

Analysis of Activity in the Open Source Software Development Community

Scott Christley and Greg Madey

Dept. of Computer Science and
Engineering

University of Notre Dame

*Supported in part by National Science Foundation, CISE/IIS-Digital Society &
Technology, under Grant No. 0222829*

Overview

- Introduction and Motivation
- Data
- Analysis of Activity
- Methods
- Results
- Discussion and Conclusions

Introduction

- FLOSS: continuing to grow in popularity and developer participation
- Few very successful projects, ... but many that are not large and/or not successful
- Voluntary participation (individuals & organizations) and many forms of participation (coding, documentation, testing, support, bug reports, feature requests, etc.)
- Multiple large research data archives available that lend themselves to research based on data mining methods
- Our results
 - Analysis of activity: social positions, temporal social positions, and temporal activity patterns
 - New data mining approaches

FLOSS Research Motivation

- Successful software development requires various positions to be filled: developers, testers, administrators, management, end-users, documentation writers, etc.
- Members of Open Source Software communities self-select into a social position on a software project.
- We have insight into these formal roles (see next slides).
- But ... what are the real positions that emerge by self-organization within the community ==> social position (from social network theory)
 - Positional analysis seeks to group actors into disjoint subsets according to their social position in the network.
 - Do people stay in same social position, or does their position change over time?

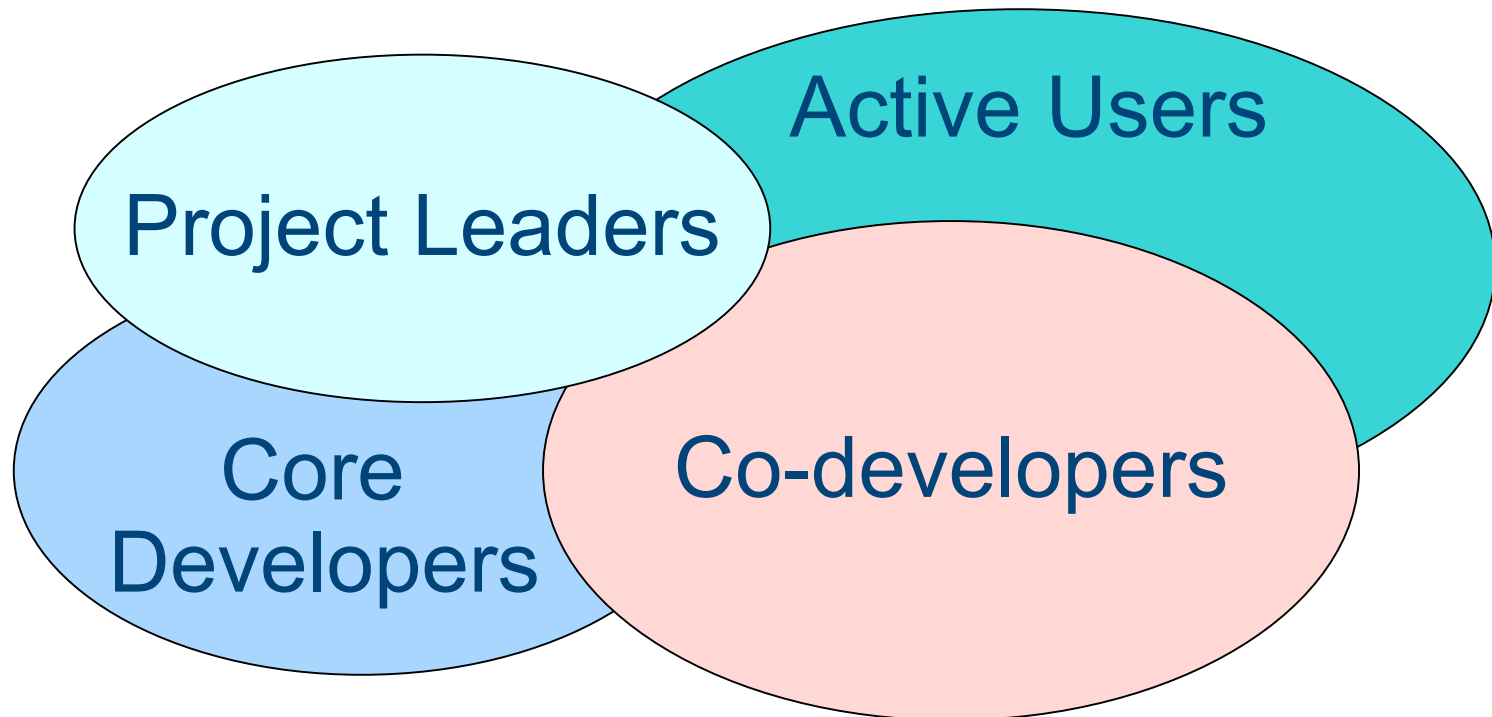


OSS COMMUNITY (previous results)

- User Group
 - Passive Users: no direct attributable contribution in the data (downloads, user base, word-of-mouth publicity, etc.)
 - Active Users: bug reports, patch submissions, feature requests, help requests, etc.
- Developer Group
 - Peripheral Developer: **irregularly** contribute
 - Central Developer: **regularly** contribute
 - Core Developer: **extensively** contribute, manage CVS releases and coordinate peripheral developers and central developers.
 - Project Leader: guide the vision and direction of the project.

