

Adobe Monarch

Community Analysis Tool

<http://code.google.com/p/monarch-flex>



Andrew Spencer

aspence3

Mariusz Choroszy

mchoro2

Ryan Lin

ryanlin2

OVERVIEW

Team

<u>Adobe</u>	<u>UIUC</u>
<p>Puneet Goel</p> <ul style="list-style-type: none">• primary liaison• Product Marketing Manager• Platform Business Unit• pugoel@adobe.com• Meeting ID: 63906• Toll-Free(US & Canada): 877-220-5439 <p>Vera Carr</p> <ul style="list-style-type: none">• Associate Product Manager• Adobe Flex• verac@adobe.com <p>Matthew Chotin</p> <ul style="list-style-type: none">• Senior Product Manager• Adobe Flex	<p>Andrew Spencer</p> <ul style="list-style-type: none">• MySQL, PHP, crawling, server, Flex• aspence3@illinois.edu <p>Mariusz Choroszy</p> <ul style="list-style-type: none">• Flex, Air, GUI, concept art• mchoro2@uiuc.edu <p>Ryan Lin</p> <ul style="list-style-type: none">• MySQL, PHP, linguistics, concept art• ryanlin2@uiuc.edu

Client Profile

Adobe

Adobe Systems is an American computer software company headquartered in San Jose, California, USA. It was founded in December of 1982 by John Warnock and Charles Geschke. As of January 2007, Adobe has 6677 employees, about 40% of whom work in San Jose. Adobe also has major development operations in Seattle, WA; San Francisco, CA; Ottawa, Ontario; Minneapolis, MN; Newton, MA; San Luis Obispo, CA; Hamburg, Germany; Noida, India; and Banalore, India.

Adobe has developed many new and innovative award-winning solutions which have served artists, developers, and businesses on a worldly scale. Their most renowned software include Photoshop and and Flash.



Liaison: Puneet Goel

Our liaison, Puneet Goel, is a Product Marketing Manager at Adobe Systems based out of San Francisco, CA. Puneet has walked to the ends of the Earth to introduce Adobe Flex and Air to not only businesses, but also to students at Universities in the US and abroad; so that they can use these new media to build rich internet applications (RIA's). Puneet's division is graciously sponsoring UIUC in areas which relate to these new and growing technologies.

Before joining Abode about one year ago, Puneet worked at Bain & Company as a consultant and at Oracle as a developer on the Supply Chain Logistics team. Puneet's impressive list of credentials include holding a bachelor's degree in Manufacturing Engineering from Indian Institute of Technology, Delhi, masters in Mechanical Engineering from Stanford University,

and an MBA from University of Chicago.

Puneet's division is in charge of marketing products. He has promoted, under the platform and business division, such products as Flash Player, Flex, Air, and Cold Fusion. Puneet does not supervise anyone in his department. He does, however, report to the director of product marketing.

Original Problem (Verbatim)

Adobe is looking for a community analysis tool. While this would be used by Adobe for its own use, we would like the architecture to be generic enough that other organizations can use it or customize it for their needs.

The tool should be able to analyze the following content types:

- blog posts
- articles
- forum posts
- news articles
- Flex and AIR applications

The tool will

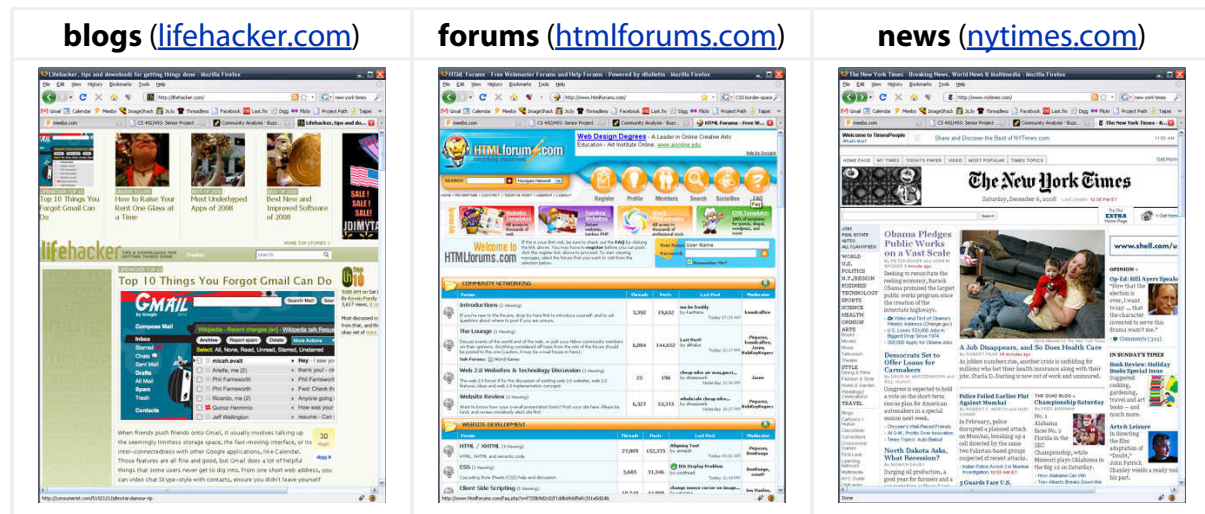
- Gather and aggregate all relevant content that matches the content type
- Perform analysis by trying to find common themes and trends and interesting data. Interesting data points might be around finding community size (number of authors), engagement in particular articles or forums (number of responses or comments), trends in topics (common keywords or phrases).
- Present the analysis using a Flex/AIR front-end.

If designed in a generic way, the tool could be used to analyze any community. The organization using the tool just needs to write adapters to scrape appropriate forums and blogs.

Revised Vision

This project is dubbed Monarch for its godlike power to oversee anything that interests you. Monarch will only be used to analyze:

- blog posts and reader comments
- forum posts
- news articles and reader comments



Monarch revolves around user-supplied keywords. These are nouns that the user is interested in. Example keywords for a music producer might include Miley Cyrus, Timbaland, microphone, and iTunes. Given these keywords, our tool will analyze what people are saying about these keywords on blogs, forums, and news sites. It will plot changes in their count and sentiment over time.



The ability to track a keyword's sentiment is unique to Monarch. Never before in the history of computer science, has there been an algorithm to detect whether someone loved or loathed something. Through weeks of laboring in the Thomas M. Siebel Center labs, our scientists were able to develop the crucial link between the brain's endorphin level and the binary thoughtlessness of a computer algorithm. One must not overlook the ramifications of this historical feat. Its impact on the human race and heavens may be even more profound than finding a polynomial solution to an NP-Complete problem. Now it is possible to exploit the stock market, track how well your product is doing, or even see the fluctuating popularity of a celebrity or politician! A detailed rundown of this magnificent algorithm is revealed in the Engineering section of this holy document.

1

2

Another key feature is discounting bogus posts written by buffoons who carelessly wrote their post concerning a keyword. Users can switch on our patented English proficiency filter to give more weight to people who have their head on straight. Our English proficiency metric is an average of spelling, punctuation, and capitalization.

Websites may be clustered into "community groups". These groups contain related websites. Users may view aggregate statistics across all websites under some particular community group. An example is a T-shirt community group with the following websites under it:

T-shirt Community Group



Threadless.com

A screenshot of the Threadless.com website. The main content area displays a grid of various t-shirt designs. On the left, there is a sidebar with navigation links such as "Guys", "Girls", "Participate", and "Info". At the top, there is a search bar and a "Post a blog!" button.

LaFraise.com

A screenshot of the LaFraise.com website. The main heading reads "15.000€ Contest". Below it, there is a detailed announcement for a contest where users can win 15,000€ by submitting their best t-shirt designs. The website has a clean, modern layout with a blue and white color scheme.

DesignByHumans.com

A screenshot of the DesignByHumans.com website. The main content area shows a grid of user profiles, each with a profile picture, a name, and a featured t-shirt design. The website has a dark theme with a grid layout for the user profiles.

Required Features

Analyze websites periodically

The crawler should run periodically to refresh the database with new statistics. If one scrape session crashes, it will not crash the entire system or delay upcoming scrapes. The user will tell us how many pages of threads to scrape and the amount of time between each scrape session, because only the user knows how lively their website is. The scraper will notify the user if it can't scrape as fast as the user wants and dynamically adjust the scrape frequency.

