Conversations in Developer Communities: a Preliminary Analysis of the Yahoo! Pipes Community

M. Cameron Jones
Yahoo! Research
4401 Great America Parkway
Santa Clara, CA 95054
mcj@acm.org

Elizabeth F. Churchill
Yahoo! Research
4401 Great America Parkway
Santa Clara, CA 95054
churchill@acm.org

ABSTRACT
In this paper we describe several issues end-users may face when developing web mashup applications in visual language tools like Yahoo! Pipes. We explore how these problems manifest themselves in the conversations users have in the associated discussion forums, and examine the community practices and processes at work in collaborative debugging, and problem solving. We have noticed two valences of engagement in the community: core and peripheral. Core engagement involves active question asking and answering and contribution of example content. Peripheral engagement refers to those who read but don't post, and those who post legitimate questions and content, but whose posts receive no response. We consider what the characteristics are of each of these groups, why there is such a strong divide, and how the periphery functions in the community process.

Categories and Subject Descriptors
D.3.0. [**Programming Languages**]: General

General Terms
Design, Documentation, Languages.

Keywords
Web mashups, developer communities, end-user programming, conversations, code, participation, question-answer forums.

1. INTRODUCTION
Web mashups are an ideal context in which to observe and analyze end-user programming activity, including the usage of visual languages and tools. By studying the interactions among members of ad-hoc developer communities we can begin to understand not only how independent developers support each other in learning about, and debugging programs, but also what the role of community participation is in such contexts. The mashup ecosystem is populated with many tools and languages designed to support mashup development; groups of developers congregate online around these tools and services in order to give and receive help, and to thus show and develop expertise. The level of organization and formality in these communities varies, and activities are dispersed across a number of media including: discussion forums, video tutorials on video sharing sites, text tutorials on personal blogs, chat conversations in IRC and/or instant messaging channels, code snippets and annotations in blogs and code snippet sharing sites, and so on. These electronic communications constitute conversations around code, and serve as a primary resource for novice, and expert programmers. They are essential to navigating the loosely connected space of heterogeneous services, tools, and resources that comprise the mashup ecosystem.

By studying the conversations surrounding the development of mashup applications, we can infer barriers which impede the development of mashup applications, and which also reflect on the general challenges imposed on application development by the nature of the mashup ecosystem. We can also see how the community works to rectify these problems, and provides technical support and debugging assistance for working around bugs in applications and mashup development platforms. In this paper we present the findings of a preliminary, qualitative analysis of conversations and discussions surrounding the development of mashup applications in the Yahoo! Pipes environment.

1.1 Social aspects of code development
Social interaction in software development has typically been studied in the context of collaborative software development in teams and organizations (e.g., [7], [8], [15], [13]), presumably because this was where collaborative programming could be observed. Increasingly, however, software development is happening outside of and between organizations. These alternative programming contexts include things like open source software projects, and hobbyist development. The social structures of open source communities have garnered research interest (e.g., [4]), and more recently, attention is turning towards the social and collaborative practices of hobbyist programmers on the web (e.g., [14], [20]).

Crowston & Howison [4] found that open-source development communities range in the degree to which there is centralized control over the code base. Strongly centralized networks were indicated by star-shaped networks, with a central hub and many developers connected only to that hub, and few connections to others. Decentralized networks lack a single, central authority, having several hubs, and resemble a thicket of inter-connections. Furthermore, Crowston & Howison found that as project sizes increase, projects tend to be less centrally controlled, most likely because the complexity of the project exceeds that which can be managed by an individual.

In studies of collaborative software development, there is a common programming task or project that defines the group being studied; all participants are working towards a common goal. However, the web is playing a more significant role in individual programming, connecting independent developers to each other and to other entities like large software projects. For example,

C&T'09, June 25–27, 2009, University Park, Pennsylvania, USA.
ACM 978-1-60558-601-4/09/06.

developer networks like the Yahoo! Developer Network (YDN) bring together programmers utilizing Yahoo! products, services, and data sources in online discussion forums and mailing lists. Looser networks are scattered across the web in the form of coding blogs (e.g., http://alistapart.com/ for CSS developers, or http://quirksmode.com/ for JavaScript), programming discussion forums (e.g., http://php-forum.com/), and code sharing sites (e.g., http://snipplr.com/).

These groups and communities are similar to those of traditional collaborative software development, in that there is a common programming context (e.g., tools, languages, environments, goals, etc), but are different in that there is not a common project, a common goal or a common organizational context of production; each developer is independently seeking individual objectives but interacting with others to achieve those goals. This reflects a shift from an understanding of software development as primarily an individual or group effort towards a more open, social approach to software development and debugging.

The web is serving as the primary medium for social engagement around software. Several studies have examined the role of the web, and web resources in the software development. Brandt, et al. [1] describe "opportunistic programming", where the ready availability of source code, tutorials, and examples on the web make for easy programming by copy-paste, allowing developers to compose applications as they opportunistically encounter code and coding resources. Stylos and Myers [17] have built search tools for facilitating the finding and reuse of coding information on the web. Programming, outside explicitly collaborative contexts, may at first appear to be a solitary act, is actually rooted in a complex social web of building off of, and with others' code found online.

The community of developers surrounding the Yahoo! Pipes environment foregrounds the role of conversational interaction in collaborative problem solving. In order to understand the nature of the programming challenges faced by Yahoo! Pipes developers, we briefly discuss the challenges of mashup programming and the Yahoo! Pipes platform. Following this, we introduce the Yahoo! Pipes developer community and discuss several conversations taken from the Pipes developer forums. These examples highlight, not only the particular barriers which developers face, but reflect on the role of the community process in support and problem solving.

