^n p_d^k \), \( p_d^k \) represents the fraction of developers who join \( k \) projects
- \( v_n = \sum_k k^n p_p^k \), \( p_p^k \) represents the fraction of projects which have \( k \) developers
DOCKING

- Three ways of Validation
  - Comparison with real phenomenon
  - Comparison with mathematical models
  - Docking with other simulations

- Docking
  - Verify simulation correctness
  - Discover pros & cons of toolkits
SOCIAL NETWORK MODEL

- Graph Representation
  - Node/vertex – Social Agent
  - Edge/link – Relationship
  - Index/degree - The number of edges connected to a node

- ER (random) Graph
  - edges attached in a random process
  - No power law distribution
SOCIAL NETWORK MODEL (Cont.)

- Watts-Strogatz (WS) model
  - include some random reattachment
  - No power law distribution
- Barabasi-Albert (BA) model with preferential attachment
  - Addition of preferential attachment
  - Power law distribution
- BA model with constant fitness
  - addition of random fitness
- BA model with dynamic fitness
OSS NETWORK

- A classic example of a dynamic social network
- Two Entities: developer, project
- Graph Representation
  - Node – developers
  - Edge – two developers are participating in the same project
- Activities
  - Create projects
  - Join projects
  - Abandon projects
  - Continue with current projects
OSS MODEL

- Agent: developer
- Each time interval:
  - Certain number developers generated
  - New developers: create or join
  - Old developers: create, join, abandon, idle
  - Update preference for preferential models
SWARM SIMULATION

- ModelSwarm
  - Creates developers
  - Controls the activities of developers in the model
  - Generate a schedule

- ObserverSwarm
  - Collects information and draws graphs

- main

- Developer (agent)
  - Properties: ID, degree, participated projects
  - Methods: daily actions
REPAST SIMULATION

- Model
  - creates and controls the activities of developers
  - collects information and draws graphs
    - Network display
    - Movie
    - snapshot

- Developer (agent)
- Project
- Edge
DOCKING PROCEDURE

- Process: comparisons of parameters corresponding models.
- Findings:
  - Different Random Generators
  - Databases creation errors in the original version
  - Different starting time of schedulers
DOCKING PARAMETERS

- Diameter
  - Average length of shortest paths between all pairs of vertices

- Degree distribution
  - The distribution of degrees throughout a network

- Clustering coefficient (CC)
  - $CC_i$: Fraction representing the number of links actually present relative to the total possible number of links among the vertices in its neighborhood.
  - $CC$: average of all $CC_i$ in a network

- Community size

Ph.D defense, CSE Dept. Univ. of Notre Dame
DEGREE DISTRIBUTION
CLUSTERING COEFFICIENT
COMMUNITY SIZE DEVELOPMENT
CONCLUSION

- Same results for both simulations
- Better performance of Repast
- Better display provided by Repast
  - Network display

Random Layout

Circular Layout
DOCKING PROCESS

Docking

toolkits

Swarm

Repast

OSS models

ER

BA

BAC

BAD

parameters

Diameter

Degree distribution

Clustering coefficient

Community size

Ph.D defense, CSE Dept. Univ. of Notre Dame