Website Adapter

Because of differing source code across websites, it is impossible to design a crawler that is generic enough to understand the structure of any website. Users must help Monarch by specifying where to start scraping, what to scrape, and the hierarchical structure of the website they need analyzed.

The database structure will only support storing regexes for traditional forums, news, and blogs. The tool will only accept Perl Compatible Regular Expressions (PCRE), which are accepted by the PHP preg_(*) functions.

Adapter editor

A user friendly GUI should guide the user on how to specify the adapter for their websites. The GUI will load up a visual website along with its source code. As the user types in a regex, the corresponding source code segment will be highlighted in realtime. Regexes will be stored in the database when the user hits submit.

Sample adapters

Our liaison wants sample adapters for Yahoo Groups, Google Groups, blog feeds, and Twitter. These samples will provide new users with some direction on how to specify their own adapters.

Results viewer

The GUI will support the following functions:

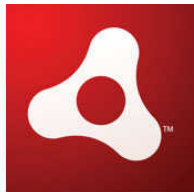
- Account registration.
- Login with username and password.
- Logout.
- Community group creation and adding websites under the community.
- Adding keywords to be scraped for under a community group.

- An edit page to delete, rename, and remove community groups, websites, and keywords.
- Adjustment of crawling frequency and depth of a website.
- View other member's community groups and websites.
- Adjustment sliders to control the weighting of a keyword/link's count, sentiment, and English proficiency towards its total "hotness".
- Default slider positions which provide the optimal hotness determined by empirical testing.
- Users will have a choice of viewing their statistical graphs in at least line or bar graph form.
- Superpositioning of multiple keywords/links stats for comparison..

These are the types of statistics that you will see in the GUI.

1. Frequency of posts
2. Number of posts in a particular time period
3. Number of total authors
4. Number of posts per author
5. Sentiment, English proficiency, and count of keywords/links.
6. Top keywords and links overall calculated by a weighting of the above three metrics.
7. Top keywords/links spoken by authoritative authors (big post count).
8. Listing of authors who are very positive or negative about a keyword/link.
9. Listing of threads which talk very positively or negatively about certain keywords.
10. Listing of threads with high engagement level.

Deployment



The final product must be displayed in Adobe Flex and Air. It must showcase the power of these Adobe tools and make them more popular in colleges across America. This is the main goal of our liaison's department.

Eventually, this project will be turned into an open source project. The code must be well organized and documented so other people can easily pick up where we left off.

Planning



Server language

For crawling and linguistic analysis, our group chose to use PHP. Although not the most flexible or high performance language, PHP allowed us to develop extremely fast. It had numerous built in functions for web stuff such as downloading pages, converting an English time into the Unix timestamp, etc...



Database

PHP has always been traditionally paired with MySQL, so we stuck to this 1-2-punch. Other reasons for choosing MySQL include it being a free alternative to Microsoft SQL and its pairing with PHPMyAdmin, an easy-to-use web interface for database administration.



Frontend requirement

The client strongly encouraged us to use Adobe Flex, a framework for developing Rich Internet Applications, and Adobe Air, a runtime environment for enabling desktop RIAs, to develop a web-enabled desktop client for the project. The final product is intended to showcase the strengths of this Adobe combo for creating a unique RIA.



Distribution of work

Ryan worked mainly on the backend scraping and algorithm design because he is comfortable with PHP and MySQL. Andrew programmed the crawler and scheduler because he is familiar with Linux has a guru-like understanding of languages in general. Andrew also delved into the frontend GUI with Mariusz. Mariusz laid most of the groundwork for the GUI, already having much experience with graphics.

Working remotely

Our team has been teleconferencing with Puneet once a week. Puneet gave us the reigns to the project from the get-go, but many times has steered the project toward success when he detected usability or conceptual issues. Many times Vera Carr and Matt Chotin have joined in to dispense their invaluable advice on user interfaces and Flex. Coast to coast communication was by no means a dainty task. Sometimes we'd mess up the time zones and other times the static on the caveman-era teleconference phone would painfully pierce the eardrum like the howling streak of nails on a chalkboard.

Our liaisons are based in California, while we are in Illinois. We have communicated mostly by teleconference, which has sometimes made it hard for each side to express exactly what they are thinking. Using screen-sharing software has helped a little for demonstrating new ideas but communication problems still exist. Sometimes we would misinterpret something one week, then work on it for a week, and come back the next week to find out that we were wrong. A key misunderstanding involved what exactly was meant by the word *community*. At one point, every member of our group and our liaison were each operating with a unique definition of *community*. It could have meant

- an entire website
- a medium within a website (e.g. forum, etc.)
- a group of web sites.

However, the operating definition of community in this project is a group of people that share a common interest. We have sometimes found it difficult to translate this idea into software but have more or less settled on a user interface that should allow for a user to view the results of community analysis.

Storage constraints

Originally, we were planning to store statistics for every word in the English dictionary. Google does this; you can search for trends of any word in Google Trends. However, due to our server's storage limits we decided to store detailed statistics only for a given set of keywords. It would be great to do it Google's way, but we just don't have enough resources or the time to make such a solution work efficiently. As a result, our system should be able to provide the user with relevant information quickly, as well as accommodate a fair number of users.

It took us a 2 months to acquire a server and software. The server was initially improperly configured to only use a 2GB partition for /var and we wasted a week before figuring this out.

English proficiency

In December, we developed various metrics for English proficiency - spelling, punctuation, and capitalization. We assumed that if at least 51% of people who rushed when typing and didn't excel in these metrics were not as trustworthy as those who did, then this metric would be of some use. Liason Vera brought up concerns of applying these metrics to people with English as a second language. We argue in the Engineering section a few pages down that foreigners can handle these metrics without problem.

Hardcore processing

Crawling and scraping one page of threads takes around 3 minutes. We need to devise more efficient algorithms and do multi-threading to make a crawl run faster. PHP does not support multi-threading directly but we may attempt to leverage the Apache web server's multi-threading to execute multiple PHP scripts concurrently. However, the fact that these scripts must contribute to solving the same problem makes this a difficult technical challenge. We recognized these limitations of PHP when we chose it as the language of the back-end as we had decided that its merits (brevity) outweighed its limitations (hardcore programming).

Bug tracking

Puneet wanted Monarch to be open source, so we uploaded it to [Google Code](#). We have been using its [issue tracker](#). These issues will provide direction for anyone wanting to work on the project. In its brief existence, the *issues* page has provided direction for development allowing us to coordinate our efforts. Prior to its existence, identification and elimination of bugs was much less central and there was no forum to discuss progress, aside from email.

Defining the Problem Space

4

Because this software operates in an entirely new problem space, we needed to hammer down exactly what was meant by "community", "crawl", "scrape", and "custom adapter" and identify how these definitions could be translated into software. You can generally describe what is meant by "online community" but in order to incorporate such an idea into a software product, a more exact definition is required. After multiple discussions amongst ourselves and discussions with our liaison, eventually we settled on the resulting data structure that can be seen in detail in the Engineering section of this document under *Database*. Not to discount the product we have developed, but if we were to restart from the beginning having familiarity with the problem space I think there would be potential for a very useful and possibly revolutionary product. Through the year, we have identified the real difficult problems in this problem space and would focus on these if given the chance to start over. For

example, a great hindrance to the usefulness of this tool is the cumbersome process of specifying regular expressions for each new website, a task that non-computer scientists would struggle with. Rethinking the process of adding a website would be the first problem that would need to be addressed if starting this project over. We did not recognize this as a real issue until we had a tool that could be tested by users very late in the development cycle. It could be a very difficult problem but it deserves heavy thinking.