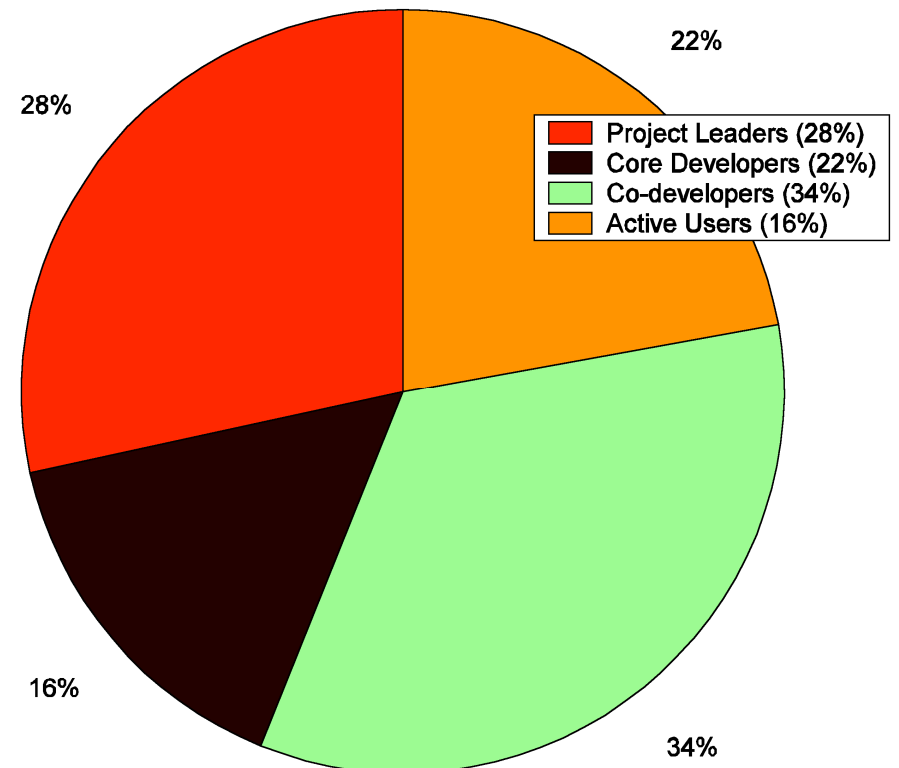
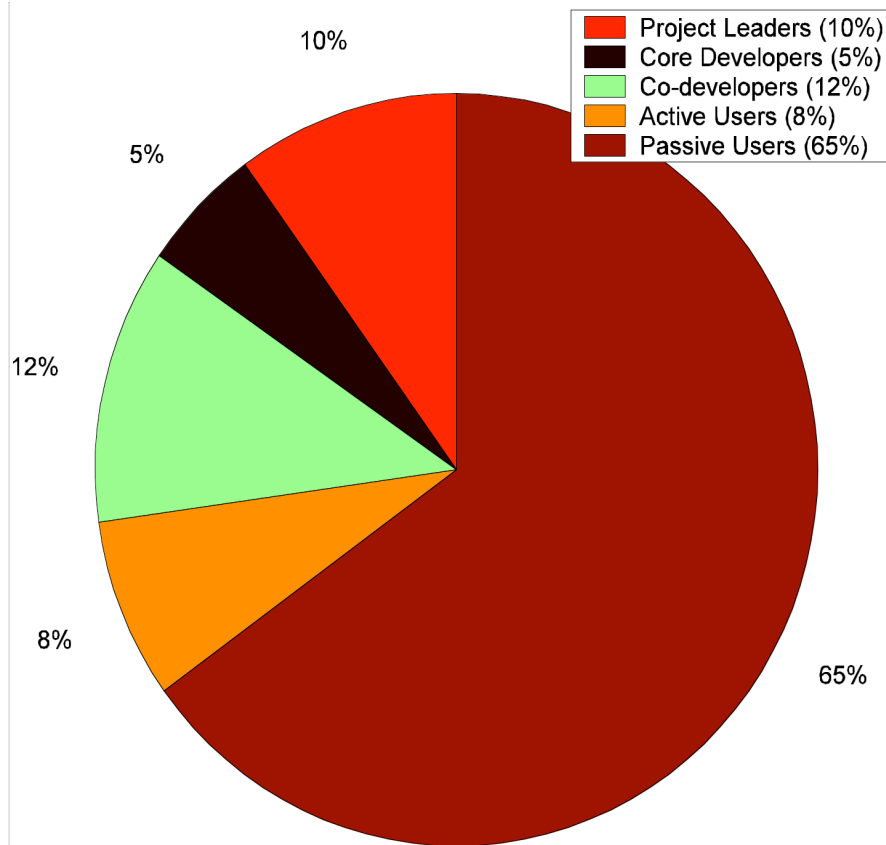


OSS DEVELOPMENT COMMUNITY





Analysis: SourceForge.net Level



Data

- Data sources
 - SourceForge.net Research Data Archive
 - Hosted at University of Notre Dame
 - <http://zerlot.cse.nd.edu/>
 - Available for use by all interested scholarly researchers under sublicense from SourceForge.net
 - SQL queries ==> 21 activity types, 2 million records
 - SourceForge.net CVS source code repositories
 - Client script ==> 8 activity types, 120 million records
 - Tech Report on methods available at archive: TR-2005-15
- Other data sources available
 - FLOSSmole
 - Freshmeat
 - CVSAAnalY
 - Savannah, and many more.

Artifact Activity Types

Activity Type	Activity Description
Submit bug (1)	Person submits a new bug report.
Assign bug (2)	Bug report is assigned to person.
Submit support request (3)	Person submits a new support request.
Assign support request (4)	Support request is assigned to person.
Submit patch (5)	Person submits a new patch.
Assign patch (6)	Patch is assigned to person.
Submit feature request (7)	Person submits a new feature request.
Assign feature request (8)	Feature request is assigned to person.
Submit todo (9)	Person submits a new to-do item.
Assign todo (10)	To-do item is assigned to person.
Submit other artifact (11)	Person submits an artifact that is not one of the predefined categories of bug report, support request, patch, feature request, or to-do item.
Assign other artifact (12)	Uncategorized artifact is assigned to person.

Communication and Management Activity Types

Activity Type	Activity Description
New forum message (13)	Person posts a new forum message.
Followup forum message (14)	Person posts a forum message that is a followup to an existing forum message.
Modify project (15)	Person makes an administrative modification to the project; the modification is uncategorized, but they are typically tasks like adding/removing members, changing permissions, updating project settings, etc.
File release (16)	Person posts a new file release; this is typically associated with releasing a new version of the software to the public.
New project task (17)	Person creates a new project task.
Assigned project task (18)	A project task is assigned to person.
Modify project task (19)	Person modifies an existing project task.
Create document (20)	Person creates a new document.
Create people job (21)	Person posts a new job; these are similar to help-wanted ads where a project is looking for somebody with particular skills.

