End-User Programmers and their Communities: An Artifact-based Analysis

Kathryn T. Stolee, Sebastian Elbaum, and Anita Sarma
Department of Computer Science and Engineering
University of Nebraska – Lincoln
Lincoln, NE, U.S.A.
{kstolee, elbaum, asarma}@cse.unl.edu

***Abstract*—End-user programmers outnumber professionals programmers, write software that matters to an increasingly large number of users, and face software engineering challenges that are similar to their professionals counterparts. Yet, we know little about how these end-user programmers create and share artifacts as part of a community. To gain a better understanding of these issues, we perform an artifact-based community analysis of 32,000 mashups from the the Yahoo! Pipes repository. We observed that, like with other online communities, there is great deal of attrition but authors that persevere tend to improve over time, creating pipes that are more configurable, diverse, complex, and popular. We also discovered, however, that end-user programmers employ the repository in different ways than professionals, do not effectively reuse existing programs, and in most cases do not have an awareness of the community. We discuss the implications of these findings.**

I. Introduction

The population of end-user programmers is quickly overwhelming that of professional programmers. In 2005 there were an estimated 55 million end-user programmers and 3 million of professional programmers in the United States. The number of end users was projected to increase to 90 million in 2012 with 13 million describing themselves as programmers [1], but it is not just their numbers that matter. Despite their lack of computer science education, end-user programmers are increasingly creating programs that are meaningful and have consequences not just to them or the businesses for which they work (e.g., a spreadsheet formula error reportedly cost a company millions of dollars [2]), but also for emerging online communities. These communities are growing rapidly, exist in many domains, and facilitate knowledge sharing and code reuse. For example, the public repositories of mashups in Yahoo! Pipes [3], web macros in CoScripter [4], web page modification scripts in Userscripts [5], and animations in Scratch [6] have tens of thousands of program artifacts submitted by tens of thousands of users.

As they develop software, these end-user programmers confront some of the same challenges as professional developers and their communities. For example, as individuals, they need to configure sample code to run in their environments, use new APIs, or find a fault causing a failure. As a group, they need to learn how to build on, share, and contribute to the community. Yet, our understanding of the challenges, motivations, and needs of these end-user programmers and their communities is quite limited. Studies of online end-user communities have sought to characterize the participants roles using social evidence [7], but little is known about the type, quantity, and quality of artifacts contributed, and how end users and their contributions evolve over time.

Building on the success of studies of open source communities through public archives (e.g., [8] [9] [10] [11]) this work aims to provide a better understanding of end-user programmers in a community setting. We perform a study of over 32,000 programs submitted to the Yahoo! Pipes public repository, characterizing the artifacts and using them to draw inferences about author behavior, skill levels, and community awareness. Specifically, we address three general research questions: *What are the characteristics of the Yahoo! Pipes community? What are the differences in pipe characteristics as authors gain experience?* and *What are the characteristics of the most prolific authors?*

Our findings reveal that, like with other online communities, there is great deal of attrition as over 81% of the authors we studied are active (i.e., contribute artifact(s) to the repository) for only one day. We also observe that the authors who persevere tend to improve over time, creating pipes that are more configurable, diverse, complex, and popular. We also discovered, however, that end-user programmers employ the repository in different ways than professionals. Approximately half of the most prolific authors usually create pipes that are very similar to pipes they had created in the past, causing the repository to be full of duplication. Additionally, authors do not effectively reuse programs in the repository, and in most cases do not demonstrate an awareness of the community; only 30% of the authors regularly submit pipes that are highly unique compared to other pipes in the repository.

II. Related Work

Two areas of related work require discussion: end-user programmers and studies on socio-technical communities with artifact repositories.

A. End-user Programmers

End-user programmers create programs and engage in programming activities to support their hobbies and work. What differentiates end-user programmers from professional programmers is that to end users, software is a means to an end, not the end itself [12]. These end users utilize programming environments and languages such as spreadsheets, databases, web macros, mashups, and many domain-specific scripting languages, many of which have large public repositories (e.g., [3] [4] [5] [6]).

Unlike professional programmers, end-user programmers do not have much support for all stages of the software lifecycle, and may have a different lifecycle than that which is used by other types of programmers. Studying end-user programmers can reveal their needs, and researchers and practitioners have started applying software engineering techniques to provide support for end users' tasks. For example, version control has been introduced to help end users during development [13] [14], debugging has been introduced to allow users to ask questions about output during development [15] or preview program output during testing [3], assertions have been used to increase the dependability of web macros during runtime [16], and strides have been made toward providing better program maintenance through refactoring support [17] and using program characteristics to predict how likely a program is to be reused [18]. However, software engineering support is far from pervasive in end-user programming environments.

Repositories provide a mechanism for end-user programmers to share code and learn from the experiences of others, and tend to attract many participants to the communities. For example, Yahoo! Pipes has over 90,000 users [7], CoScripter has over 6,000 users [18], Userscripts has over 57,000 users [5], and Scratch has over 500,000 users [19]. Beyond the number of participants, the repositories maintained by these communities contain thousands of public artifacts. For example, the Yahoo! Pipes repository contains over 92,400 artifacts [3], the CoScripter repository contains over 5,430 public scripts [4], the Userscripts repository contains over 57,200 scripts [5], and the Scratch website contains over 47,800 galleries with as many as 1,944 projects per gallery [6].

B. Studies on Communities

Researchers in software engineering and computer supported cooperative-work have sought to understand the motivations and social organizations of developer communities. Research on communities with public artifact repositories has been particularly successful in open-source (e.g., [8] [10]), and researchers are beginning to leverage repositories to also study end-user programmer communities (e.g., [7] [19]). Even though the open-source and end-user programmer communities have many differences, there are commonalities in the activities performed by some roles in the open-source communities, such as bug fixers and bug reporters, and by programmers in end-user communities [20]. Here, we consider previous work that explores how developers join these communities and social factors that govern their contributions.

Becoming an active member of an open source project is meritocratic, with joiners starting at low technical skill and low responsibility roles, such as participating in the mailing list. As they gain more experience learning both the technical parts and the social norms in the project, they advance to more central roles [8] [9] [10] [11]. Contrastingly, becoming an active member in many end-user programmer communities seems to be universally accessible. Contributors are typically not required to demonstrate any expertise to participate.