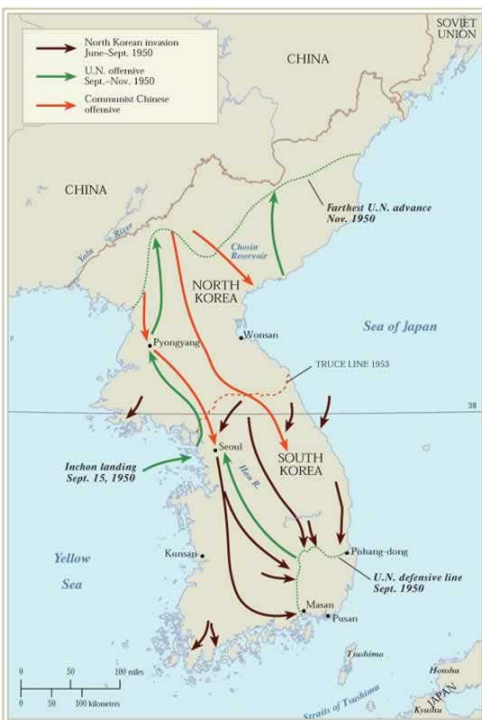
Scheduling

After limited success with PHP scheduling systems and Linux cronjobs, we decided for a quick, simple solution: a short python script. Unlike the other two options, we are aware of everything that is happening. It turns out to be a solid solution because the script can be run from any computer that has a Python installation and an internet connection. It is multi-threaded and uses negligible resources. Also, it can be readily extended to incorporate more complicated scheduling behavior such as dynamic adjustment of task periods.

Regular Expression Editor

5

Lieutenant Doo-Yong Chung's defection



Doo-Yong Chung joined our group in the first semester. He did minimal work. He never came back for second semester. We called him and sent him e-mails, but he never responded. A near exact situation occurred with Ryan's South Korean roommate. We speculate that many South Koreans were deported back to Korea to fight in a top secret war.

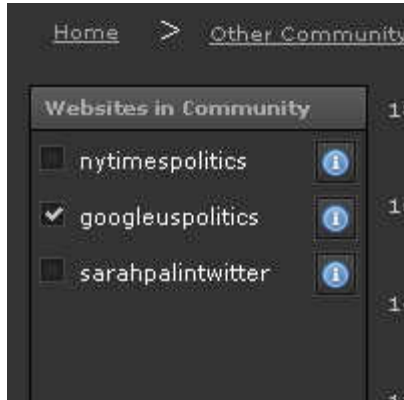
6

When the war quelled, Doo-Yong made a return to the States. One of Andrew's private investigators spotted Doo-Yong in downtown Chicago.

Doo-yong's defection is an insult to our group. He will never be forgiven.

Unfinished Business

Aggregate stats



You cannot aggregate stats between two websites in the same community.

7

Dynamic crawler

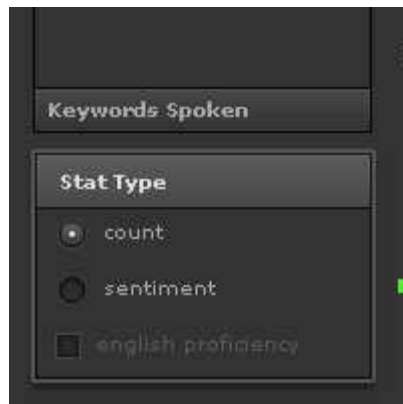
The crawler does not adjust its frequency and depth if it can't stay on schedule. . We provided an alternative solution by forcing a lower bound on the time between scrape sessions and an upper bound on the number of threads to crawl on each scrape (monarch-client\AdobeAnalyticsTool\src\cRegularExpressionEditorCanvas.mxml). These bounds still allow for the vast majority of sites to be fully analyzed and they also put less stress on the server. If you truly wanted dynamic frequency and depth, edit monarch-server/scheduling/scrapeOrderer.py .

Yahoo Groups adapter



Puneet wanted an adapter for Yahoo Groups. Although we were fully capable of providing one, we chose not to. Nearly all of the lively Yahoo Groups were private, meaning that we could not crawl the group. We felt that crawling a public group with only a few posts per day would not yield meaningful statistics. If you happen to stumble upon a lively public Yahoo Group, feel free to add the adapter by opening up the GUI and going through the regex helper.

Graph sliders

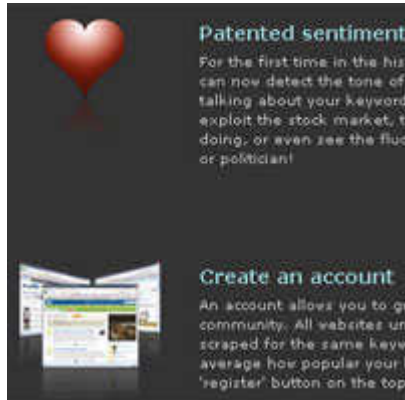


The image shows a dark-themed user interface. At the top, there is a header area. Below it, a section titled 'Keywords Spoken' is visible. Underneath this, there is a 'Stat Type' section. This section contains three radio button options: 'count' (which is selected), 'sentiment', and 'english proficiency'. The interface has a clean, modern look with subtle borders and a consistent color palette.

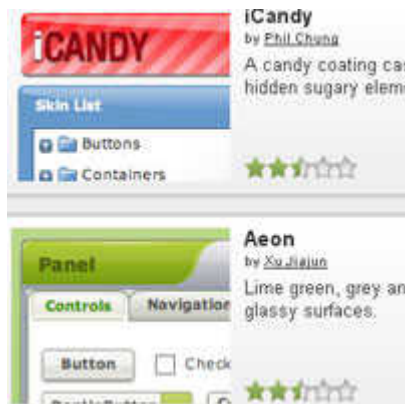
We listed graph control sliders as a requirement but removed them due to the feedback from our liasions. They felt that the sliders overwhelmed the end user with too many options and would confuse him/her. To simplify things, radio buttons replaced the sliders. Although not nearly as powerful, they provide a one click solution for controlling the graphs. If you are confident that you can devise user-friendly slider controls, edit the code in `monarch-client\AdobeAnalyticsTool\src\cCommunityGroupBrowserCanvas.mxml`.

Improvement

Recommendations from Adobe



The Adobe representatives have suggested in many of our teleconference meetings to improve the usability of the tool. The biggest issue is that the wording does not help newcomers understand how to use the tool. We had trouble coming up with words that could describe a service that never existed before. To alleviate the problem, we have added tooltip rollovers almost everywhere. However, there is still work to be done to improve the user experience.



A secondary issue that we have been struggling with since first semester was designing an aesthetically appealing GUI that would appease our liason. We agree with Puneet saying that a tool must be visually appealing to capture the interest of potential users. Our team has been hard at work trying out and adapting new skins from www.scalenine.com. Although the current skin, Kingnare, is a drastic improvement from our early versions, we still feel that more vivid color and images could be added to spice up the design.

Both of these issues will involve extensive changes to the files in monarch-client/AdobeAnalyticsTool/.

Recommendations from UIUC team

- Security from users entering SQL injection attacks through the GUI (monarch-server\Client).
- Scientifically-based linguistic analysis or use of AI techniques like Bayesian classifiers (monarch-server\Client\Linguistics.php).
- Cached analysis. The GUI would query this cached table instead of recalculating statistics. Using a tool like Memcached is probably the most promising option as it only requires some changes to database query code.
- Other types of graphics for displaying keyword/link stats such as pie charts or word clouds (monarch-client\AdobeAnalyticsTool\src\cCommunityGroupBrowserCanvas.mxml).
- Efficient code that runs in at most linear time in all cases (monarch-server\Client\Linguistics.php).

- Mapping specific threads and users onto the keyword/link stat line graphs. For example, the user sees a sharp peak in the graph. She clicks the peak and it takes her to a thread or user which confirms to her the peak behavior (monarch-client\AdobeAnalyticsTool\src\cLineChartCanvas.mxml).
- Exporting of data as text (e.g. CSV format) and exporting of graphs as images.
- Help files with table of contents and search. This could take the form of local hyperlinked help or online help.
- Graceful error reporting - similar to Microsoft, whose products allow a user to send automatic error reports which allow the developers to identify and fix bugs.
- Displaying software news updates on the login page. This might be implemented in a way similar to Meebo's blog posts.
- Multi-threading of a single crawl for faster execution. Each thread would analyze only a fraction of a top-level page. Since the backend uses almost exclusively PHP, multi-threading is not supported. This would either require porting everything to another language or designing a way in which a single crawl may be broken up and executed by multiple PHP scripts.
- Distributed databases - splitting data over several databases on different servers for performance (monarch-server\database).