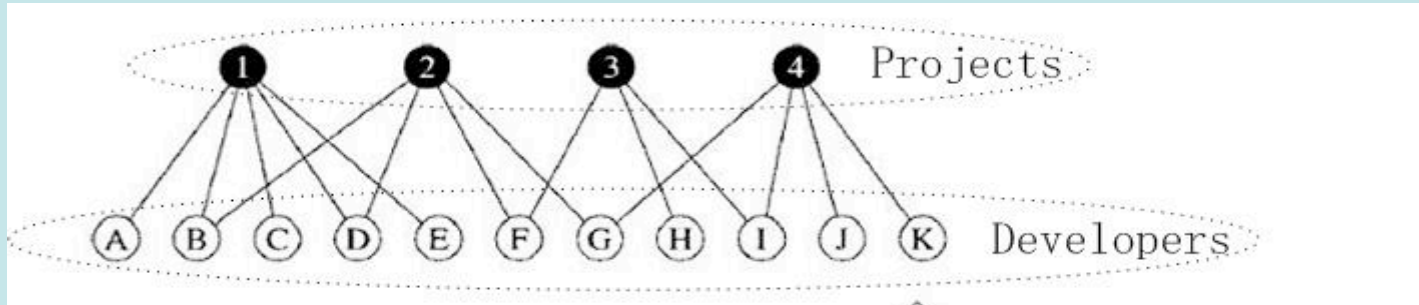
CVS Source Code Activity Types

Activity Type	Activity Description
Checkout source code (22)	Person checks out source code from CVS repository.
Export source code (23)	Person exports source code from CVS repository.
Release source code (24)	Person releases check out of source code from CVS repository.
Tag source code (25)	Person tags source code in the CVS repository with a label.
Add source code file (26)	Person adds a new source code file to the CVS repository.
Remove source code file (27)	Person removes a source code file from the CVS repository.
Modify source code file (28)	Person commits a source code modification to the CVS repository.
Update source code (29)	Person updates local checked out source code with any changes in CVS repository.

Analysis of Activity

- Limited data available on individuals
- Social network analysis can be used to infer information about those users
 - Positional analysis
 - Social position: pattern of embeddedness in the social network
 - Structural equivalence
- Temporal Analysis
 - Absolute time vs relative time
 - Relative time: each person's first appearance in the social network is time zero (month 1)

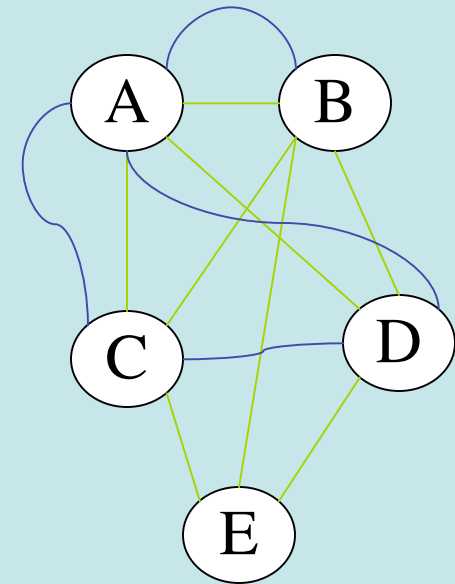
OSS Activity



- User performs an activity for a project.
- 29 activities; submit bug, submit feature request, assign bug, post forum message, create file release, create project task, etc.
- Multi-relational, weighted, bipartite network.
 - Activity = relation, weight = activity count
- Activity distribution for user/project pair defines a sample for our analysis.
- That is, the activity distribution of a user on a project defines their social position for that project.

Structural Equivalence

- Actors who are similarly embedded occupy similar social position.
- $C \sim D$ have same relationships with same other actors.
- Exact equivalence is too strict so use an approximate measure, like Euclidean distance.
- Weighted relationships



Methods

- Discovery of social positions
 - Clustering (metric approaches) => not suitable
 - Clustering (activity distribution) => new algorithm
- Discovery of temporal social positions
 - Extension of the above clustering “new algorithm”
- Discovery of temporal activity patterns
 - Method similar to the data mining Apriori Algorithm

Clustering

- Standard data mining algorithms
 - K-means, Expectation-Minimization (EM)
- What's wrong with Euclidean distance?
 - Data mapped to points in an N-dimensional space.
 - Points “close” in space are in same cluster.
 - Normalization techniques very important.
 - Not comparing the underlying distributions.
- Assume Gaussian (normal) distribution
- What can we use instead of a distance metric?
 - Statistical test