In open source communities, most communication and project activities are archived through mailing lists, bug discussions, bug activities, versioning systems [8]. End-user communities, on the other hand, have been observed to communicate through user comments associated with artifacts [18] [19] and public message boards [7]. These differences in communication mechanisms may be rooted in fundamental differences between the groups, where generally the open-source programmers work toward a common goal and end-user programmers work toward an individual goal [12].

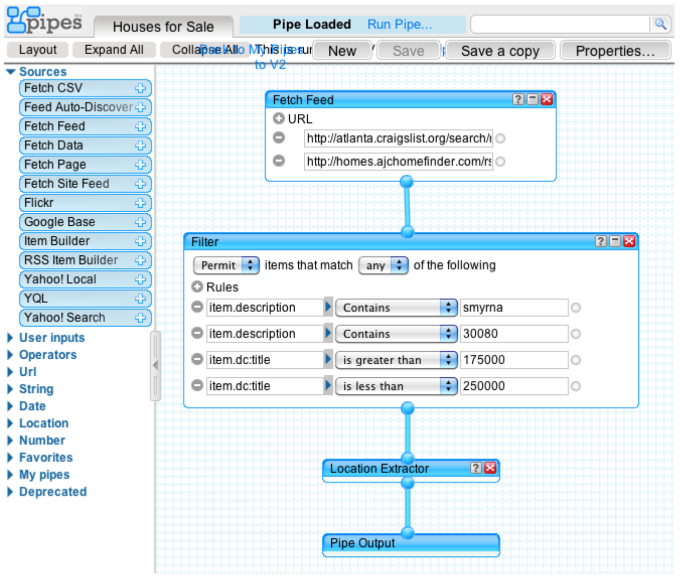
Social factors have been shown to be important in both the social organization and retention of developers in open-source communities. Studies have identified strong inherent social structures in the open-source community based on mail messages and found that successful members were also social hubs [21]. Power law relationships have been shown to hold on project sizes, the number of developers per project, and project memberships (number of projects joined by a developer). This is largely because of social relations, where members like to join projects that are already popular or join projects where they know some of the key players [22]. Another study found that the social network and the strength of the ties in the community was a good indicator for retention of members in the community in the face of external factors such as external projects and monetary incentives [23].

Yet for end-user communities, and specifically for Yahoo! Pipes – the particular subject of our study – the social factors may be different. Previous work has explored the nature of participation in the Yahoo! Pipes message boards [7], but little is known about the organization, participation, and growth patterns for the participants who contribute to the public artifact repository.

III. About Mashups and Yahoo! Pipes

The Yahoo! Pipes community is among the largest end-user programmer communities that has emerged in recent years. Released in 2007, the Pipes environment provides language and development support for the creation of web mashups.

A mashup is an application that manipulates and composes existing data sources or functionality to create a new piece of data or service that can be plugged into a web page or integrated into an RSS feed aggregator. One common type of mashup, for example, consists of grabbing data from some data sources (e.g., house sales, vote records, bike trails, map data), joining those data sets, filtering them according to a criterion, and plotting them on a map published at a site [24]. This type of behavior is naturally expressed on Yahoo! Pipes, as shown in Figure 1, which provides an example of the Pipes Editor, the Yahoo! Pipes development environment, and shows a pipe taken from the community that plots home sale information on a map.



http://atlanta.craigslist.org/search/
http://homes.ajchomefinder.com/rs

Permit items that match any of the following

item.description Contains smyrna
item.description Contains 30080
item.dc:title is greater than 175000
item.dc:title is less than 250000

Fig. 1. Yahoo! Pipes development environment operating within a browser, and an example pipe

The structure of a pipe resembles a graph, where the nodes are referred to as modules (boxes in the figure), and the edges are referred to as wires (connections between the modules). The behavior of the pipe can be best understood from top to bottom, as the data flows in a directional manner from the top of the pipe through the output at the bottom. At the top is a module named *Fetch Feed*, which accesses external data sources and provides data to the pipe; this module contains two fields, each specifying a different website. The *Fetch Feed* module feeds data to a *Filter* module, which removes data from the feed based on the specified criteria. In the example, the filter module permits data that matches any of the four specified criteria. Next, a *Location Extractor* module geotags the data, allowing it to be plotted on a map. Lastly, the data flows to an *Output* module, the final module for any pipe.

An author can create a pipe from scratch or by cloning an existing pipe. Once a pipe has been created, it can be shared with the community; all pipe authors are free to contribute to the public repository. An author can commit any of their own pipes by clicking the *publish* button from a pipe's infomation page (accessible by clicking the *Run Pipe...* link from the Pipes Editor, shown at the top of Figure 1).

IV. Study

Through exploration of the Yahoo! Pipes repository, we have identified several research questions and conducted an empirical study to assess those questions.

A. Research Question

We pose three broad research questions in this work. The first is about the community at large, the second is about pipe characteristics as authors gain experience, and the third is about the characteristics of the most prolific (most contributing) authors in the Yahoo! Pipes community.

RQ1: What are the characteristics of the Yahoo! Pipes community?

- *RQ1a:* How much attrition is there among the authors in the community?
- *RQ1b:* How much do authors typically contribute?
- *RQ1c:* What are the general characteristics of the pipes in the community, considering structural diversity, popularity, size, and configurability?

RQ2: How do pipe characteristics change as the authors gain experience?

- *RQ2a:* What are the differences between pipes contributed when authors are new to the community versus when they have been involved for a determined amount of time?
- *RQ2b:* What are the differences between pipes contributed by authors with few contributions versus those contributed by authors with many contributions?

We view the most prolific authors as those who have the greatest impact on the repository in terms of quantity. This leads us to explore characteristics of these authors' contributions:

RQ3: What are the characteristics of the pipes created by the most prolific authors? That is, how different are a prolific author's contributed pipes compared to *their previous contributions* and *the pipes in the community*?