FOR THE CLIENT

Complete Feature List

- register for an account
- login
- logout
- create a community with a list of keywords
- view communities other people have created
- view only communities that you've created
- sort communities by date added, views, and alphabetically.
- edit a community that you've created
 - change its name
 - upload an icon
 - delete keywords
 - add new keyword to analyze
 - remove a website from the community
 - add a new website under the community
- regex editor for adding a website
 - step by step instructions with screenshot help
 - highlights HTML source code that matches the regex
 - extracts all matches of the regex
- graph of keyword stats over time
 - number of mentions each day
 - change in sentiment over time
 - apply an English proficiency weight to the sentiment of a keyword
 - adjust time frame to view these keyword stats
- general stats for a website
 - average number of posts per day
 - number of posts today
 - number of posts analyzed
 - number of threads analyzed
 - number of posters
 - most spoken link
 - most hated link
 - most loved link
 - profile URL of the person who made the mosts posts
 - profile URL of the person who joined the website earliest
 - profile URL of the person who joined the website latest
 - URL of thread with most responses
 - URL of thread with most views
 - most mentioned external link
 - most liked external link
 - least liked external link
- detailed website specific keyword stats

- top 3 threads that love the keyword
- top 3 threads that hate the keyword
- top 3 threads that mention the keyword a lot
- top 3 users that love the keyword
- top 3 users that hate the keyword
- top 3 users that mention the keyword a lot
- top 3 users that use sophisticated prose when writing about the keyword

Installation



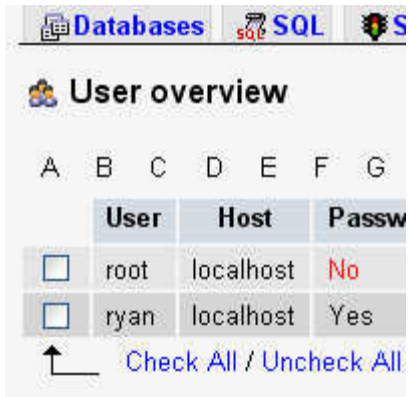
Obtain a server. Any OS will work although these configuration instructions only apply to Redhat Enterprise Linux release 5.2. It's recommended that you get a server with a multi-core CPU because the crawler serves many clients' websites simultaneously.



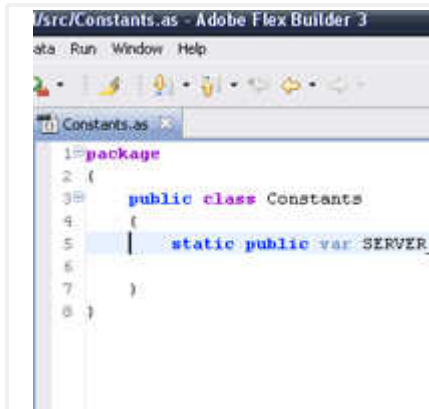
At minimum, install these versions:

- Apache 2.2.8
- MySQL 5.0.51b
- PHP 5.2.8

You can conveniently install all three at once by downloading LAMP (or WAMP on Windows).



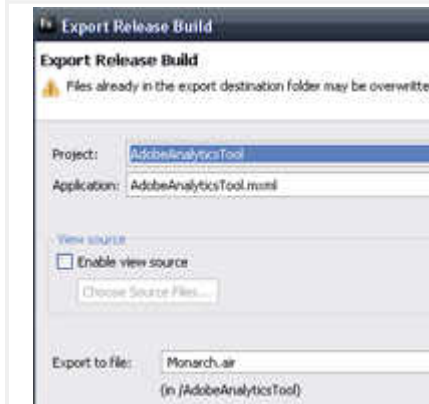
In PHPMyAdmin, click the Privileges tab and create a new user with privileges granted for everything.



Open up Adobe Flex Builder 3. In path/Constants.as, change the server variable to where the Client folder is.

14

15




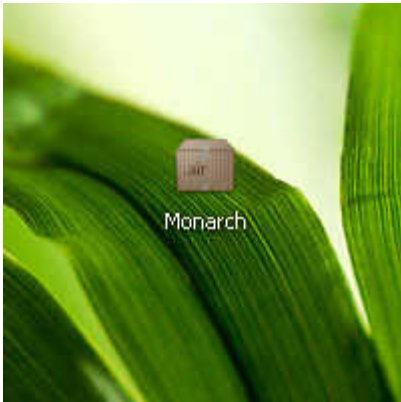
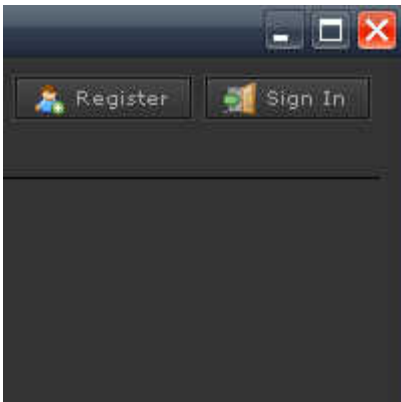
Re-export as an Air installer so the tool users can use connect to the right server. Re-upload that Air file to Google Code because the old one is pointing to the UIUC server.



Run the python scheduling script 'scrapeOrderer.py' on the server machine. This script will automatically run web crawls based on the websites in the database and the crawling parameters for that website. This script should run forever unless your server crashes or you accidentally stop it. See source for more information.

16

User Manual

	<p>Install Adobe Air for your operating system. Monarch will run on top of the Air runtime environment.</p>
	<p>Double click Monarch.air to install Monarch.</p>
	<p>Monarch should start automatically after installation. Register for an account and you will be automatically signed in.</p>

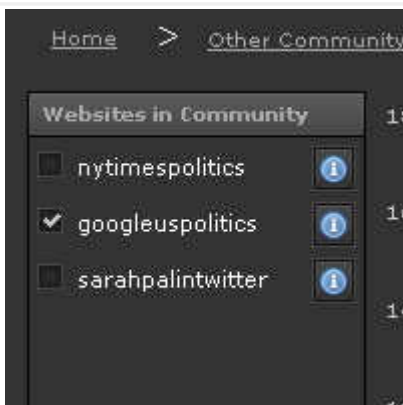


Before you create your own community groups, look in check out other people's existing community groups to see what Monarch is capable of.



You can sort other people's community groups by time, by how many times they've been viewed (a good gauge of their popularity), or alphabetically.

Click any community group you're interested in to continue.



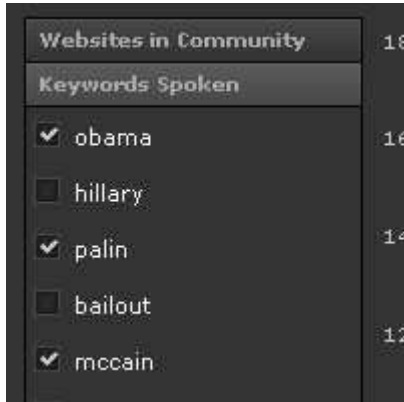
Check any website under the community to view stats from it. Check more if you want to aggregate stats from two or more websites.

17



By default, you will view stats for the current week. Try larger time spans by using the toggle menu in the lower right to see older stats. Old stats will only appear if the creator had been put this community into Monarch a long time ago.

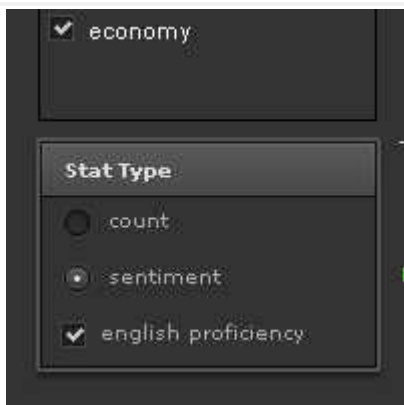
If the graph lines are going out of the viewing window, click the magnifying glasses in the lower right to readjust the y-axis.



If your line graph is very cluttered, open up the keywords control in the accordion on the left and uncheck some keywords to remove them from the graph.