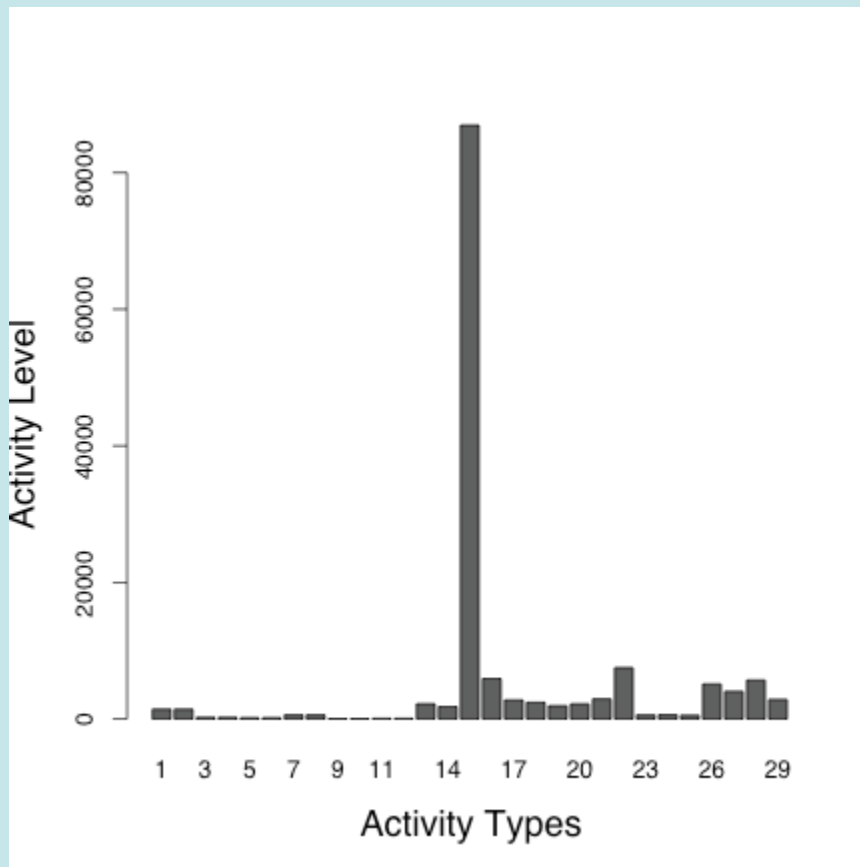
Clustering with a Statistical Test

- Fisher's contingency-table test (non-parametric)
 - Chi-square family of goodness-of-fit tests
- Given two independent samples
 - First sample, S_1 , with n_1 random variables
 - Second sample, S_2 , with n_2 random variables
 - Where n_1 not necessarily equal to n_2 , each r.v. in each samples placed in one of C categories.
- H_0 : The distributions of S_1 and S_2 do not differ.
- H_A : The distributions S_1 and S_2 differ.
- Structural In-equivalence

Results

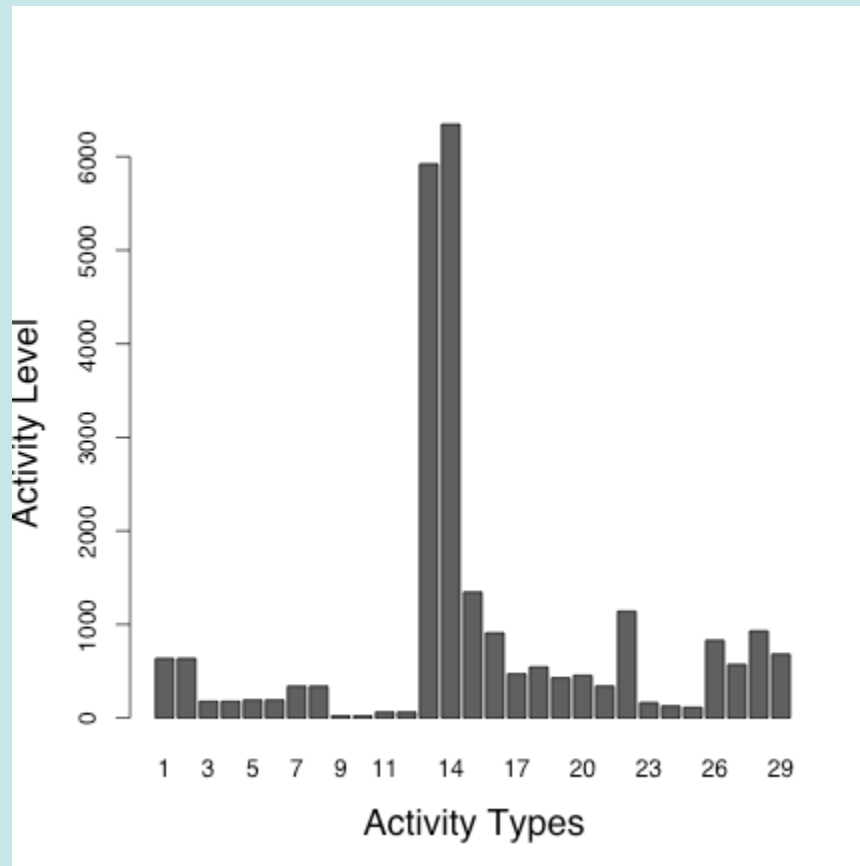
- Seven major social positions discovered
 - Clusters formed using a statistical test
 - Structural equivalence based on similarity of activity distributions
- Six major temporal social positions discovered
 - Relative time based on person's first appearance in the social network
- Several high frequency software development processes identified

Project Administrator



Primary activity:
Modify project
(15)

Message Poster

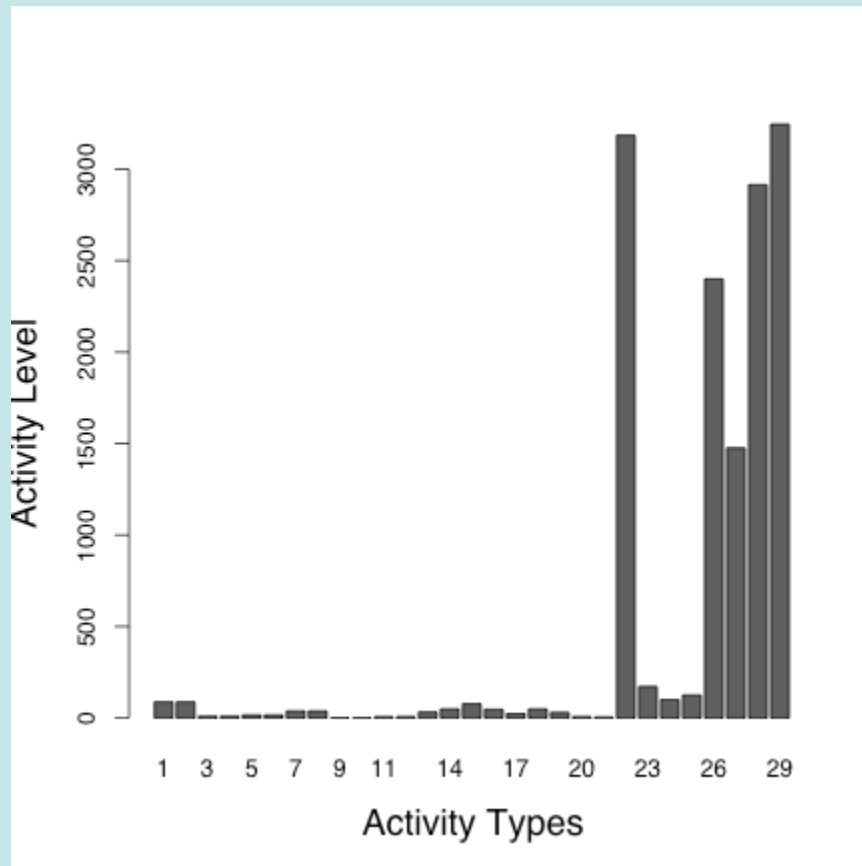


Primary activities:

New forum message (13)

Followup forum message (14)

Software Developer



Primary activities:

Checkout source code (22)

Add source code file (26)

Remove source code file (27)

Modify source code file (28)

Update source code (29)

Social Positions at Sourceforge.net

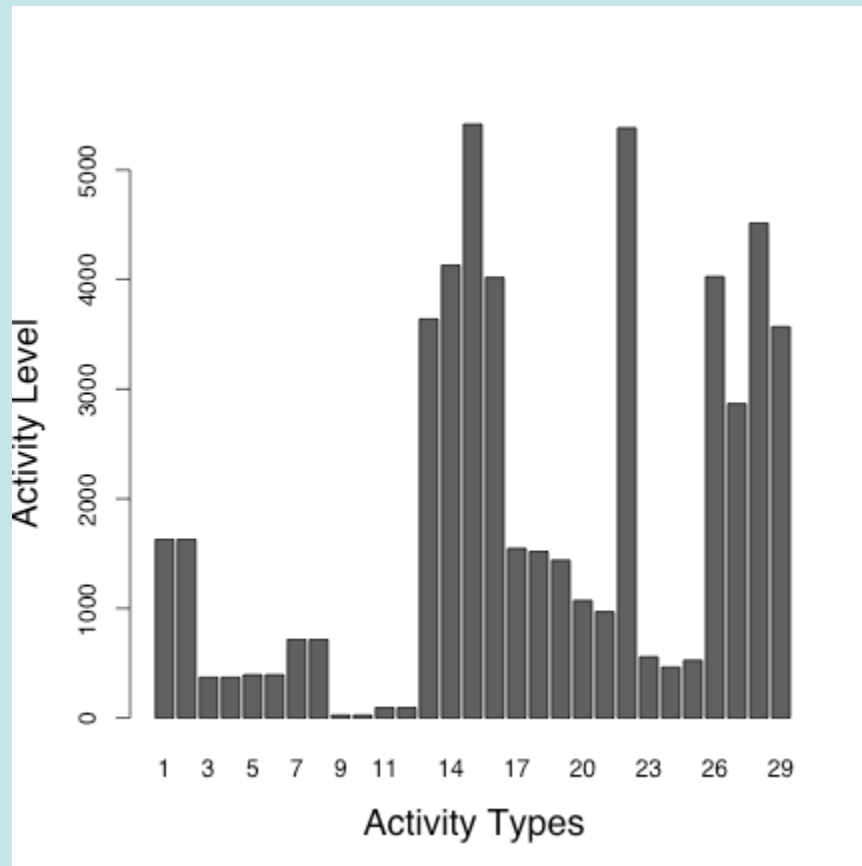
Social Position	Description	Size
Software User	The largest cluster with the primary activities of posting new forum messages, followup forum messages, and checkout out source code.	111889
Project Administrator	The second largest cluster with the primary activity of making project modifications; the project administrator also performs file releases, but most other activities are relatively minor or non-existent.	93199
Software Developer	Primary activities are source code operations like checking out source code, add/remove source code files, modify source code, and update source code. The social position contains 39 clusters all with different relative proportions of the source code operations, and some software developers have significant levels of project modification and file release activities.	47495
Task Management	Significant usage of the project task management provided by SourceForge.net.	2181
Bug Reporter	Significant bug reporting activity with a slight amount of features requests, support requests, and patches.	1138
Feature Requester	Primary activity was submission of feature requests but also has a significant amount of bug reporting.	370
Handyperson	The handyperson has significant activity for many different activity types including source code modifications, bug reporting, project modifications, file releases, and project tasks.	271
Not Categorized	The remaining very small clusters that were not analyzed.	14818
Total User/Project Pairs		271307

Temporal Social Positions at Sourceforge.net

Social Position	Month 1	Month 2	Month 3	Month 4
Project Administrator	86951	0	0	0
Message Poster	96052	7315	0	0
Software Developer	67488	32126	21054	18239
Release Management	11700	7227	0	0
Task Management	1775	0	0	0
Handyperson	1768	120	14050	10709
Not Categorized	3066	1638	1712	1611
Total User/Project Pairs	268800	48426	36816	30599

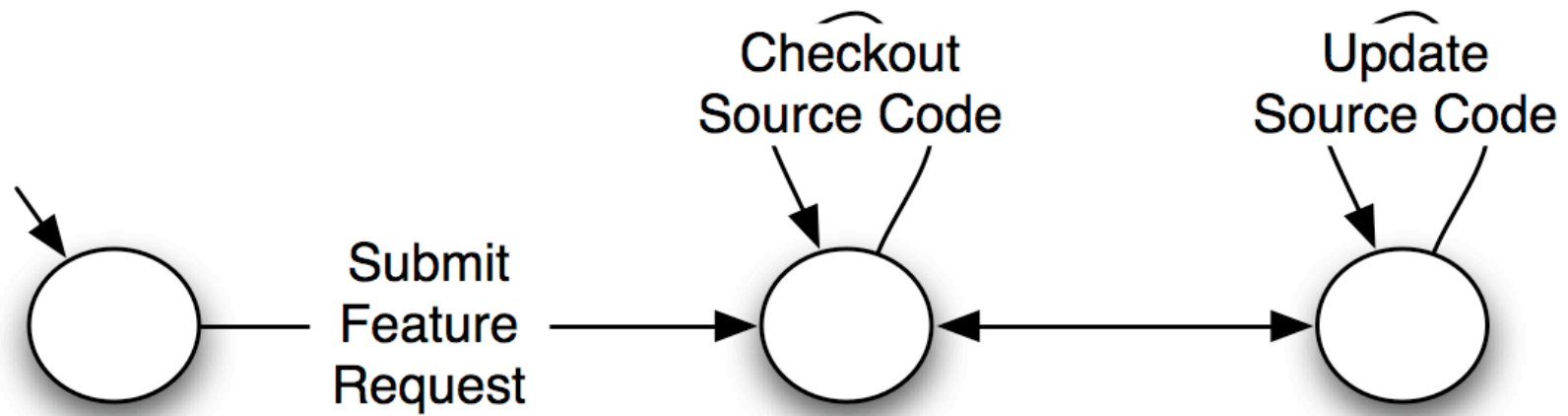
- Dip in total after first month: Many people drop out after their first month of activity.
- Rise of the Handyperson by the third month to take over duties of project administration, release management, etc.