- *RQ3a:* What implications does the uniqueness of an author's contributions have for the types of activities in which the author engages?
- *RQ3b:* What implications does the uniqueness of an author's contributions over time have for an author's evolution in terms of skill level and the value provided to the community?
- *RQ3b:* What implications does the uniqueness of an author's contributions have for the author's awareness of the community?

For our analysis we look at four dependent variables: the configurability, popularity, and size of the pipes, and diversity (or uniqueness) when compared to pipes created by the author and by the community at large. Each of these variables is defined in the next section. We manipulate several independent variables related to author experience to uncover trends, including the days of experience an author had when the pipe was created and number of pipes created by an author.

B. Study Setup

To address the research questions, we conduct an empirical study using artifacts from the Yahoo! Pipes repository. In performing this study, we had three main challenges: obtaining the artifacts, analyzing the artifacts, and measuring the differences (i.e., diversity) among artifacts. In this section, we describe the methods for each of these steps.

1) Artifact Collection: To perform this study we leveraged an infrastructure from a previous study to scrape the Yahoo! Pipes repository for artifacts [17]. Between January and September 2010, we scraped 32,887 pipes created between

TABLE I
SUMMARY OF DIVERSITY METRIC LEVELS

Level	Criteria
1	The structure and content of the pipes are identical
2	The structure and fields per module are the same, but the field values relax
3	Topography/structure is the same; the field counts and values relax
4	Module bag (module names and counts) is the same
5	Module set is the same
6	Type bag is the same
7	Size is the same
8	The pipes exist

TABLE II
DURATION OF AUTHOR ACTIVITY (DAYS)

Duration	# Authors	% Authors
1 day	16,592	81.68%
2 days to 1 week	957	4.71%
1 week to 1 month	655	3.22%
1 month to 6 months	928	4.56%
6 months to 1 year	537	2.64%
1 year to 3 years	631	3.10%
More than 3 years	13	0.06%
all	20,313	100.00%

February 2007 and September 2010 from the Yahoo! Pipes repository. This number corresponds to the set of distinct pipes returned from approximately 50 queries against the repository, each of which returned a maximum of 1,000 pipes. To obtain a representative pool of pipes without restricting the selection based on configuration or structure (since that may impact the effectiveness of this study in terms of measuring diversity among artifacts), we issued queries for pipes that utilized the 50 most popular data sources.[1]

2) Artifact Analysis: Once the artifacts were collected, the next step was to measure different properties of the pipes. The pipe structures are returned from Yahoo!'s servers in JSON [25] format; we were able to reuse parts of the decoding and analysis infrastructure from our previous study [17] and extended it to measure the additional properties needed for this work. Now, we define each of the dependent variables measured for this study:

a) Size: Pipe size is measured by the number of modules. In the example given in Figure 1, the size of the pipe is four, since it has four modules. Every pipe is required to have an *Output* module, and so the minimum size is one for a pipe that has no behavior. Among the sample we studied, the maximum size is 287 with an average of 8.2 and a median of 6.0.

b) Configurability: The configurability of a pipe is measured by the number of *user-setter* modules in a pipe, where a *user-setter* module allows a user to specify field values at run-time [17]. Configurable modules allow authors to create more general pipes that can serve a variety of purposes. Within the sample, one-third of the pipes have at least one *user-setter* module.

c) Popularity: The popularity of a pipe is measured in the number of clones; a clone is created when a user creates an exact copy a pipe in the repository for their own purposes. This copy can then be saved and modified by the user, allowing them to reuse their own work or the work of others. The number of clones is reported for each pipe in the repository. Among the sample we studied, over 54% of the pipes had been cloned at least once.

d) Diversity: Due to the size and the limited expressiveness of the Yahoo! Pipes language, we conjectured that there was much similarity among the artifacts in the repository. To assess this conjecture, we defined an ordinal diversity metric to measure the types of differences (i.e., uniqueness) among the artifacts and determine how much novelty exists in the repository as a whole. The diversity metric has eight levels (1 ... 8) to describe differences between any two pipes in the repository, summarized in Table I. Given two pipes, p_1 and p_2, a low diversity level indicates that p_1 and p_2 are very similar (i.e., there are few differences between the pipes). Higher levels of diversity indicate that p_1 and p_2 are less similar and thus more unique.

Level 1 represents pipes that have exactly the same behavior as another pipe in the population, possibly resulting from a clone. Level 2 represents pipes with the same structure in terms of modules, the number of fields per module, and connections, but the parameter values can change, whereas Level 3 represents pipes with the same structure, relaxing all parameter values and counts. Level 4 relaxes the connections between the modules but requires that the module bags (module names and frequencies) are the same, and Level 5 relaxes the frequencies and considers only the set of modules. Level 6 considers the bag of modules based on types (i.e., *generator*, *setter*, *path-altering*, and *operator* [17]), Level 7 considers only the number of modules, and Level 8 is a catch-all for all pipes not clustered in an earlier level (truly unique pipes). The goal was to create a diversity gradient where the lower levels apply to pipes that are very similar, and the higher levels to pipes that are very diverse, with the assumption that differences in field values are less impactful than differences in topology. In summary, levels 1, 2, and 3 consider changes to fields but keep the structures the same. Levels 4 and 5 consider changes to the connections between modules, but utilize the same language features (modules). Levels 6, 7, and 8 represent pipes that are quite different.

V. ANALYSIS OF COMMUNITY

In this section, we explore how much and how often authors contribute to the repository, and the characteristics of the contributed pipes.

RQ1a: Author Attrition

From the sample of 32,887 pipes, we found they were created by 20,313 distinct authors. Most authors do not stay active in the community for very long, where activity is measured by the difference between the earliest and latest creation dates on the pipes they contributed. Approximately 82% of the authors were active for only one day, and only 13 authors were active for more than three years (maximum was 1,253

[1]To facilitate replication, the data used in this analysis is available online: *http://cse.unl.edu/∼kstolee/esem2011/artifacts.html*

TABLE III
AUTHOR CONTRIBUTIONS (IN # OF PIPES)

# of Pipes	# Authors	% Authors
1	15,420	75.91%
2	2761	13.59%
3	930	4.57%
4 to 5	642	3.16%
6 to 10	381	1.87%
11 to 15	98	0.48%
16 or more	81	0.39%
all	20,313	100.00%

days), as shown in Table II. The *Duration* column indicates the length of time an author was active and the *# Authors* column indicates how many authors were active for this time duration. As shown, the Yahoo! Pipes community suffers from attrition levels similar to other online communities [22].