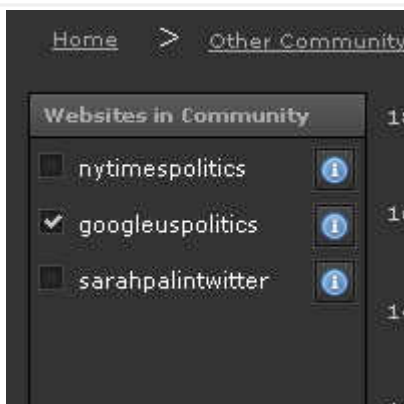


By default, you are viewing the count of the keywords. This keeps track of how many times each keyword was spoken by the users of this community's websites each day. The count can be used to gauge the buzzworthiness of a keyword.



You can also switch over to the sentiment stat view. This is the unique selling point of Monarch! Sentiment will detect if people are praising or trash-talking a keyword. When sentiment is on, you can optionally turn on the English proficiency. This puts more weight on people who wrote carefully (spelling, punctuation, capitalization) when speaking the keyword.

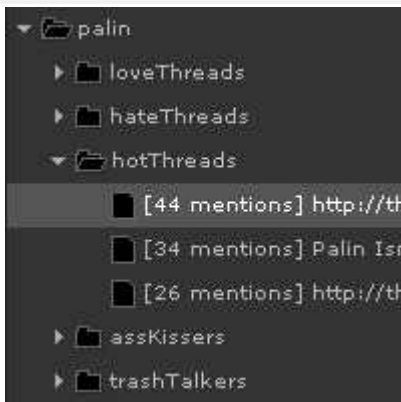
When viewing the graph, negative values mean hate and positive values mean love.



What you've been viewing so far are aggregate stats across one or more websites in the community. To find website specific stats, click the blue "i" next to a website in the accordion.

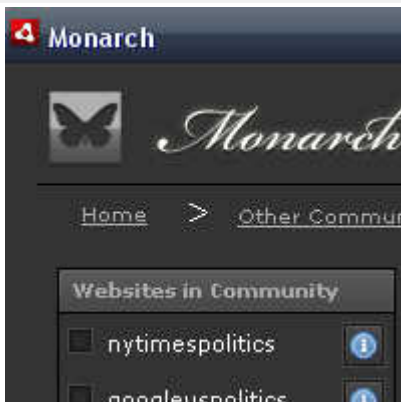


While in the website specific stats page, the left column shows general stats and external URL's that people have been linking to in their posts.



On the right column, open up the tree to go to actual threads or user profiles that have been talking about a specific keyword. Click a leaf node to open up the page in your browser.

- love threads: threads whose general consensus is that the keyword sucks.
- hate threads: threads whose general consensus is that the keyword rules.
- hot threads: threads with many mentions of the keyword.
- ass kissers: user profiles of people who like the keyword.
- trash talkers: user profiles of people who hate the keyword.
- chatterboxes: user profiles of people who mention the keyword a lot.
- sophisticated orators: user profiles of people who write the keyword with thoughtful prose.



Now we'll show you the other side of Monarch. Either click the Monarch logo or click the "home" breadcrumb.



Now click on "My Community Groups" to view the communities that you've created or create a new one if you don't have any.

 A screenshot of the "Create" form for a new community group. It features a "Community Group Name" field at the top, followed by eight "Keyword" fields labeled "Keyword 1" through "Keyword 8". A "Save" button is located at the bottom right.

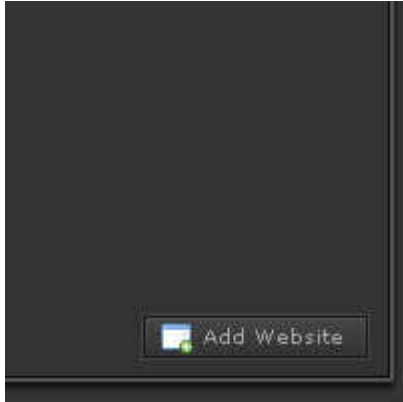
Click the "create community group" button on the upper right. Enter in a subject that you're interested in and some keywords relating to that subject that you want Monarch to analyze.

 A screenshot of the "Icon" upload screen. It shows a small image of a person walking in a park. Below the image is a "Save" button.

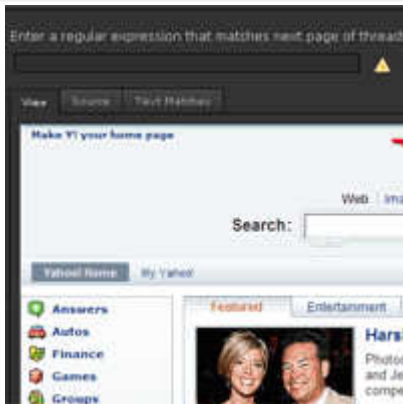
Go into your group. Click "edit community group" in the upper right. Upload an icon so you can easily identify your community later.

 A screenshot of the "Edit Community Group" page. It displays a list of keywords: "sony", "windows", "samsung", "adobe", and "dell". Each keyword has a "remove" button next to it. At the bottom, there is an "add" button with a green plus icon.

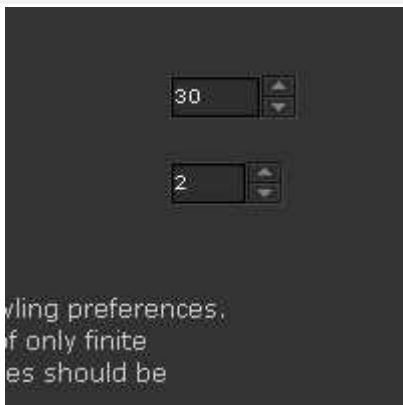
In the middle column of the edit community page, you can remove keywords to tell Monarch to stop analyzing the keyword. You can also add new keywords at the bottom.



A community group without any websites under it is useless, so the next thing you'll want to do is add a website. Click the "add website" button in the lower right to begin.



You'll have to enter in regular expressions to let Monarch know how to crawl a website. Enter in a proper regex for each step to make the warning sign disappear. The view tab shows the graphical rendering of the website, the source tab shows the HTML source, and the text matches tab shows which parts of the HTML your regex is matching. If you need additional help, click the blue question mark button to bring up detailed help for the particular regex. Click the green arrow button to go to the next step.



The last step of adding a website is to specify how often and how much should Monarch scrape the website. This requires some personal judgement. Determine how lively the website is so that you don't miss any new posts between scrapes.

Testing

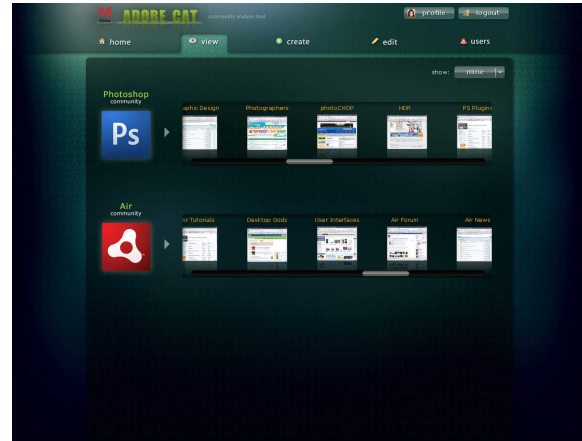
User test #1

We showed Ryan's mockups to our liasons. Below each screenshot are their comments.



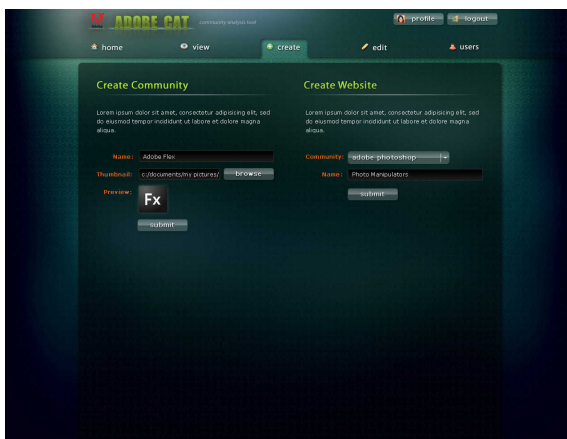
Home

Liaison Vera told us to take out the "about" column after a user has already signed in.