1.2 Why community help matters for mashups

Mashup programming "in the wild" as represented by many of the applications listed on ProgrammableWeb.com is a complex and informal programming ecosystem. Unlike the orderly, monumental, cathedral-like arrangement of classes and resources in languages like SmallTalk or Java, mashup programming is a chaotic bazaar of offerings. The mix of heterogeneous services, each with their own application programming interfaces (APIs), data types and structures, programming models and patterns, quickly becomes unmanageable.

Within a conventional designed programming environment, care has been taken through the refactoring of elements into code libraries to achieve compatibility between data structures and function calls, as well as a degree of semantic and syntactic consistency in how these are provided. In the mashup eco-system, however, there is no authoritative control, or oversight; no architect designing things on a global scale.

Stylos & Myers [18] have outlined the many design decisions that go into the formalization of an API, and although they were not studying web service interfaces in particular, their findings highlight the complexity of the process. Often, the design of APIs externalizes the models, process, and inner-workings of the system, which does not always correspond to the ways in which external developers (i.e., users of the API) want to use the API or think about the problem. This makes learning and using one API challenging enough – but in the mashup context when interfacing with multiple APIs, created and maintained by different organizations or developers, there is the additional challenge of mapping between the possibly conflicting models and abstractions.

Jones, Churchill, and Twidale [9], have taken the cognitive dimensions framework (c.f., [6]), and applied it to understanding the complexities of mashup development. They discuss the difficulties which arise from: conflicting levels of abstraction in APIs; low consistency in the ecosystem in general; the many hidden dependencies inherent in mashup development; the inability to effectively drill down beyond the black-box of the API; and the often necessary mental acrobatics developers must perform in order to achieve an end result. They also describe additional concerns salient to the mashup context which complicate development: the relatively low stability of service APIs and the mashup ecosystem over time, as services and technologies change and evolve; the unreliability of services, and data in mashups, which often are provided with no service-level agreements or guarantees of accuracy, availability, or consistency; and the relative difficulty with which knowledge is sharable and transferrable in mashup development.

In their survey of web-active end-users, Zang & Rosson [20] asked participants to describe how a mashup is made, the majority of the respondents had trouble breaking down the mashup into even a basic three-step process of: collect, transform, display. This resonates with the findings that similar hurdles are encountered when teaching mashups to novices [5]; learners don't know how to translate the idea they have into a computational model or procedure for developing a mashup.

Given the complexities in mashup development, and the difficulties end-user programmers have in understanding the development process, users are prone to make errors and introduce bugs into their applications. Visual programming tools may help mitigate many types of errors, but they may also introduce of other types of errors, obfuscate the origin of errors, or interfere with effective communication and explanation. In the following sections we will introduce the Yahoo! Pipes mashup development environment, and present several conversations we have observed surrounding the development of mashups in Yahoo! Pipes. These conversations reflect more general challenges for visual language approaches to mashup development.

2. The Context of Study: Yahoo! Pipes

Yahoo! Pipes is a web-based visual programming language for constructing data mashups. Yahoo! Pipes was originally developed as a tool to make extracting, aggregating, and republishing data from across the web easier. Since its launch in February 2007, over 90,000 developers have created individual pipes on the Yahoo! Pipes platform, and pipes are executed over 5,000,000 times each day. Figure 1 shows the Yahoo! Pipes editing environment. The environment consists of four main

regions: a navigational bar across the top, the toolbox on the left, the work canvas in the center, and a debug-output panel at the bottom. The toolbox contains modules, the building blocks of the Yahoo! Pipes visual language.

Yahoo! Pipes' namesake is the Unix command-line pipe operator, which allows a user to string together a series of commands, where the output of one is passed as input to the next. In the graphical language of Yahoo! Pipes, modules (operators) are laid out on a design canvas. Modules may have zero or more input ports, and all have at least one output port; additionally, modules may have parameters which can be set by the programmer, or themselves wired into the output of other modules so that the value of the parameter is dependent upon a runtime value specified elsewhere. The input and output ports are wired together, representing the flow of data through the application. Selecting an output port, highlights all the compatible input ports to which the user may connect it.

There are a number of data types within Yahoo! Pipes which determine what inputs and outputs are compatible. In the most general terms, there are simple scalar data *values*, and *items*, which are sets of data objects (e.g., items in an RSS feed, or nodes in an XML document). Values include types like text, urls, locations, numbers, dates, and times.

In Yahoo! Pipes, data flows from the initial module(s), where user-data are input, or external data are retrieved, through subsequent modules in the pattern and order dictated by the wiring diagram. All applications in Yahoo! Pipes have a single output module, which is wired to the end of the execution sequence, and collects the final data stream for distribution via RSS, JSON (JavaScript Object Notation), or a variety of other formats. Drawing on the Unix command-line metaphor, the output module is akin to "standard out" or the user terminal.

Unlike the Unix command-line pipe, Yahoo! Pipes allows users to define complex branching, and looping structures, have multiple sources and execution paths executing in parallel, and in general, create programs of arbitrary complexity. There is no direct method for writing recursive functions, and Yahoo! Pipes does not allow for cycles in the program structure (i.e., where the output of a module is fed back to the input of a module further 'upstream'). This enforces a mostly linear execution flow to the applications which is bounded by the amount of data being processed.