RQ1b: Author Contributions

Contributions are measured in number of pipes, and the average author contributes 1.62 pipes. Among the authors, 15,420 (76%) submitted only one pipe, as shown in Table III. This accounts for 47% of the pipes in the sample. The remaining 24% of the authors are responsible for over 53% of the artifacts in the sampling, following a skewed distribution with a long tail; the most prolific author created 98 pipes.

RQ1c: Artifact Characteristics

We explore the overall characteristics of the artifacts, considering each of the dependent variables: diversity, popularity, configurability, and size.

Diversity: We create clusters among the pipes in the sample given the diversity metric in Table I. When a pipe p matches another pipe at some levels l, we say that p is *clustered* at level l, where l is the minimum of all levels in which a match occurs. If we count the number of pipes that are clustered at level 1, we see that only 1,731 (5.26%) of the pipes out of 32,887 have an exact match elsewhere in the sample, as shown in Table IV. The *Diversity Level* column indicates the level of diversity, the *# Clustered* column indicates the number of pipes that were clustered at the given level, and the *% of Pipes* column identifies the percentage of pipes can be clustered at a given level.

Table IV shows that there is much diversity among the pipes in the repository at low levels of abstraction (only 5% of the pipes are clustered at level 1), but there is not as much diversity at higher levels (nearly 60% of the pipes have a match at level 3, and 89% at level 6). Similar to other repositories of code [26], the Yahoo! Pipes repository is full of duplication at higher levels of abstraction. This high frequency of similarity from a structural perspective may occur because authors can easily copy a pipe for their own usage by cloning; there is little incentive to start from scratch if a user can start with a baseline pipe from another user.

Popularity: We associate a high number of clones with high popularity. Within the sample we studied, the average number of clones per pipe was 5.67 with a median of one clone per pipe. We observe that 17,874 (54.35%) of the pipes had been cloned at least once, and the distribution of clones over pipes is shown in Table V. Approximately 11% of the pipes have more than five clones, so the overall majority have been cloned very few times. This low frequency of cloning may be because authors often cannot find pipes in the repository that suit their needs.

Configurability: The average number of user-setting modules across the pipes in the sample is 0.650, with a maximum of 73 configurable modules. Across all the pipes we studied, 33.81% have at least one configurable module. The distribution of configurable modules over pipes is shown in Table VI. The majority of pipes were not made to be configurable. There may be many reasons for this, such as a lack of understanding of the *user-setter* modules, being unaware of the benefits of generalizability in code, or being unable to configure some modules (e.g., some fields are set using a drop-down box, which cannot be configured at run-time).

Size: The average size across pipes in the community is 8.2 with a median of 6.0 modules per pipe. The distribution of sizes over pipes is shown in Table VII. We observe that more than two-third of the pipes have between three and ten

TABLE IV
CLUSTERING OF 32,887 PIPES. *% Pipes* INDICATES PIPES THAT HAVE AT LEAST ONE MATCH IN THE COMMUNITY GIVEN THE DIVERSITY LEVEL

Diversity Level	# Clustered	% of Pipes
1	1,731	5.26%
2	15,186	46.18%
3	19,319	58.74%
4	20,262	61.61%
5	24,316	73.94%
6	29,346	89.24%
7	32,862	99.93%
8	32,887	100.00%

TABLE V
POPULARITY PER PIPE IN COMMUNITY

Clones	# Pipes	% Pipes
0	15,013	45.65%
1	7,175	21.81%
2	3,290	10.00%
$[3, 5]$	3,766	11.44%
$[6, 10]$	1,632	4.96%
$[11, 50]$	1,529	4.64%
$[51, 9180]$	482	1.46%

TABLE VI
CONFIGURABILITY PER PIPE IN COMMUNITY

Configurable Modules	# Pipes	% Pipes
0	21,768	66.19%
1	6,301	19.15%
2	2,286	6.95%
3	1,590	4.83%
$[4, 73]$	942	2.86%

TABLE VII
SIZE PER PIPE IN COMMUNITY

Modules	# Pipes	% Pipes
$[0, 2]$	2,691	8.18%
$[3, 5]$	11,171	33.97%
$[6, 10]$	11,084	33.70%
$[11, 20]$	6510	19.79%
$[21, 287]$	1,431	4.35%

TABLE VIII
CHARACTERISTICS OF PIPES CONTRIBUTED EARLY OR LATE IN AN AUTHOR'S LIFESPAN IN THE COMMUNITY. $\alpha = 0.01$.

Characteristic	**Early (1)**	**Late (2)**	H_0 :	**p-value**
# of Pipes	27,555	5,332		
Diversity	3.519	4.126	$\mu_1 > \mu_2$	2.200 * 10^{-16}
Popularity	4.984	9.254	$\mu_1 > \mu_2$	2.200 * 10^{-16}
Configurability	0.614	0.838	$\mu_1 > \mu_2$	2.200 * 10^{-16}
Size	7.919	9.587	$\mu_1 > \mu_2$	2.200 * 10^{-16}

TABLE IX
CHARACTERISTICS OF PIPES BY AUTHORS WITH MANY CONTRIBUTIONS AND AUTHORS WITH FEW CONTRIBUTIONS. $\alpha = 0.01$.

Characteristic	**Few (1)**	**Many (2)**	H_0 :	**p-value**
# of Pipes	30,503	2,384		
Diversity	3.639	3.355	$\mu_1 > \mu_2$	1.000
Popularity	4.302	23.250	$\mu_1 > \mu_2$	2.200 * 10^{-16}
Configurability	0.644	0.729	$\mu_1 > \mu_2$	2.114 * 10^{-11}
Size	8.194	8.136	$\mu_1 > \mu_2$	0.001799

modules, but that there is a long tail on the distribution where the largest pipe has 287 modules. This shows a large range in the complexity and size of pipes created by the community, indicating a range of skill levels and investment by the authors.