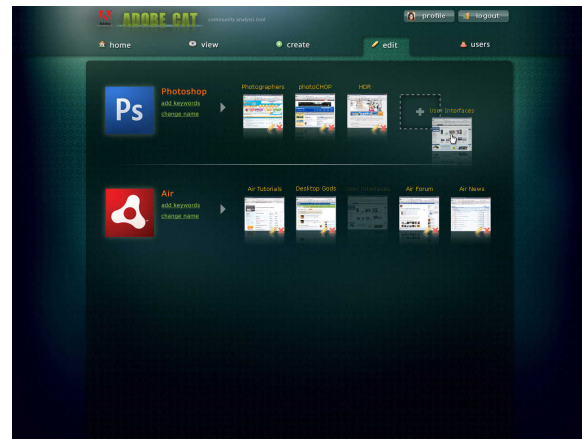
View

This page is fine.



Create

Liaison Puneet wanted to change the phrase "create website" to "add website" because confused individuals might think that they are actually going to code up a website.



Edit

This page will be merged into the "view" page. Vera wishes to have these features while you are viewing stats for a community group or website.

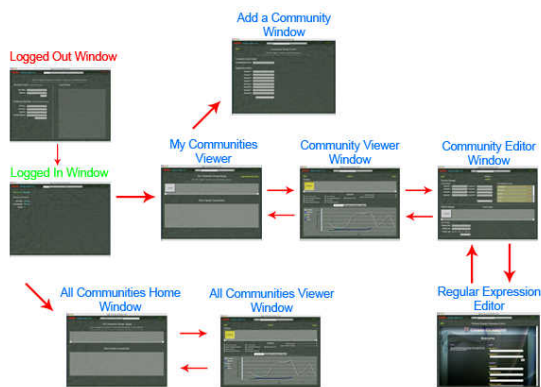


Users

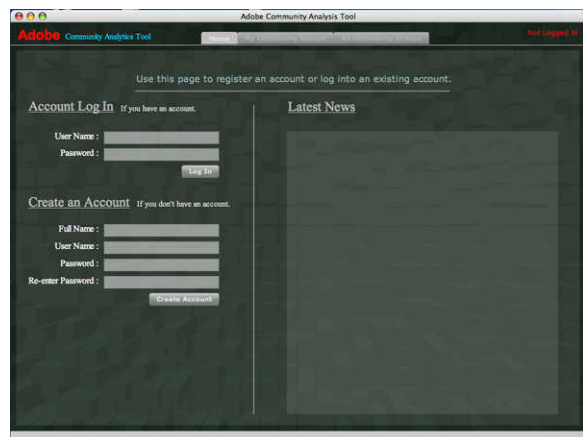
This page will be scrapped because Puneet does not expect an extreme number of users to warrant social networking.

User Test #2

We showed Mariusz's mockups to our liasions but did not explain them because Mariusz was not present at the meeting.