2.1 View Source, Cloning, and Embedding

Each pipe application is individually addressed by a unique ID and URL. Users may publish their pipes in the public directory, where they can be searched, browsed, and viewed by anyone. However, Yahoo! Pipes has a very open security model, allowing any user to view and run any pipe, so long as they know the URL, even if it is not published in the directory. This design was intentional, as the Yahoo! Pipes developers wanted to foster the kind of learning-by-example which Netscape's "View Source" feature made easy in HTML. Thus, every pipe application which is created has a "View Source" button attached to it, allowing users to inspect how a pipe works. This allows users not only to share links to their in-progress, and unpublished pipes, but view and modify each others pipes; allowing users to collaboratively debug problems.

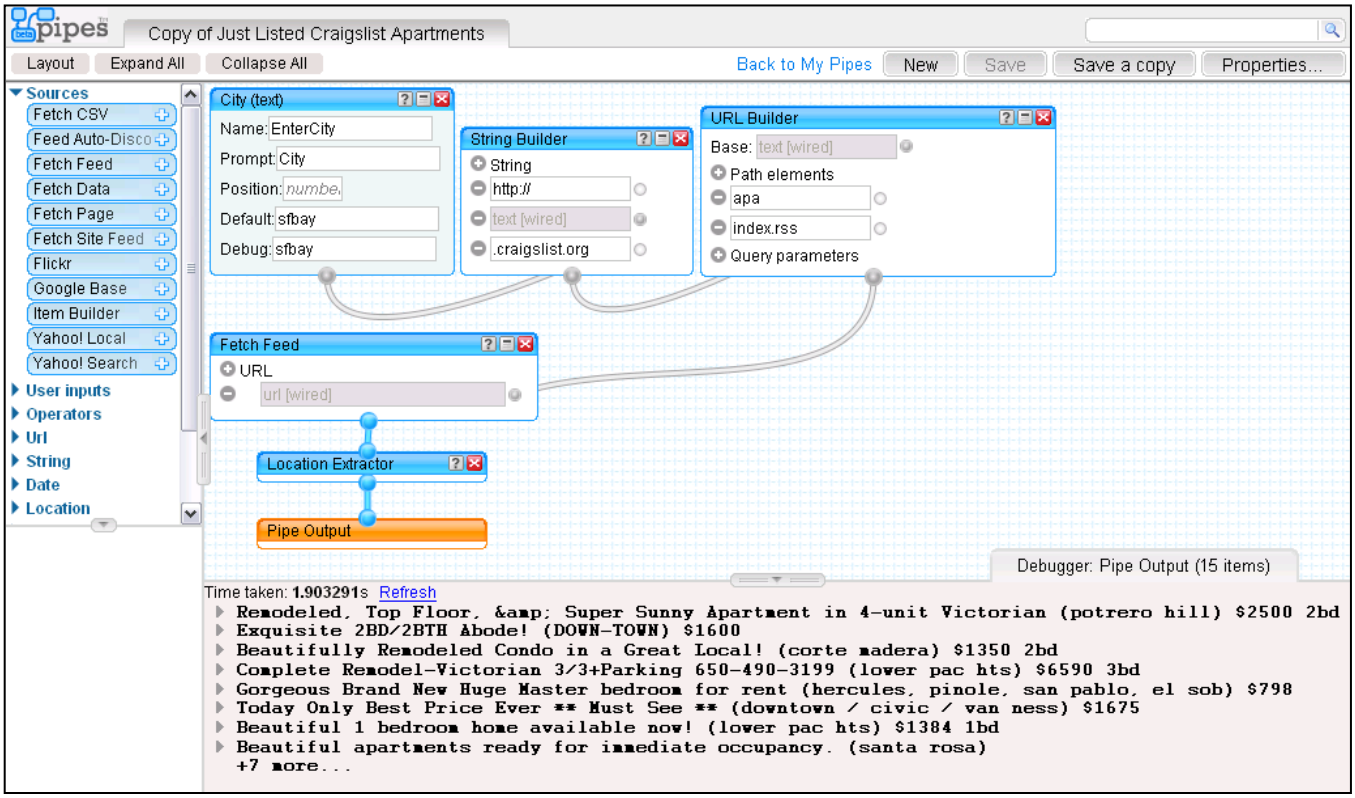

Figure 1. The Yahoo! Pipes editing interface consists of four regions: the editing canvas, the module toolbox, the navigation bar, and the debugger.

Figure 2. The social interaction network of participation in the Yahoo! Pipes discussion forums from December 2008. Participants who appear in the conversational examples provided below have been identified.