VI. Analysis of Artifacts and Author Experience

In this section, we explore differences among pipes that have been created by authors with different levels of experience. We measure experience along two dimensions: the number of days of experience an author had when a pipe was created, and the total number of contributions by an author. Here, we explore differences in the community artifacts by segmenting along these lines, addressing each subpart of *RQ2*.

RQ2a: Contributions Based on Experience (time)

From Table II, we see that approximately 10% of the authors submitted pipes one month or later after submitting their first pipe. With this threshold in mind, we want to see if there are differences in the contributions made early in an author's experience (i.e., within the first month) versus late in their experience (i.e., after the first month). One month seemed reasonable time period for authors to gain sufficient experience with the environment; the results are shown in Table VIII.

For all the dependent variables, diversity, popularity, configurability, and size, one-tailed Mann-Whitney tests where $H_0 : \mu_{early} > \mu_{late}$ and $\alpha = 0.01$ reveal significant differences between the sample means. We therefore reject the null hypothesis and show that the sample means for all dependent variables are smaller for the pipes submitted within the first month versus after the first month of author experience. Therefore, experience seems to play a role in increased diversity, popularity, configurability and size of contributed pipes.

RQ2b: Comparisons Between Contribution Levels

In Table III, we see that less than 0.5% of the authors created more than 15 pipes in the sampling of the repository. With this threshold in mind, we segment the pipes into two groups, those created by prolific authors who contributed more than 15 pipes, and those created by less prolific authors. The results are shown in Table IX.

For three of the dependent variables, popularity, configurability, and size, one-tailed Mann-Whitney tests where $H_0 : \mu_{few} > \mu_{many}$ and $\alpha = 0.01$ reveal significant differences, causing us to reject the null hypotheses. This indicates that authors who create more pipes have more clones, make their pipes more configurable, and make their pipes larger. Note that for size we reject the null hypothesis even though the means support it; after further inspection we confirmed that this is correct as the mean numbers were caused by a handful of pipes in the "few group" with more than 200 modules that account for its large mean value. For diversity, the null hypothesis is *not* rejected. This is likely because, within the most prolific authors, some only submit pipes that are very similar to others they have submitted in the past, a phenomenon we will explore further in Section VII.

VII. Analysis of the Most Prolific Authors

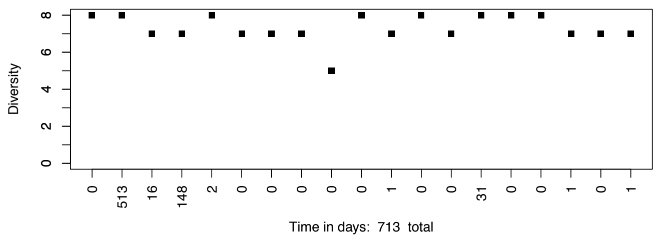
In this analysis, we concentrate on the individual authors and the uniqueness of their contributions, addressing each subpart of *RQ3*. We chose to consider the most prolific authors since their contributions have a greater impact on the repository. To identify the most prolific authors, we selected authors who had contributed more than 15 artifacts to the repository. This threshold balanced our need to do individual author analysis while having enough samples to generalize across prolific authors. In total, we studied 81 authors ($< 1\%$ of the authors in the study), who contributed 2,384 pipes ($\sim$7% of the pipes in the study).

We have identified three categories of interest for characterizing the participants and their contributions: author activities, author evolution, and author awareness.

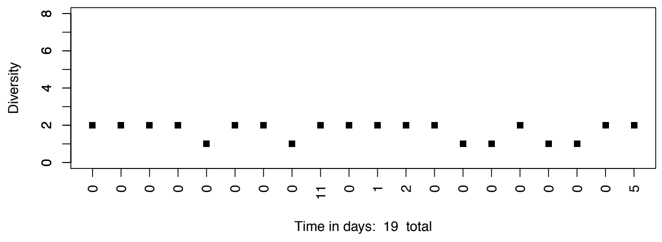
RQ3a: Author Activities

Each pipe submitted by an author represents an activity the author is performing, and the level of diversity of one pipe compared to those created previously by that same author gives an indication of the goal the user had when creating the pipe.

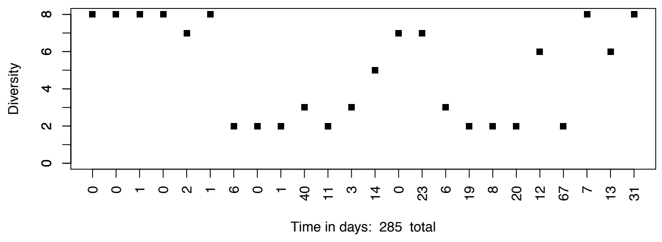
To identify such activities, we first perform a rolling cluster analysis over time of the pipes contributed per author. That is, we identify the level at which each pipe is clustered as it is added to the set of pipes created by an author. This produces a graph, like that shown in Figure 2(a). Time on the x-axis represents the number of days since the most recent pipe was submitted, and the diversity levels are on the y-axis. More concretely, the left-most dot represents the level at which the second pipe was clustered, compared to the first. The second left-most dot represents the level at which the third pipe was clustered, when considering the first two pipes. Thus, each subsequent pipe is compared to all those that came before it. For example, in Figure 2(a), we see the third dot at diversity level 7, with an x-axis label of 16. This means that when the fourth pipe was created, it was clustered at a level 7 compared to those that came before it and was created 16 days after the third pipe.



(a) Pipes Clustered Over Time for One Author, High Skills, Mean = 7.32



(b) Pipes Clustered Over Time for One Author, Low Skills, Mean = 1.70



(c) Pipes Clustered Over Time for One Author, Variable Skills, Mean = 4.96

Fig. 2. Rolling Cluster Analysis Examples. The y-axis shows diversity levels from Table I. The x-axis represents the number days since the creation of the previous pipe.