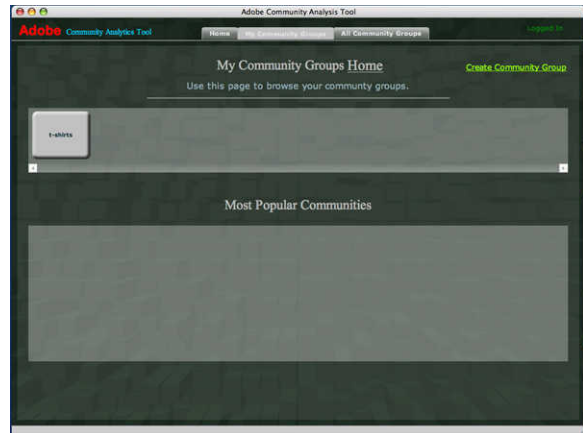
Flow chart



Login



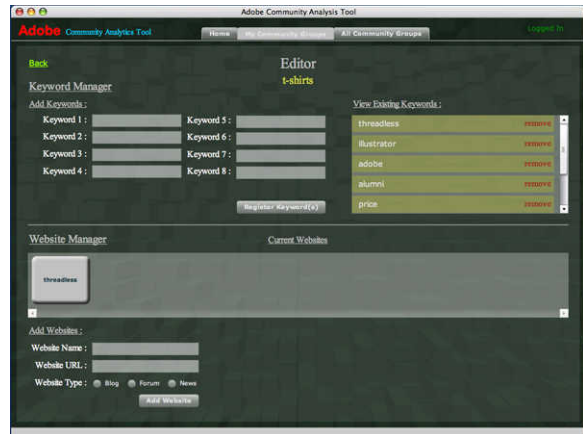
Logged in



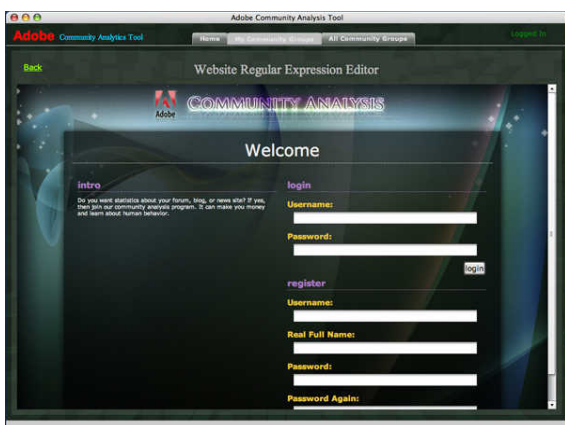
My community groups home



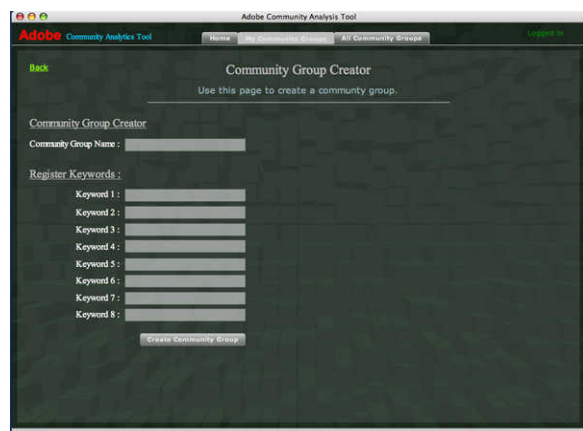
Community group view



Community group editor



Regular expression editor

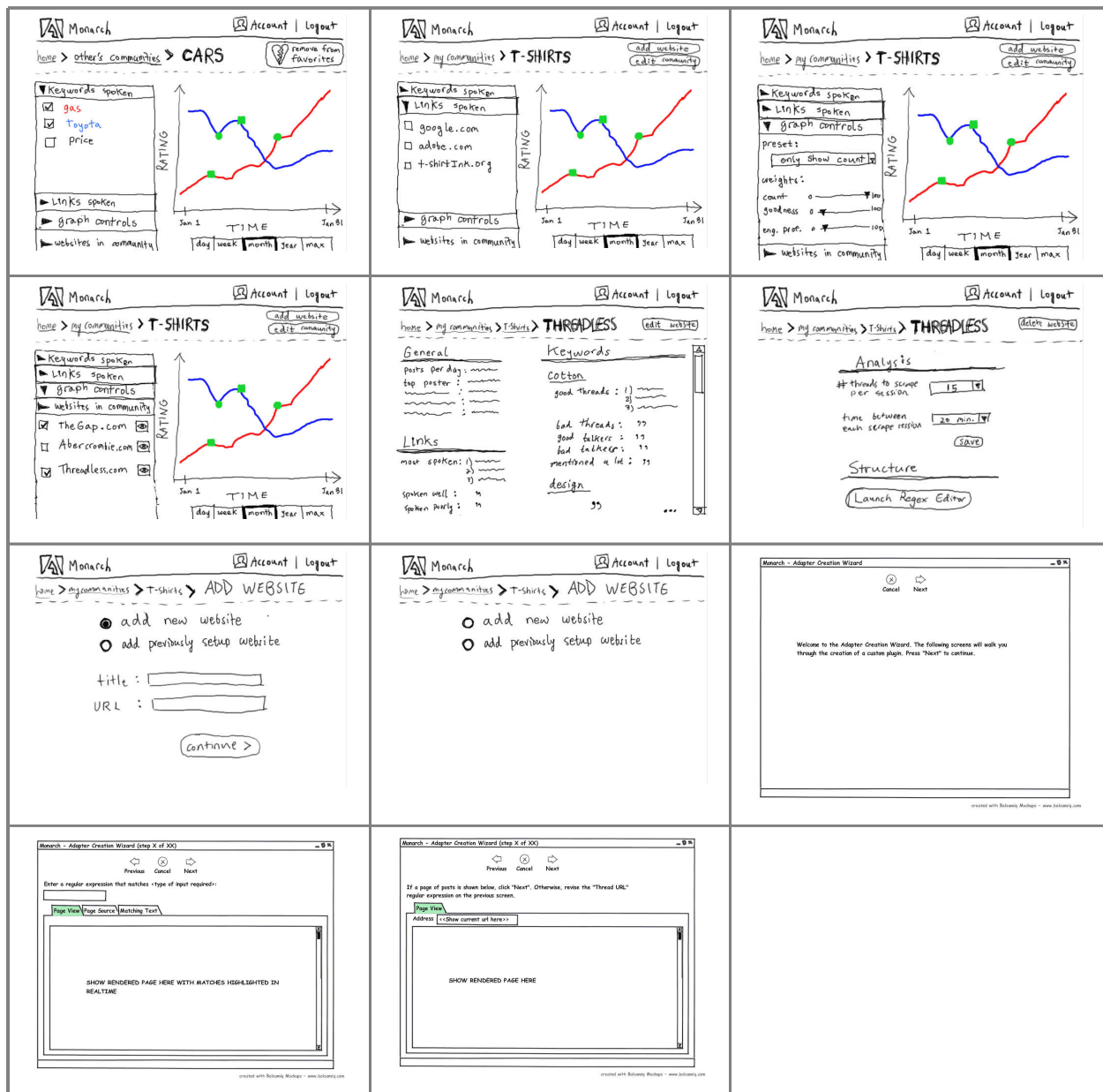


Add community group & keywords

User Test #3

Taking the comments of the two previous user tests, we constructed a third user test. This time, we drew out *every* required feature. We showed the mockups to two people at Adobe, one who was already familiar with the project and one who had never heard of it. Each person was asked how they would register, get into the Cars community group, determine whether people liked or hated Toyota, and how to go to threads and user profiles that were speaking about Toyota.

<p>Monarch Account Logout</p> <p>home > my communities > T-shirts > ADD WEBSITE</p> <ul style="list-style-type: none"> ○ add new website ● add previously setup website <div> <input type="text" value="gucci.com"/> <input type="text" value="CarMechanic.net"/> <input type="text" value="abercrombie.com"/> <input type="text" value="gucci.com"/> </div> <p>continue ></p>	<p>Monarch register login</p> <p>Hot Communities Info</p> <div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> </div> <p>previous next back next</p>	<p>Monarch register login</p> <p>Hot Communities Tinfo</p> <div> <div> <p>Register</p> <p>email <input type="text"/></p> <p>Password <input type="password"/></p> <p>Submit</p> </div> <div> <p>this tool is for you should see.</p> </div> </div> <p>previous next back next</p>
<p>Monarch register login</p> <p>Hot Communities Tinfo</p> <div> <div> <p>Login</p> <p>email <input type="text"/></p> <p>Password <input type="password"/></p> <p>Remember me <input checked="" type="checkbox"/></p> <p>Submit</p> </div> <div> <p>this tool is for you should see.</p> </div> </div> <p>previous next back next</p>	<p>Monarch Account Logout</p> <p>My Communities other's Communities</p>	<p>Monarch Account Logout</p> <p>home > MY COMMUNITIES Create Community</p> <p>newest most viewed alphabetical</p> <div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> </div> <p>previous next back next</p>
<p>Monarch Account Logout</p> <p>home > my communities > CREATE</p> <p>community name: <input type="text"/></p> <p>websites under this community (separate with comma): <input type="text"/></p> <p>Keywords to scrape for (comma separated): <input type="text"/></p> <p>Icon: <input type="text"/> browse</p> <p>Submit</p>	<p>Monarch Account Logout</p> <p>home > my communities > t-shirts > EDIT</p> <p>Basics</p> <p>Name: <input type="text" value="t-shirts"/></p> <p>Icon: <input type="text"/> browse</p> <p>Save</p> <p>Keywords</p> <ul style="list-style-type: none"> <input type="checkbox"/> cotton <input type="checkbox"/> design <input type="checkbox"/> price <p>add new keyword</p> <p>Websites</p> <ul style="list-style-type: none"> <input type="checkbox"/> www.1.com <input type="checkbox"/> www.2.com <input type="checkbox"/> www.3.com <input type="checkbox"/> www.4.com <p>add website</p>	<p>Monarch Account Logout</p> <p>home > OTHER'S COMMUNITIES</p> <p>newest most viewed alphabetical favorited</p> <div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> <div> <p>COMMUNITY NAME WEBSITE: www.1.com, www.2.com Created by: admin</p> </div> </div> <p>previous next back next</p>
<p>Monarch Account Logout</p> <p>ACCOUNT delete account</p> <p>Real Name <input type="text"/></p> <p>Username <input type="text"/></p> <p>Password <input type="password"/></p> <p>Password Again <input type="password"/></p> <p>Avatar <input type="text"/> browse</p> <p>Save</p>	<p>Monarch Account Logout</p> <p>home > my communities > T-SHIRTS add website edit community</p> <p>Keywords spoken</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> cotton <input checked="" type="checkbox"/> design <input type="checkbox"/> price <p>Links spoken</p> <p>Graph controls</p> <p>Websites in community</p> <p>RATING</p> <p>Jan 1 Jan 91</p> <p>day week month year max</p>	<p>Monarch Account Logout</p> <p>home > other's communities > CARS Add to favorites</p> <p>Keywords spoken</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> gas <input checked="" type="checkbox"/> Toyota <input type="checkbox"/> price <p>Links spoken</p> <p>Graph controls</p> <p>Websites in community</p> <p>RATING</p> <p>Jan 1 Jan 91</p> <p>day week month year max</p>



We received the following comments for this user test and have since implemented all of them:

- The graph controls should always be visible.
- Label the graph lines clearly. Always make the legend visible.
- There needs to be context sensitive help on the accordion because the sliders and "websites in community" are not easily understandable.
- Change "goodness" to "sentiment".
- Ask for the password twice when registering in case the user makes a mistake.
- Have a default added community under "my communities" for new users.
- Matt doesn't like the word "spoken" in the accordion.

- The accordion needs to be split up into two groups: keywords/links & graph controls/websites.
- Change the eye icon to a magnifying glass because eye icons usually mean visibility, not "more detail".
- Incorporate "My Communities" and "Other Communities" links into the homepage.
- Combine the two registration steps.

Algorithm Test

Various articles were passed into our sentiment, spelling, punctuation, and capitalization checkers. We inspected results visually and used our human judgement. There was no way to automate our algorithm testing because computers can never fully understand the English language (not until 2092, when they become smart enough to enslave the human race).

trash talking

Adobe is bad. It sucks. I hate it.

bad (after inversion/amplification) decreased the rating of adobe by -0.500000

sucks (after inversion/amplification) decreased the rating of adobe by -0.250000

hate (after inversion/amplification) decreased the rating of adobe by -0.166667

goodness of "adobe": -0.916667

negated trash talking

Adobe is not bad. It does not suck. I don't hate it.

bad (after inversion/amplification) increased the rating of adobe by 0.333333

suck (after inversion/amplification) increased the rating of adobe by 0.142857

hate (after inversion/amplification) increased the rating of adobe by 0.100000

goodness of "adobe": 0.576190

ass kissing

Adobe is good. It is great. I love it.

good (after inversion/amplification) increased the rating of adobe by 0.500000

great (after inversion/amplification) increased the rating of adobe by 0.200000

love (after inversion/amplification) increased the rating of adobe by 0.142857

goodness of "adobe": 0.842857

severe ass kissing

Adobe is very good. It is so great. I love it.

good (after inversion/amplification) increased the rating of adobe by 0.666667

great (after inversion/amplification) increased the rating of adobe by 0.285714

love (after inversion/amplification) increased the rating of adobe by 0.111111

goodness of "adobe": 1.063492

split personality

Adobe sucks. I love Adobe.