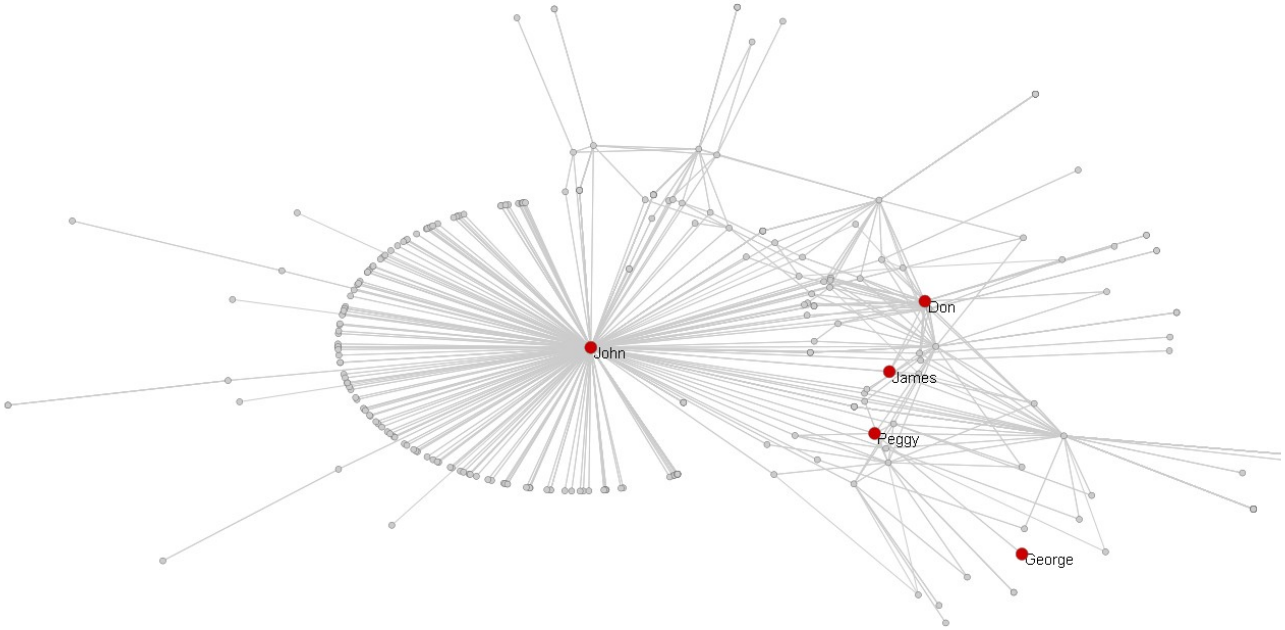

Figure 3. The social interaction network of members of the Yahoo! Pipes discussion forums who have communicated with each other in two or more threads. Participants who appear in the conversational examples provided below have been identified.

When a pipe application is viewed by someone who is not the owner of the pipe, a local copy is made in the viewer's browser. Any changes that are made by the viewer are saved to a new copy on the server, preserving the original pipe. Additionally, each pipe has a "clone" button, which creates a copy of an existing pipe; it is possible to make copies of one's own pipes or of other people' pipes.

In addition to entire pipes being copyable and modifiable, pipes can be embedded within one another as a "sub-pipe". This allows developers to create and share reusable components, and generate intermediate levels of abstraction in their applications. An embedded sub-pipe is represented as a module in the Yahoo! Pipes interface, which can be wired to other modules, or other sub-pipes. Users can drill-down into embedded sub-pipes, to inspect and modify the included functionality.

3. Conversations on Pipes

Our approach to understanding the issues users have with developing Yahoo! Pipes applications has been to start with looking at the conversations users are having about Yahoo! Pipes. We actively monitored and followed the discussions on the Yahoo! Pipes forums since February 2008. Additionally, we took a complete snapshot of the forum contents on 01 December 2008; this snapshot provides a full history of the forums dating back to February 2007.

As of the first of December 2008, there are 2,081 participants in the Pipes forums, participating in 2,548 conversations. In our following of the forum discussions, we were non-systematically reading and annotating posts which were interesting, and taking notes on who was actively participating. For the purposes of this study, we plotted the entirety of the Pipes discussion forums as a social network where the social ties reflect a binary relationship of "co-participating in the same conversation". Figure 2 depicts interaction network of Pipes discussion forum participants. The most salient feature of the network is the division between the core and the periphery. The core represents participants who are interconnected, and the periphery are isolated sub-networks of users who either post and receive no replies from others, or receive replies from other isolated individuals, not connected to the core.

We sampled several key members (i.e., hubs) from the network. We then analyzed all the conversations involving the selected participants for common patterns of interaction and problem solving, paying attention to how problems were characterized, localized, and resolved. We have highlighted, and labeled in Figure 1, all of the developers[1] that participate in the conversational examples included later (note that not all participants we studied have highlighted in the graph, only those which are included in examples).

The Yahoo! Pipes discussion boards (http://discuss.pipes.yahoo.com/) have been active since Pipes was launched in February, 2007. The forums have been a significant source of information on programming in Pipes, as there is not much documentation for the language, merely some tutorials and annotations. We have studied several snapshots of the Pipes discussion forums over the past two years, the most recent snapshot from December 2008.

[1] All participant names have been changed; however, as the gender of participants is not known, no attempt was made to preserve gender when assigning pseudonyms.

The discussions in the Pipes forums are divided into three areas: Developer Help, General Discussion, and a section for showing off Pipes applications. Table 1 provides some general statistics about the activity of the forums as of 01 December 2008.

Table 1. Statistics of the Pipes developer forums through the first of December 2008.

Pipes Forums Activity Data	Developer Help	General Discussion	Show Off Your Pipe
Number of Threads	1731	576	241
Number of unanswered posts	347	165	149
Avg. Thread Length	3.85	3.18	1.69
Std. dev. of Thread Length	4.06	3.26	1.12
Number of participants	1523	638	236
Avg. num. participants per thread	2.34	2.14	1.34
Std. dev. of num participants	2.13	2.06	0.70

Most of the activity on the Pipes discussion forums is question-answer-type interactions in the Developer Help forum, a form of social search. The majority of posts in this forum receive replies of some kind. However, about one in five posts had not received a response at the time of our data collection.

The question-answer-type interactions show up very clearly in both Figure 2 and Figure 3 as evidenced by the clear definition of hubs in the network, e.g., the large fan-shaped structure in Figure 2 (more pronounced in Figure 3), surrounding the user John. Figure 3 is a filtered view of Figure 2, showing only those members who have interacted in two or more conversation threads. John is at the center of a large group of isolated individuals who are not communicating much with each other. In fact, John has actively contributed to over 990 threads in the forums, or nearly 40% of all discussions. John actively responds to new-comers and does a lot of question-answering, and resource marshalling for other developers.