On average, over one-quarter of the pipes created by a prolific author are highly unique (level 8) compared to the author's previous contributions. The percentage of pipes clustered within author at each diversity level and averaged across authors are shown in Table X. On average, 32.79% of an author's pipes are clustered at level 2, the most common level.

Clearly from the results, authors tend to submit pipes that are either very similar (levels 1, 2 and 3), or very different (levels 6, 7 and 8) to what they submitted in the past. Based on this observation and further pipe examination, we were able to map these results to two typical author activities. First, project **initiation** refers to pipes that are drastically different from those pipes an author had created previously. It is likely that the newly submitted pipe has a different purpose or goal from the previous ones. Second, pipe **tweaking** refers to pipes that are quite similar to those pipes created previously. It is likely that the author created something very similar to a pipe they submitted in the past (e.g., an author changes the filter criteria for the home search in Figure 1, giving a diversity level of 2 or 3, depending on the change).

We see that for the average author, 52% of the pipes created represent new initiatives, while 43% represent tweaks.

TABLE X
PERCENTAGE OF PIPES ADDED AT EACH DIVERSITY LEVEL, AVERAGED ACROSS AUTHORS

Level	Avg. % of Pipes
1	1.63%
2	32.79%
3	8.40%
4	1.26%
5	4.35%
6	5.64%
7	19.36%
8	26.56%

TABLE XI
AUTHORS CLASSIFIED BY SKILL LEVEL

Cluster Average	Skill Level	# Authors	% Authors
$(5, 8]$	High	45	55.56%
$(3, 5]$	Variable	14	17.50%
$[1, 3]$	Low	22	27.16%

RQ3b: Author Evolution

As authors gain more experience with the Yahoo! Pipes language, it was expected that they will become more able to regularly create unique pipes, demonstrating increased skills and providing more value to the community. To investigate this conjecture, we perform two analyses. The first estimates a skill level of the author based on their ability to regularly create unique pipes compared to their previous contributions. The second measures the value of author contributions by correlating experience in terms of days with the uniqueness of their pipes compared to other pipes in the community, with the assumption that more unique contributions are more valuable to the community.

For the skills analysis, we need to gauge the skill levels of the authors. To do so, we use the rolling cluster analysis described for *RQ3a* and calculate the average cluster level for each author to represent the average uniqueness of each pipe an author submitted to the repository, when compared to what they had previously submitted. A high average, like that illustrated in Figure 2(a), indicates an author who regularly submitted distinct pipes, and can be considered a *high skills* author. A low average, like that shown in Figure 2(b), indicates an author who regularly submitted pipes very similar to those of the past, representing an author with *low skills*. A medium average, like that shown in Figure 2(c), indicates an author who submitted pipes with varying uniqueness and has *variable skills*.

Table XI shows the percentages of authors that fall into each of the skill categories, where the first column indicates the range of average cluster values that map to each skill level. Approximately half of the authors are highly skilled submitting pipes that are distinct to the previous ones they submitted. Over 27% of the authors have low skill levels. These authors tend to submit pipes that are very similar to other pipes they have submitted, in essence using the public repository as their own personal repository. They are not able to identify the unique pipes among those they have created and their contributions clutter the repository. The remaining 18% of the authors have variable skills. These authors contribute

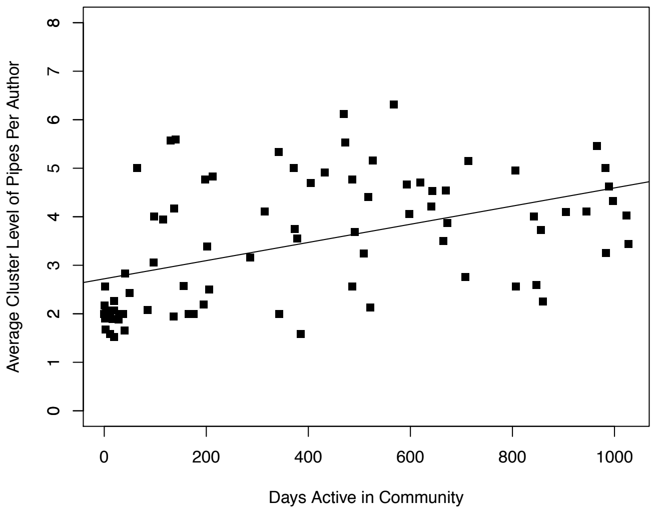


Fig. 3. Author Pipes Average Cluster Levels vs. Days Active for the Most Prolific Authors

a variety of pipes, some that are unique, and some that are similar to what they have previously submitted, but do not exhibit a clear pattern high or low.

For the value analysis, we assume that more unique pipes are more valuable to the community, and investigate if prolific authors with more experience create more valuable pipes. To do this, we measure the number of days of experience the author had when each pipe was created, and correlate that with diversity against the community. There is a positive correlation (Spearman's $r = 0.42136$), so it appears that as the most prolific authors gain experience, the pipes they create tend to be more unique (and to a certain extend more configurable as well with Spearman's $r = 0.30866$). This could indicate that over time, their contributions to the community become more valuable. We explore this relationship further by collapsing across authors and plotting the total number of days an author was active with the average diversity level of all pipes they submitted, shown in Figure 3. Among the prolific authors, there is a clear upward trend between experience and diversity to the community (Spearman's $r = 0.53694$).

RQ3c: Author Awareness

Authors have different levels of awareness about what they submit and what is available in the public repository. As authors contribute more artifacts to the community, it is expected that their awareness of the community will increase. We study this awareness using the uniqueness of each pipe compared to *the author's previous contributions* (local) and *the pipes in the community* (community).

Using the skill levels, we classify the uniqueness for each pipe as high, medium, or low. For each pipe contributed by an author, we look at how the local uniqueness differs from the community uniqueness, and then draw conclusions about their awareness. We observe that authors submit some pipes that are very similar to other pipes they have already submitted (i.e., low local uniqueness, which implies low community uniqueness), so we say these authors have *no awareness*. Other authors submit some pipes that are very unique compared to what they had done in the past, but very similar to other pipes in the community (i.e., local uniqueness is strictly greater than community uniqueness); these authors have *local awareness*. Last, there are authors who submit pipes that are very unique compared to what they had done in the past and also very unique compared to the community (i.e., high local and community uniqueness, or medium local and community uniqueness); these authors have *community awareness*.