Handyperson



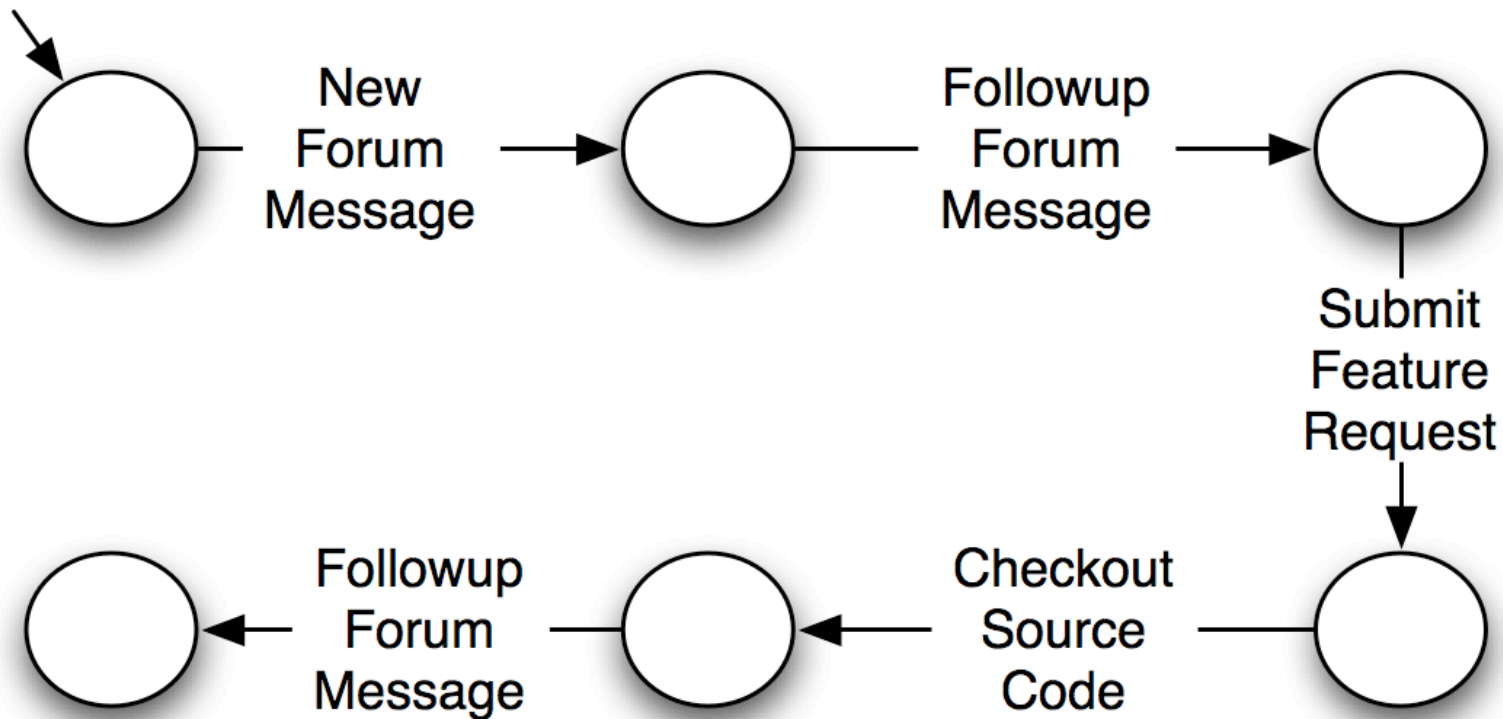
The handyperson has significant activity for many different activity types including source code modifications, bug reporting, project modifications, file releases, and project tasks.