In the following sections we present several examples and snippets of conversations, selected from the participants we selected to study. We focus on conversations involving more than one participant that illuminate the interactions developers have around software debugging and collaborative problem solving.

4. Localizing Bugs

Software debugging is a major activity in the software development process. Debugging consists of identifying, localizing, and correcting/fixing errors in a software application [11]. In debugging errors in Yahoo! Pipes, a common issue we observed in many of the discussions was bug localization, i.e., determining the source or cause of the error. In a standard programming debugging procedure, this usually involves stepping through code, or inserting break points on statements, and watches on data elements in order to determine precisely where the program deviates from the expected behavior. However, many debugging conversations in the Yahoo! Pipes discussion forums are similar to the example provided in Figure 4.

Rob

I have created an images-only RSS feed that basically grabs image such as these: http://www.vrindavandarshan.com/yr2008/aug08/30aug2008_gnf.jpg
The problem is the images are not showing up in-line as "content", rather you have to click through to actually see them. This makes the RSS feed rather useless for displaying inside other modules or devices.
Could someone look at my pipe and let me know what I am doing wrong? All I want is for the output of the RSS feed to simply contain the 3 images of the day. Here's the pipe as far as I have gotten it: http://pipes.yahoo.com/pipes/pipe.info?_ _id=cA3y8aJ23RGhv6G3_g6H4A

John

When I looked at your RSS feed - http://pipes.yahoo.com/pipes/pipe.run?_id=cA3y8aJ23RGhv6G3_g6H4A&_render=rss - the images were displayed. Are you still not seeing the images? If so you will need to supply more details about how you are viewing the RSS feed. It may be that you are experiencing caching issues that should not last more than about 30 minutes.

Rob

I'm trying to use Google Reader, which does not seem to show a "preview" image in-line. I have to click through to each image to see it. Maybe this is just a google reader thing, but I have other feeds (eg, Dilbert web comic) that shows the image right on the page.

John

When using "Expanded view" in Google Reader both the Dilbert feed and you feed show images without any need for clicking. At least, that's what I'm seeing. I can't see any "preview" images for either feed in either List or Expanded view.

Rob

This is very strange, because the feed will not work for me unless I manually load the images by typing the image URL directly into the address bar first. After I do this, the image seems to 'preload' and then the RSS feed works fine. If I don't do this I get a 403 forbidden error from the site.
Is there some standard template that can be used for creating an RSS feed that is simply a group of images?

John

I think we are in the realm of browser/operating system/security settings differences.
All I can say is that the feed works for me in Google Reader using FF3 and W2K.
You may be able to find more help from the Google Reader Help group. http://groups.google.com/group/Google-Reader-Help/

Figure 4. A discussion between Rob and John trying to localize the source of an error.

In the exchange between Rob and John, we can see the progressive peeling away of layers of abstraction and execution. Rob begins with the assumption that there is a problem in his pipe application, that he has caused an error. User John replies stating that the pipe appears to work for him, and does not exhibit the problematic behavior Rob is reporting; he offers the suggestion that the problem may be in the caching behavior of the Yahoo! Pipes platform. Rob responds that he thinks it might be a problem

of the Google Reader RSS viewer application, and not a problem with his pipe or the Yahoo! Pipes cache. After several more exchanges, John asserts that the problem is probably being caused by an incompatible browser or operating system setting.

In this example, we can see that the potential sources of the errors are far more numerous than typically considered in debugging. Typically, when a program does not work as expected, the programmer assumes there is a problem in his/her code, trusting that the underlying compiler/interpreter, operating system, networking stack, etc. are working properly. Rarely would we expect an error in program execution to be caused by a bug in the underlying operating system, for example. However, in Yahoo! Pipes, the underlying infrastructure for interpreting and executing the pipe application may itself have bugs; or the browser or other application in which the pipe is being executed, or the output is being rendered may be incompatible with certain aspects of the Yahoo! Pipes system or data formatting; or there may be a problem with improperly formatted data being fed into the pipe; or some other problem further upstream in one of the data sources. Many of these problems are outside the user's control, making them nearly impossible to resolve.

While this problem may be particularly salient in the Yahoo! Pipes context, given the additional layers introduced by the visual language, we believe these problems to be inherent in mashup programming in general. Web mashups are necessarily embedded in a web of interdependent services, platforms, and data objects, many of which are not as robust or verified as modern compilers, or the underlying operating system stack. While web mashups are often discussed in the context of the "web as operating system", the reality is that the web is not as stable or robust as a standard desktop operating system. It is often that case that services have bugs or fail, network connections are not reliable, and data are not properly formatted (often because the standards are underspecified).

In this example, we can also see the value of the exchange between the two and the work of coming to a shared understanding where a recommendation as to next steps makes sense. Alone neither would have derived the solution and much of the conversation is about explaining what each 'sees' – actually and conceptually.

Jones, Churchill, and Twidale framed these challenges within the cognitive dimensions framework [9]. They argue that the existing cognitive dimensions do not account for the additional complexity and challenges imposed by the open, heterogeneous nature of the mashup ecosystem, and point to the affordances development tools, like Pipes, have for sharing and collaborative debugging as a possible mechanism for cognitive offloading, and effective resolution to complex problems.