TABLE XII
AUTHOR AVERAGE SUBMISSIONS CLASSIFIED BY AWARENESS

Awareness	Avg. % of Pipes
None	50.06%
Local	19.20%
Community	30.74%

The average community uniqueness level for the pipes created by the most prolific authors was 3.35, and the average local uniqueness level was 4.39. On average, 50% of the pipes submitted per author represented no awareness, 20% represented local awareness, and 31% represented high awareness, as shown in Table XII.

VIII. Discussion

From the analysis, we have made several general observations about the Yahoo! Pipes community, and these have led to some implications on how to better support the community.

A. Observations

Few authors are responsible for most artifacts. Like with other online communities, Yahoo! Pipes suffers from attrition and the contributions of the participants follow a skewed distribution with a long tail. Most of the participants ($> 81\%$) are active for only one day, and only 24% of the participants are responsible for over 53% of the artifacts in the repository.

Authors evolve over time. Pipes created early in authors' careers are significantly less diverse, popular, configurable, and large. This is shown by the significant differences between the groups of pipes when controlling for experience (Table VIII) and by the positive correlation between average clustered level with the community and days active in the community for the most prolific authors (Figure 3). Additionally, author skill level is strongly correlated with the total time an author is active in the community (Spearman's $r = 0.61111$). These are positive observations for the community, showing that authors are able to grow over time.

Authors have varying levels of community awareness. The trend of community awareness among all authors is unclear, as few exact matches in the repository indicates high awareness, but little diversity at higher levels of abstraction indicates low awareness. The low frequency of exact clones found in the repository ($\sim 5\%$, Table IV) would suggest that authors may have some understanding of community and the value of contributing pipes that are somewhat different. Further, most pipes have been cloned. This means that if authors clone a pipe but do not modify it, they delete it or do not share it, possibly indicating some level of community awareness. However, despite the evidence of awareness at low levels of abstraction, there remains much similarity among the artifacts at higher levels of abstraction. We find that 59% of pipes are structurally similar to another in the repository (Table IV), indicating much duplication in the repository and potentially little community awareness among all authors.

Among the most prolific authors, most demonstrate little awareness, but a small minority demonstrate higher awareness. About 43% of the pipes submitted by the most prolific authors represent tweaks of something they have created in the past and 50% of the most prolific authors tend to submit pipes that have the same structure (if not also the same fields) as other pipes they had submitted in the past (Table XII), suggesting that even the most prolific authors do not have much awareness. Yet for a minority of the most prolific authors (30%), the pipes they create are different from what they have created in the past and different from what already exists in the community (Table XII). Further study is needed to understand what features differentiate authors with high awareness of the community from authors with low awareness.

Experienced authors make more configurable pipes. A third of all pipes across the community are configurable (Table VI), and authors with more experience (Table VIII) and who create more pipes (Table IX) tend to make pipes that are significantly more configurable. This indicates that authors with more experience may have a greater interest in serving themselves or the community through their more configurable contributions.

Community participants seem interested in utilizing the community knowledge. Most pipes have been cloned at least once, with a small minority of pipes having hundreds, if not thousands, of clones (Table V). This indicates that many participants are likely interested in building on the expertise of others.

B. Implications

Authors may need artifact maintenance support. Over one quarter (27%) of the prolific authors were seen to have low skills (Table XI), and over 43% of the author submissions represented tweaks on already-submitted pipes. This indicates that authors may be using the public repository as a private repository to store incremental changes on pipes, and they may benefit from a versioning system. It may also be that authors are unable to configure their pipes or do not know about the configuration options, and so they are forced to create pipes with few deviations from existing pipes. To assess these conjectures, further analysis is needed to see if authors have patterns of progressive "adding" over time. That is, they make several pipes, $p_1 \ldots p_n$ where $content(p_i) \subseteq content(p_{i+1})$, and content is measured in terms of fields, modules, and connections.

The repository may need moderators. We observed that pipes created by prolific authors are significantly more popular and configurable (Table IX). However, with the number of authors who are only active for a short period of time (Table II), the repository gets cluttered with highly similar and less configurable pipes. An alternative approach might be to notify authors of highly similar, existing pipes. Considering that 70% of the pipes submitted demonstrate no awareness or local awareness (Table XII), a mechanism to alert authors when they are re-inventing the wheel might reduce the clutter in the repository and support end users in becoming more efficient.

Authors may need better search for the repository. Only 5% of the pipes have an exact clone elsewhere in the community, and over 46% have an exact match except for the field values (Table IV). It may be that the high frequency of similarity is a result of an author's inability to find an appropriate pipe from the clutter in the repository.

Authors may need help understanding pipe behavior. Since there is no correlation between popularity and uniqueness (Spearman's $r = 0.06088$), it is likely that either other authors are unable to understand the more unique pipes, authors cannot find what they need in the repository, or that the highly unique pipes solve a very narrow problem. If the pipes cannot be understood, authors may re-invent the wheel and create their own pipe with different syntax yet the same semantics. Perhaps there needs to be better support for helping authors understand the semantics of pipes created by others. One way to go about this would be through the inclusion of a *comments* module so authors can annotate their code, or through automatically-generated documentation.

Authors may need better development support. Given that approximately 50% the authors are unable to consistently produce unique pipes (Table XI), pipes are not very configurable (33%, Table VI), and most pipes contain unfavorable characteristics (as found in previous studies [17]), there is an implied need for better compositional support to allow authors to create higher quality, more diverse and general pipes. Considering also the high frequency of tweaking (43% of pipes) that is performed by the authors and the high frequency of pipes with similar structures (59%, Table IV), authors would likely benefit from support in composition and design.

IX. Threats to Validity

Here we discuss the threats to validity for our study.

A. External

In this study, we consider only one domain, that of Yahoo! Pipes. Our results may not generalize to other end-user repositories, but we attempted to structure our analysis in such a way that new diversity metrics and notions of popularity, size, and configurability could be defined for new domains and then the results compared to ours.