Feature Request



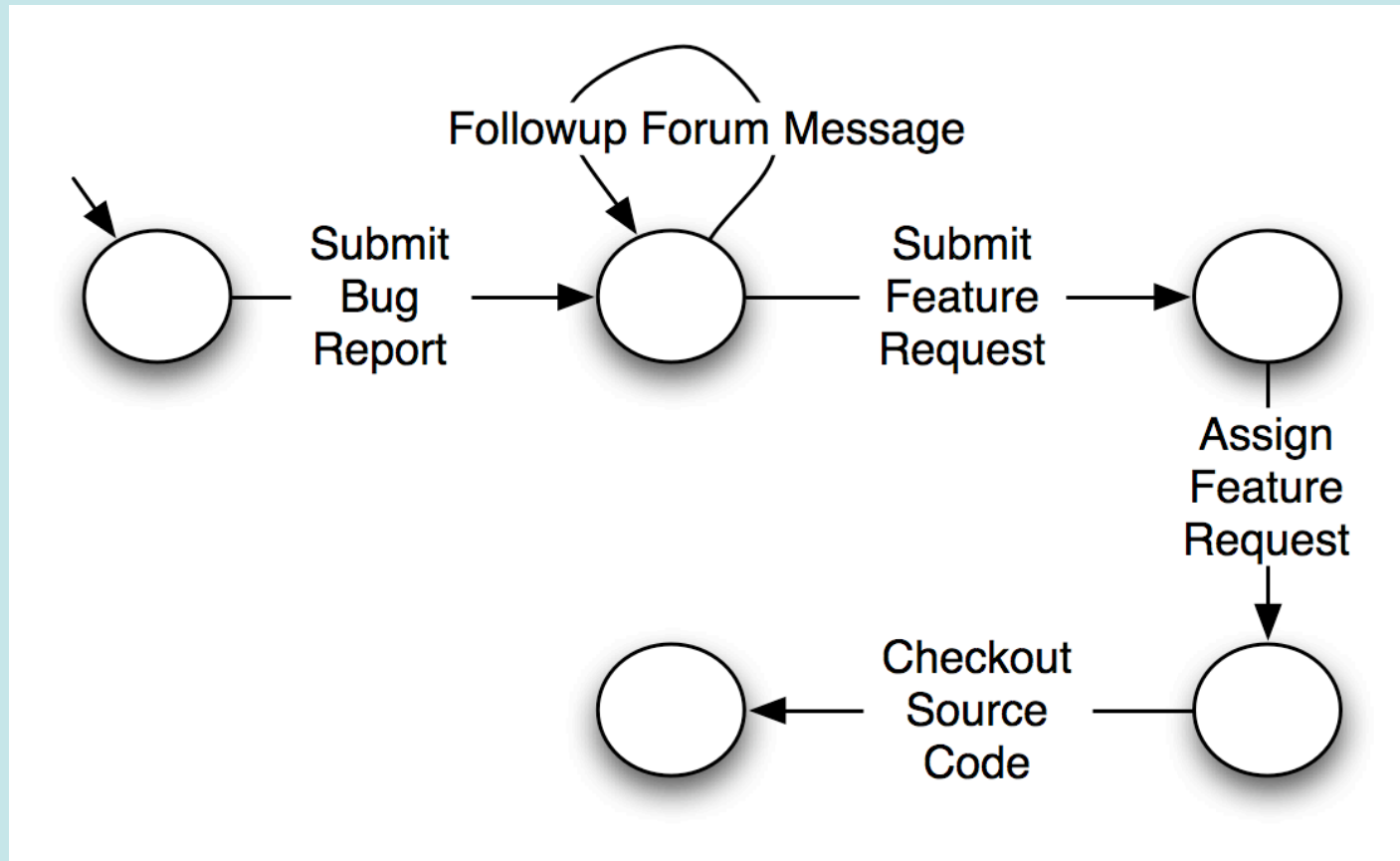
Typical process which shows an initial submission of a feature request followed by a series of checkouts and updates of the source code.

Feature Request



Possible process for a feature request discussion, submission, and resolution.

Bug Report -> Feature Request



Possible process for a bug report being turned into a feature request.

Discussion

- Availability of large electronic data archives and data mining enables research on FLOSS
 - Identified social positions that emerge on projects, both static and temporal analysis
 - Temporal analysis shows that most specialized positions disappear after a few months leaving only the software developer and handyperson
 - Many 1 month contributors!
- Limitations
 - We did not use all available data in the archives
 - Large amount of data, but important data not in electronic archives
 - Potential automation bias (only looking under the light posts!)
 - Did not “talk” to the people!

Conclusions

- Demonstrated the potential to discover a great deal about FLOSS ... increasing our understanding of the phenomenon
- Displayed methods for data mining the digital archives
- There is a value is collecting, integrating, improving and currating research data archives
- Sourceforge.net Research Data Archive
 - <http://zerlot.cse.nd.edu/>

[article](#)
[discussion](#)
[view source](#)
[history](#)

Main Page

SourceForge Research Data Archive: A Repository of FLOSS Research Data

- This Wiki contains information about using the SourceForge Research Data Archive. Links to query the SourceForge Research data warehouse can be found to the right. Your same userID and password that provides access to the query form, also can be used to login to the wiki. After logging in, all content can be edited, including creating new pages.
- This Data Repository is a by-product of an NSF funded research project on "Understanding Open Source Software". See [Introduction](#) for more information about the project. Data obtained from SourceForge.net to support that project, are made available to the academic and scholarly research community under a [sublicense](#) from SourceForge.net.
 - This Data Repository is hosted by the Department of Computer Science & Engineering, University of Notre Dame and administered by Greg Madey
 - The material presented at this web site is based in part upon work supported by the National Science Foundation, CISE/IIS-Digital Society & Technology, under Grant No. 0222829.
 - Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
 - For questions, comments, and suggestions, send email to <oss at nd.edu>

Recent News

- November schema ([sf1106](#)) has been loaded.

Top Links

- [Schema Browser](#)
- [Query Form](#)
- [Research Data](#)
- [Making Queries](#)
- [Resources](#)
- [Papers](#)
- [Contact](#)
- [FAQ](#)
- [Schemas](#)
- [All tables](#)



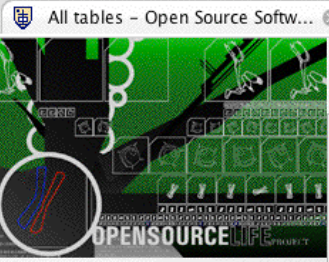
navigation

- [Main Page](#)
- [Introduction](#)
- [OSSR Dataset](#)
- [SQL Query](#)
- [Resources](#)
- [FAQ](#)
- [Suggestions](#)
- [Schemas](#)
- [Help](#)

search

toolbox

- [What links here](#)
- [Related changes](#)
- [Upload file](#)
- [Special pages](#)
- [Printable version](#)
- [Permanent link](#)



[article](#) [discussion](#) [edit](#) [history](#)

All tables

Through October 2006:

- [activity_log](#)
- [activity_log_old](#)
- [activity_log_old_old](#)
- [activity_log_regs](#)
- [activity_log_regs_tmp](#)
- [admin_annotations](#)
- [artifact](#)
- [artifact_canned_responses](#)
- [artifact_category](#)
- [artifact_counts_agg](#)
- [artifact_file](#)
- [artifact_ftispool](#)
- [artifact_group](#)
- [artifact_group_list](#)
- [artifact_history](#)
- [artifact_message](#)
- [artifact_monitor](#)
- [artifact_perm](#)
- [artifact_resolution](#)
- [artifact_status](#)
- [audit_trail](#)

Schemas
▪ January 2003 - sf0103
▪ November 2004 - sf1104
▪ December 2004 - sf1204
▪ February 2005 - sf0205
▪ March 2005 - sf0305
▪ April 2005 - sf0405
▪ May 2005 - sf0505
▪ June 2005 - sf0605
▪ July 2005 - sf0705
▪ August 2005 - sf0805
▪ September 2005 - sf0905
▪ October 2005 - sf1005