5. Marshalling Resources

The Yahoo! Pipes development environment affords sharing and building off of the work of others. This makes it easy for users to point questioners to working solutions, rather than descriptions of how to solve the problem. Often these solutions can be found in existing pipes applications; occasionally they are constructed in response to a specific question. Figure 5 is an exchange between Harry and Don, in which Don points Harry to an existing solution, after Harry has gone to great lengths to articulate his problem.

In this case, it would seem that the pipe being referenced is treated as an informational commodity (c.f., [3]), which has implications for how Harry comes to make sense of the program logic, and apply it to his specific context. Indeed, in this particular thread, Harry replies back stating that he was unable to integrate the solution provided by Don into his problem. Don responds with a pointer to a simpler version of the algorithm, presumably one which is more commoditized and easier to consume.

Harry
Hi, I've just created a feed that contains several ones as input. But I did not manage to extract the source name of each feed. *Here is an example :* *2 input = feedOne ans feedTwo* *1 output = feedOne mixed with feedTwo* *but for each title , I want :* *<title message feedOne> + <name feedOne>* *How can I do this?*
Harry
I've found a way but it is not a pleasant one : after each Fetch Module, I add a regExp module to change manually the title of each post. *I published my pipe to show you : name is "Bourse (pipe Harry)".* *I would have prefered to find a way to extract the name of the source rather than write it by myself.* *Thanks for your help*
Don
http://pipes.yahoo.com/pipes/pipe.info?_id=Qg5X6Rf82xG4xdYjdrq02Q
Harry
That's great, but I can't get your stuff to work with mine. *I ended up having to manually hack this crap, as suggested in the first post on this thread. See "Fresh Gadget News Feed" at http://pipes.yahoo.com/pipes/pipe.info?_id=1PgGpDcE3BGxjDZsEpPZnA*
Don
Why not? Clone/copy manually all my submodules, and rebuild it yourself. It should work. *However, if that is too complicated to your likings, you can use this other pipe:* *http://pipes.yahoo.com/pipes/pipe.info?_id=Uqs_KBf82xG_avvYjknRlg* *It is simpler, but you have to insert the title yourself. Just have to "Save a copy" and then replace each one of your "Fetch"+"Regex" with one of those. The result will be cleaner.*

Figure 5. Harry is having trouble with a pipe, Don points Harry to a solution, which Harry is unable to make work.

As was mentioned before, there is little formal documentation of the Yahoo! Pipes platform. Thus, the discussion forums are a primary source of information on how to do things in the Pipes platform. Figure 6 shows user John helping Mary locate a previous discussion which explains how to solve her problem. John provides some basic explanation for how Mary can solve the problem using a Google Spreadsheet, and points to a thread in which it is explained how to import URLs from a Google spreadsheet application into your pipe; helping contextualize the offered solution.

Mary
For the past day or two I've been trying to save edits to and republish this pipe, but Pipes cannot read any of the source modules anymore in the edit mode. The pipe is inordinately large, but it continues to function and output RSS when any of the sources update. I just cannot save any changes. *Here is the pipe:* *http://pipes.yahoo.com/pipes/pipe.info?_id=6Fh8U37K2xG_TTT_p2IyXQ*
John
You could try using a Google spreadsheet to store your feeds and title annotations. *See this thread:* *http://discuss.pipes.yahoo.com/Message_Boards_for_Pipes/threadview?m=te&bn=pip-DeveloperHelp&tid=1100&mid=1100&tof=36&frt=2*

Figure 6. John helps Mary by pointing her to a previous discussion on the topic.

A further example of how users collaboratively marshal resources can be seen in Figure 7. The exchange depicted in Figure 7 highlights collaborative information retrieval behavior (c.f., [10]), where John and Susan are collecting tutorials for Tim to use. John even goes so far as to cite the sources for the links he is offering, pointing Tim towards new information resources for future reference. Granted, this is a rather simple example of social search behavior, and many of the other examples provided point to more complex conversations through which the information need is articulated and explicated, and hopefully resolved.

Tim
Can anyone show me a simple way to integrate a pipe in your own site? I cannot find that information anywhere.
John
Hopefully these links will help. *http://blog.pipes.yahoo.com/2007/06/12/working-with-pipes-on-your-web-site/* *This link is from the Pipes Blog (http://blog.pipes.yahoo.com/)* *http://www.hunlock.com/blogs/Yahoo_Pipes--RSS_without_Server_Side_Scripts* *This link is from the Pipes del.icio.us pages (http://del.icio.us/rss/pipes.yahoo.com).*
Susan (superspacetyrant)
This worked for me. *http://comments.deasil.com/2007/02/19/pipejax-pure-javascript-version-yahoo-pipes-to-ajax-bridge/*

Figure 7. John and Susan point Tim to several tutorials for embedding Pipe output in a webpage.

6. Peripheral Participation

The partition of the Pipes forum into a core and periphery (see Figure 2) raises questions of what is happening in the periphery of the network, why so many people are disconnected from the rest of the network, and why posts are going unanswered. We have observed several patterns of interaction in the periphery which may explain the divide between it and the core, these include: people asking questions and not getting replies, people replying to themselves, and people getting responses and not returning.

The most obvious reason for why some people are isolated from the rest of the social group is that no one is replying to their posts. The reasons for this may be simple netiquette, the perception of naiveté, or possibly that other users are not experiencing the same problem.