The sample of pipes we scraped are those that were returned by the Yahoo! search results. Since we do not have control over the search mechanism, these pipes may not be representative of the population. To reduce this threat, we obtained a large sample for analysis.

Along these same lines, the observations on individual authors only considered the most prolific authors and the pipes they submitted to the public repository that happen to be within our sample. This does not account for all the pipes created by these authors, nor all the pipes they submitted to the repository.

B. Construct

In our diversity analysis, we consider only structural similarity, not semantic similarity. Pipes with the same structure may not be similar at all. Future work is needed to control for this threat. One avenue would be to refactor the pipes to remove some structural variability and re-cluster, or to measure the diversity of data sources as an indication of semantics.

C. Internal

The most clear internal threat is that all the analysis was done through the lens of a public repository, which offers limited visibility.

Some threats arise based on our selection criteria for the artifacts. For the author analysis, we selected prolific authors who had more than 15 pipes submitted within our sample. It is possible that this threshold is not the optimum to identify prolific authors. The clone counts were gathered at the time each pipe was scraped from the repository, and so the clone values among the pipes were often collected on different days.

We use the diversity of pipes to draw conclusions about the skill levels of the authors, with the assumption that the creation of more diverse pipes is an indication of higher skills. However, we do not measure how functional the submitted pipes are, nor can we tell the provenance of any pipe and so the complex structures may not have originated with the submitting author.

Another internal threat concerns the correctness of the tools we have developed, including the infrastructure to obtain and analyze the artifacts. While we have developed unit tests for all analyses and manually verified anomalies and interesting points in the data, the threat remains.

X. Conclusion

In this work, we present an artifact-based analysis of an end-user community, observing how authors grow with time and the impact of different variables such as time and proliferation on the diversity, popularity, size, and configurability of artifacts in the Yahoo! Pipes public repository. Similar to other communities, there is high attrition and the contributions of the participants follow a skewed distribution where few of the authors are responsible for a majority of the artifacts. However, authors who stay active with the community seem to evolve. We have observed that more experienced authors tend to make pipes that are more diverse, popular, large, and configurable than authors with less experience, and that only 30% of the most prolific authors are able to regularly submit pipes that are highly unique compared to the community. From the results of our analysis, we have identified several implications for how end users could be better supported as they interact with the Yahoo! Pipes language and community, with the hope that some of these findings can be extended to other end-user communities.

Acknowledgments

This material is based upon work supported by the NSF Graduate Research Fellowship under CFDA#47.076, NSF Award #0915526, and AFOSR Awards #9550-10-1-0406 and #9550-09-1-0129.

References

[1] C. Scaffidi, M. Shaw, and B. Myers, "Estimating the numbers of end users and end user programmers," in *Symposium on Visual Languages and Human Centric Computing*, 2005.

[2] R. Panko, "What we know about spreadsheet errors," *Journal of End User Computing*, vol. 10, pp. 15–21, 1998.

[3] "Yahoo! Pipes," http://pipes.yahoo.com/, February 2011.

[4] "CoScripter," http://coscripter.researchlabs.ibm.com/, February 2011.

[5] "Userscripts," http://userscripts.org/, February 2011.

[6] "Scratch," http://scratch.mit.edu/, February 2011.

[7] M. C. Jones and E. F. Churchill, "Conversations in Developer Communities: A Preliminary Analysis of the Yahoo! Pipes Community," in *International Conference on Communities and Technologies*, 2009.

[8] W. Scacchi, "Free/open source software development: Recent research results and methods," *Advances in Computers*, vol. 69, pp. 243–295, 2007.

[9] G. Von Krogh, S. Spaeth, and K. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research Policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[10] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open source software development: What we know and what we do not know," *ACM Computing Surveys*, 2010.

[11] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work*, vol. 14, pp. 323–368, 2005.

[12] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw, and S. Wiedenbeck, "The state of the art in end-user software engineering," *ACM Computing Survey*, 2011.

[13] "IBM Mashup Center," http://www.ibm.com/software/info/mashup-center/, August 2009.

[14] S. K. Kuttal, A. Sarma, A. Swearngin, and G. Rothermel, "Versioning for mashups an exploratory study," in *International Symposium on End-User Development*, 2011.

[15] A. J. Ko and B. A. Myers, "Debugging Reinvented: Asking and Answering Why and Why Not Questions About Program Behavior," in *International Conference on Software Engineering*, 2008.

[16] A. Koesnandar, S. Elbaum, G. Rothermel, L. Hochstein, C. Scaffidi, and K. T. Stolee, "Using assertions to help end-user programmers create dependable web macros," in *International Symposium on Foundations of Software Engineering*, 2008.

[17] K. T. Stolee and S. Elbaum, "Refactoring pipe-like mashups for end-user programmers," in *International Conference on Software Engineering*, 2011.

[18] C. Scaffidi, C. Bogart, M. Burnett, A. Cypher, B. Myers, and M. Shaw, "Predicting reuse of end-user web macro scripts," in *Symposium on Visual Languages and Human-Centric Computing*, 2009.

[19] A. Dahotre, Y. Zhang, and C. Scaffidi, "A qualitative study of animation programming in the wild," in *International Symposium on Empirical Software Engineering and Measurement*, 2010.

[20] G. Fischer, A. Piccinno, and Y. Ye, "The ecology of participants in co-evolving socio-technical environments," in *Conference on Human-Centered Software Engineering*, 2008.

[21] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *International Symposium on Foundations of software engineering*, 2008.

[22] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon: An analysis based on social network theory," in *Americas Conference on Information Systems*, 2002.

[23] W. Oh and S. Jeon, "Membership herding and network stability in the open source community: The Ising perspective," *Management science*, vol. 53, no. 7, p. 1086, 2007.

[24] J. Wong and J. Hong, "What Do We "Mashup" When We Make Mashups?" in *International Workshop on End-User Software Engineering*, 2008.

[25] "JSON," http://www.json.org/, August 2009.

[26] M. Gabel and Z. Su, "A study of the uniqueness of source code," in *International symposium on Foundations of software engineering*, 2010.