- navigation
- [Main Page](#)
 - [Introduction](#)
 - [OSSR Dataset](#)
 - [SQL Query](#)
 - [Resources](#)
 - [FAQ](#)
 - [Suggestions](#)
 - [Schemas](#)
 - [Help](#)

search

- toolbox
- [What links here](#)
 - [Related changes](#)
 - [Upload file](#)
 - [Special pages](#)
 - [Printable version](#)
 - [Permanent link](#)

Most Recent Description

[\[edit\]](#)

Table "sf1106.users"

Column	Type	Modifiers
user_id	integer	not null default nextval(('users_pk_seq'::text)::regclass)
user_name	text	not null default ''::text
realname	character varying(32)	not null default ''::character varying
status	character(1)	not null default 'A'::bpchar
unix_uid	integer	default 0
add_date	integer	not null default 0
people_resume	text	not null default ''::text
timezone	character varying(64)	default 'GMT'::character varying
language	integer	not null default 275
cf_uid	integer	
stay_anon	integer	default 0
donation_request	text	
donate_optin	integer	default 0
last_sitestatus_view	integer	default 0
row_modtime	integer	
Indexes: users_pkey primary key btree (user_id),		
users_uniqid unique btree (user_id),		
users_uniqname unique btree (user_name)		

Thank You

Questions?

<http://zerlot.cse.nd.edu/>

Extra Slides

Concurrent Versions System (CVS)

- Source code management system.
 - Client/server architecture
- Uses the sandbox model, each developer has their own copy of the source code. Change conflicts are handled upon commit.
 - Compare to lock model where the developer acquires an exclusive lock to modify a file.
- Commits are performed wholesale; i.e. commit a whole set of changes at once versus file by file.
- CVS maintains history records for server operations.
 - We parsed these history records to get CVS activity.

CVS Workflow

- Developer
 - cvs checkout (obtain local copy of source code)
 - cvs update (pull changes from server committed by other developers into local copy)
 - cvs add/remove (add/remove files from local copy)
 - cvs commit (commit changes in local copy to server repository)
- Release Management
 - cvs tag (attach a label to the source code, allows retrieval of exact version).
 - cvs checkout (option to create separate development branches; i.e. support released/development versions at the same time)
 - cvs export (local copy of source code minus CVS meta-data, suitable for public release)

Algorithm 1 cluster(S : set of samples)

```
 $UC = S; i = 1;$ 
while  $UC \neq \emptyset$  do
   $C_i = UC; UC = \emptyset;$ 
  while some samples not yet pairwise compared do
     $A =$  Pick unmarked sample from cluster,  $C_i$ 
    for each other sample,  $B$ , in cluster  $C_i$  do
      Run statistical test on  $A$  and  $B$ 
      if significant result then
         $C_i = C_i/B;$ 
         $UC = UC \cup B;$ 
      end if
    end for
    Mark  $A$  as being pairwise compared
  end while
   $i = i + 1;$ 
end while
return  $C_{1\dots i}$ 
```

Algorithm 2 countCandidates(S : activity sequence, C : set of candidates)

```
int counts[length of C] = 0;
for each candidate  $c \in C$  do
  boolean flag[length of S] = false;
  repeat
    found = false;
    k = 0; // event index for candidate
    // Scan through activity sequence
    for j = 0; j < length of S; ++j do
      if flag[j] then
        continue; // already matched
      end if
      // Does candidate event match sequence event
      if c[k] == S[j] then
        ++k;
        flag[j] = true;
      end if
      // Found match for all events in candidate
      if k == length of candidate c then
        found = true;
      end if
    end for
    if found then
      ++counts[c]; // increment count
    end if
  until found == false;
end for
return counts;
```

Algorithm (Intersection)

While (still unclustered samples)

Put all unclustered samples into one cluster.

While (some samples not yet pairwise compared)

A = Pick sample from cluster

For each other sample, B, in cluster

Run statistical test on A and B.

If significant result

Remove B from cluster.

- Rejection of null hypothesis means A and B **must** be in different clusters.
- Confidence level tightens/broadens cluster inclusion.
- Any statistical test for a two-sided test problem.

Social Positions of OSS

Social Position	Size	# of clusters
Brief Flame	122654	1
Message Posting	50067	4
Task Management	2762	5
Release Management	6509	5
Documentation	1266	4
Job Posting	899	2
Artifact Management	1674	6
Administrators	10377	4
Not Categorized	13786	1546
Total User/Project Pairs	209994	

Temporal Analysis

- Previous analysis, activity over 10 years, lose knowledge of evolution of positions.
- How to deal with time (data)?
 - Global time; snapshot of the whole network at points in time: node/edge add/remove, attribute change, tends to get aggregate measures.
 - Local time; user/project's first activity is time 0, aligns actors in a time-relative way to the network, egocentric viewpoint.
- Chunk data into monthly activity, run clustering algorithm for data for each time period.

Temporal Social Positions of OSS

Social Position	Period 1	Period 2	Period 3	Period 4
Brief Flame	127302	0	0	0
Message Posting	49754	1418	828	151
Administrators	10356	5415	905	496
Release Management	6304	1001	796	869
Task Management	3466	625	254	401
Artifact Management	1967	0	0	0
Documentation	1130	0	0	0
Job Posting	1125	0	0	0
Not Categorized	4904	2002	1313	1105
Handyperson		7282	8280	6664
Total User/Project Pairs	206308	17743	12376	9686
Total Clusters	397	183	143	139

Summary

- Clustering algorithm using a statistical test.
 - Don't have to specify # of clusters a priori.
 - No assumption of underlying distribution.
 - Must be appropriate statistical test.
- Temporal Analysis
 - How you organize/view your data is important.
 - Global metrics --> global time
 - Egocentric measures --> local time

Iterative Classification

- Order of comparison matters.
- Clustering is NP-complete so intractable to check all combinations to find the optimal.
- Iterative approach
 - Perform initial clustering
 - Calculate cluster center