For example, when a user asks, "i'm new to this Pipes. one quick question how can i make rss feed (Pipes) from the forum in yahoo groups even though i'm not signed in?", this question reflects a lack of familiarity with the technology which does not support authenticated requests, and of the community which has extensively discussed adding authentication to Pipes. In another example, a user asks, "I was editing a large and complicated pipe last night - and it started giving me 'Problem saving' errors periodically. At some point I gave up and exited out. This morning I discovered that the entire pipe is EMPTY now. … Anyone else having trouble saving and previewing frequently?" In this case, if no one else is having problems, there is little motivation to respond to this question.

Stan
I am trying to build a web service with pipes and I am unable to get the results parsed correctly upon return. I even took the example servlet code to handle the post action but still get a unable to parse response: {"items": [{ "description": "... response within the editor. *Has anyone else had any luck building a "Web Service" and getting the results parsed.*
Stan
I just answered my own question. The path to item list needed to be plural items not item singular.

Figure 8. User Stan answering his own question.

We also have seen users posting responses to their own questions. It is often the case that a user has resolved the problem they initially posted, although they may also be replying back with additional information. An example of the former is given in Figure 8, where Stan posts a question about a problem he is having and answers it. This kind of self-response is common, happening 126 times in the entire dataset, 66 times in the Developer Help forums. These conversations provide documentation of a problem coupled with a resolution that allows them to be searched and used by other developers who may be in similar situations.

Table 2. As the network size increases over time, the density of edges in the network also increases.

	# Nodes	# Edges	Avg. Degree	Degree Centrality
Mar-07	369	2084	5.65	0.21
Jun-07	697	4320	6.20	0.32
Sep-07	892	5394	6.05	0.26
Dec-07	1090	6984	6.41	0.32
Mar-08	1279	8186	6.40	0.36
Jun-08	1551	10374	6.69	0.40
Sep-08	1809	13401	7.41	0.45
Dec-08	2081	15962	7.67	0.49

The activity in the periphery may be interpreted as "legitimate peripheral participation" [12]. Legitimate peripheral participation describes the evolving role of members of a community of practice, from newcomers, to old-timers. If this is indeed the case, we expect to see a trend of members moving from the periphery towards the core of the network, as their involvement in the community increases with their increased experience. This movement towards greater connectivity would be reflected in an increase in edge density in the graph over time.

Table 2 shows that the average degree of the nodes in the graph does increase over time, indicating that the network is becoming more connected as it grows in size. However, as the network grows, the network is becoming more centrally organized, as indicated by the increasing degree centrality of the network over time. At first, this might appear to counter the findings of Crowston & Howison [4] which showed that larger open-source communities have less centralized control; however, the actual degree centrality (0.49) for the Pipes community as of December 2008, when there were 2081 members of the community, is not inconsistent with some larger community networks Crowston & Howison studied.

The evolution of the Pipes community forum does seem to follow the progression of expertise as described by Lave and Wenger, although the trend towards increased participation is not very strong. Most developers only contribute a limited number of times, and do not sustain engagement in the community. Figure 9 shows the percentage of members of the network who are actively participating new messages to the discussion over time. The percentage who engages in active conversation drops over the past two years. This might be due to the transactional nature of question-answer format discussions, where participants ask a question, get an answer, and leave.

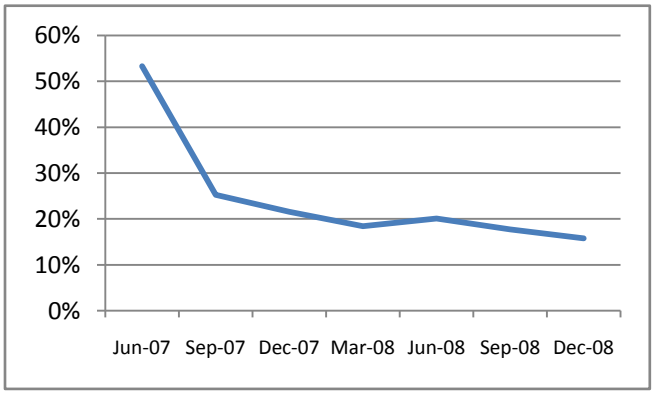

Figure 9. Percentage of community active over time.

It is not clear how to reconcile the observation that the network is becoming more interconnected, but the relative proportion of the community actively engaging in the conversations is dropping. One explanation might be that the community is maturing, achieving a relatively stable core of active experienced members and old-timers. This stable core reaches out to new comers, responding to questions, reinforcing the star-shaped structure of a centralized network. Although the relative percentage of active participants has dropped, the total size of active discussants has remained relatively constant, ranging between 236 and 377 (avg of 291) active participants (see Figure 10). This might indicate a "natural size" for this community in terms of the number of active conversations that can be managed and maintained simultaneously.

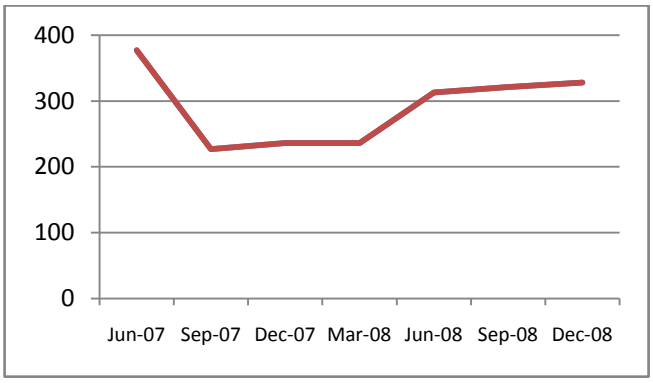

Figure 10. Raw number of active participants over time.

7. Conclusions

This paper has outlined a preliminary analysis of issues commonly discussed among user-developers of the Yahoo! Pipes visual languages. Yahoo! Pipes seeks to support end-user programming of web mashups, and provides an expressive visual language and programming environment in which users can create mashup applications. However, there are many challenges end-user programmers face in developing mashup applications, both with and without the support of visual programming tools like Yahoo! Pipes.

The role of developer communities, like the Pipes community, in helping people resolve their programming problems is only beginning to be understood. Coding communities serve as a vital resource in localizing bugs in applications. This is facilitated where code is easily shared among members. Communities also serve as a resource for direct problem solving, helping developers accomplish a goal, although the extent to which help seekers are able to interpret and understand the solutions provided is unclear.

The collaborative marshalling of information and resources, both from the Pipes ecosystem and the larger web, highlight a social element to software development which is often ignored in the design of current software development tools, e.g., debuggers. As the complexity of the development context grows, and the number of interdependencies between elements increases, with increasing abstractions and indirections, so too does the complexity of the debugging and problem solving. Rooting out the source of a problem may touch on more systems than the developer is aware of, and this is where the diverse knowledge and expertise of a community of developers is able to help individuals navigate the issues and sort out solutions. These conversations not only serve the immediate needs of their participants, but serve as public records for future search and are actively referenced and sourced by the community as such. The conversational medium does not necessarily produce well-structured or well-organized documentation, but it is a de facto documentation.

The Pipes community also highlights the role of the periphery in the community process, where at first glance it may appear not much is happening, developers are engaged in collective documentation and problem solving. We do not understand very well what is happening in the periphery of coding communities, and how engaged those persons are in the 'community'. However, the periphery is not entirely void of activity, and in future work,

we wish to dive deeper into the dynamics of peripheral engagement. Do members make their way from the periphery towards to core? We also wish to understand how these conversations are not only directly serving the current participants in their immediate programming needs, but also are being searched, viewed, and reused in future programming contexts by other developers, who may never post questions or comments to the forums at all.

8. References

[1] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, S. R. Klemmer. (2009). Two Studies of Opportunistic Programming: Interleaving Web Foraging, Learning, and Writing Code. In Proceedings of CHI 2009.

[2] M. Burnett (1999) Visual Programming. In: Wiley Encyclopedia of Electrical and Electronics Engineering (J. Webster, ed) John Wiley & Sons, Inc.

[3] B. Dervin, P. Dewdney (1986). Neutral questioning: A new approach to the reference interview. Research Quarterly, 25 (4), 506-513.

[4] K. Crowston & J. Howison (2005). The social structure of Free and Open Source software development. First Monday, February, 2005.

[5] I. R. Floyd, M. C. Jones, D. Rathi & M. B. Twidale (2007) Web Mash-ups and Patchwork Prototyping: User-driven technological innovation with Web 2.0 and Open Source Software. In Proceedings of HICSS'07.

[6] T. R. G. Green & M. Petre (1996) Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. Journal of Visual Languages and Computing 7, 131-174.

[7] C. A. Halverson, J. B. Ellis, C. Danis, & W. A. Kellogg, (2006). Designing task visualizations to support the coordination of work in software development. In Proceedings of CSCW '06. 39-48.

[8] J. D. Herbsleb, A. Mockus, T. A. Finholt, & R. E. Grinter, (2000). Distance, dependencies, and delay in a global collaboration. In Proceedings of CSCW '00. 319-328.

[9] M. C. Jones, E. F. Churchill, & M. B. Twidale (2008). Mashing Up Visual Languages and Web Mashups. In the Proceedings of the 2008 IEEE Symposium on Visual Languages and Human-Centric Computing.

[10] M. Karamuftuogulu (1998). Collaborative Information Retrieval: Toward a social informatics view of IR interaction. JASIST 49(12):1070-1080.

[11] I. R. Katz and J. R. Anderson (1987). Debugging: An Analysis of Bug-Location Strategies. In Human-Computer Interaction. vol. 3: Taylor & Francis.

[12] J. Lave & E. Wenger (1991). Situated Learning: Legitimate Peripheral Participation. Cambridge University Press: Cambridge, UK.

[13] R. Lougher & T. Rodden. (1993) Supporting long-term collaboration in software maintenance. In the Proceedings of the conference on Organizational computing systems.

[14] M. B.Rosson, J. Ballin & J. Rode (2005). Who, What, and How: A Survey of Informal and Professional Web Developers. In Proceedings of VL/HCC'05.

[15] S. Sawyer, J. Farber, R. Spillers (1997). Supporting the social processes of software development Information Technology & People, 10(1): 46.

[16] J. Stylos, B. Graf, D. Busse, C. Ziegler, R. Ehret & J. Karstens (2008). A Case Study of API Redesign for Improved Usability. In the Proceedings of VL/HCC 2008.

[17] J. Stylos and B. A. Myers.(2006). Mica: A Web-Search Tool for Finding API Components and Examples. In Proceedings of VL/HCC 2006.

[18] J. Stylos & B. Myers (2007). Mapping the Space of API Design Decisions. In the Proceedings of the 2007 IEEE Symposium on Visual Languages and Human-Centric Computing.

[19] A.T.T Yang, J.L Wright, & S. Abrams. (2005). Source code that talks: an exploration of Eclipse task comments and their implications to repository mining. ACM SIGSOFT Software Engineering Notes, 30(4).

[20] N. Zang & M. B. Rosson. (2008). What's in a mashup? And why? Studying the perceptions of web-active end users. In the Proceedings of the 2008 IEEE VL/HCC.