sucks (after inversion/amplification) decreased the rating of adobe by -1.000000

love (after inversion/amplification) increased the rating of adobe by 0.333333

sucks (after inversion/amplification) decreased the rating of adobe by -0.333333

love (after inversion/amplification) increased the rating of adobe by 1.000000

goodness of "adobe": 0.000000

multi-word inverters

Adobe is hardly ever respectable.

respectable (after inversion/amplification) decreased the rating of adobe by -0.250000

goodness of "adobe": -0.250000

wishes

I wish Photoshop had been more efficient. He suffered from using the slow Photoshop.

efficient (after inversion/amplification) decreased the rating of photoshop by -0.250000

suffered (after inversion/amplification) decreased the rating of photoshop by -0.166667

slow (after inversion/amplification) decreased the rating of photoshop by -0.100000

efficient (after inversion/amplification) decreased the rating of photoshop by -0.142857

suffered (after inversion/amplification) decreased the rating of photoshop by -0.200000

slow (after inversion/amplification) decreased the rating of photoshop by -1.000000

goodness of "photoshop": -1.859524

misspelling

My spelleeng iss horibel. I am nott a trustwoorthy persun.

spelling: 0.200000

correct spelling

My spelling is good. My posts can be trusted.

spelling: 1.000000

bad punctuation

I am bad at " punctuation) and (cannot) be "trusted"

punctuation: 0.416667

good punctuation

I am good at "punctuation" and (can) "be" trusted.

punctuation: 1.000000

bad capitalization

i rush when typing... often i don't capitalize!! don't trust me?!?! I only capitalize one sentence

capitalization: 0.250000

good capitalization

I am a careful writer! I take my time to capitalize. Trust me!

capitalization: 1.000000

new york times

With the economy deteriorating rapidly, the nation's employers shed 533,000 jobs in November, the 11th consecutive monthly decline, the government reported Friday morning, and the unemployment rate rose to 6.7 percent.

english proficiency: 0.923077

idiot from youtube

old ur breath and copy this and then paste it to another vid. if u dont let ur breath go ur a gd kisser. (i managed)

english proficiency: 0.564103

capitalization: 0.000000

punctuation: 1.000000

spelling: 0.692308

idiot from threadless

do you like sci-fi? watch Firefly. do you like westerns? watch Firefly. do you like space? watch Firefly. do you like well-written dialog? watch Firefly. do you like awesome characters? watch Firefly. ARE YOU GETTING THIS?

english proficiency: 0.679426

capitalization: 0.090909

punctuation: 1.000000

spelling: 0.947368

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.111111

like (after inversion/amplification) increased the rating of firefly by 0.066667

like (after inversion/amplification) increased the rating of firefly by 0.045455

awesome (after inversion/amplification) increased the rating of firefly by 0.043478

like (after inversion/amplification) increased the rating of firefly by 0.111111

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.111111

like (after inversion/amplification) increased the rating of firefly by 0.062500

awesome (after inversion/amplification) increased the rating of firefly by 0.058824

like (after inversion/amplification) increased the rating of firefly by 0.066667

like (after inversion/amplification) increased the rating of firefly by 0.111111

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.333333

like (after inversion/amplification) increased the rating of firefly by 0.100000

awesome (after inversion/amplification) increased the rating of firefly by 0.090909

like (after inversion/amplification) increased the rating of firefly by 0.045455

like (after inversion/amplification) increased the rating of firefly by 0.062500
like (after inversion/amplification) increased the rating of firefly by 0.100000
like (after inversion/amplification) increased the rating of firefly by 0.250000
like (after inversion/amplification) increased the rating of firefly by 0.333333
awesome (after inversion/amplification) increased the rating of firefly by 0.250000
like (after inversion/amplification) increased the rating of firefly by 0.034483
like (after inversion/amplification) increased the rating of firefly by 0.043478
like (after inversion/amplification) increased the rating of firefly by 0.058824
like (after inversion/amplification) increased the rating of firefly by 0.090909
like (after inversion/amplification) increased the rating of firefly by 0.250000
awesome (after inversion/amplification) increased the rating of firefly by 0.333333
goodness of "firefly": 4.831258

Schedule

Date	Task	Leader	Status
10/15/08	architecture discussion	us, Puneet	simple backend is working
10/22/08	demo of of GUI and plugin editor	us, Puneet, Matt	working on them
10/29/08	<ol style="list-style-type: none"> 1. Ability to add a community and edit it's plugin. Architecture where many websites may fall under one community. Cached analysis. 2. Scheduling the scraper to run once a day. Fix the calculation of average time for keywords. 3. Show more stuff in the GUI. 4. Make the scraper traverse pages correctly. Hyperlink stats. 	<ol style="list-style-type: none"> 1. Ryan 2. Doo Yong 3. Mariusz 4. Andrew 	<ol style="list-style-type: none"> 1. cached analysis done, redefinition of community done. 2. done 3. done 4. Hyperlink counts working. For the traversal, I want to do some code restructuring: mostly done
11/05/08	<ol style="list-style-type: none"> 1. Only scrape for user specified list of keywords. Time based trends on keywords. Positiveness / negativeness of a keyword. 2. Figure out how to display a progress loading bar on the scraper, so we aren't staring at a blank screen while it's loading 3. Fix everything in Scrape.php. 	<ol style="list-style-type: none"> 1. Ryan, Mariusz 2. Doo Yong 3. Andrew 	<ol style="list-style-type: none"> 1. done 2. not done 3. I've been rewriting things.
11/12/08	<ol style="list-style-type: none"> 1. Advanced linguistic analysis implementation. (?) 2. User accounts, allow user to create communities and websites. 3. Make a line chart of the keyword trends. 4. Fix the table creation bug. 	<ol style="list-style-type: none"> 1. Andrew 2. Ryan 3. Mariusz 4. Doo Yong 	<ol style="list-style-type: none"> 1. done 2. done - hacked to work on our broken server, but is not secure. 3. done 4. done
11/19/08	<ol style="list-style-type: none"> 1. User interface mockups. 2. How do you create database and table names with capital letters? Insert all these into communityanalysis->badwords: http://www.noswearing.com/dictionary. Install mysql_real_escape_string in our PHP source code. 3. Combine Andrew's crawling code with Ryan's scraping code. 	<ol style="list-style-type: none"> 1. Ryan, Mariusz 2. Doo Yong 3. Andrew 	<ol style="list-style-type: none"> 1. done 2. done 3. done
11/21/08	Thanksgiving break starts	us	-
12/01/08	Thanksgiving break ends	us	-
12/10/08	Connect the backend database with the GUI. Presentation to the class.	us	buggy

12/19/08	<p>Winter Break starts.</p> <ol style="list-style-type: none"> 1. stemming of keywords, goodwords, badwords (and unstemming). 2. Bypassing token URL hacks by websites trying to prevent crawling. 3. Make the GUI look good. 4. Link to the actual thread which most closely corresponds to the local maxima and minima on the line graphs. 5. Figure out what to do with our goodwords/badwords dictionaries not being the same sizes and commonness of the words. 6. Isn't websiteDB->stats->count dependent on how many new posts are scraped or the # of total words that were scraped, or...? So can't use raw count directly in graphs as indication of hotness. 7. Capitalization gave the "idiot from youtube" a 0.333 in <u>our demo</u>. 8. Extend the plugin editor to allow the user to specify a general structured crawl (arbitrary number of page levels and link types). 9. Generate adapters for new sites and generalize ForumPostProcessor if necessary so that it works with all site types. 10. "so", "very", "not", "don't" are not the only modifiers of adjectives. Take "much" and "really" for example. 11. Doo Yong gave me some other dictionary files - test which one is the best empirically. 12. Insert thread titles. Figure out why & crashes am insertion query. 13. forumpostprocessor.php has a private \$start_time which is never used. 14. thread stats (posts, views) are only set when the thread is first viewed. Need to constantly update. 15. There's always a blank named user and some posts have time of 0. 16. Duplicate post checking is not robust - dupes are found on the first crawl! 	us	<ol style="list-style-type: none"> 1. not done 2. fixed by mimicking browser with Curl. 3. not done 4. website->threadstats table tells you this info. 5. not done 6. not done 7. fixed - now I count punctuation at beginning of sentence as miscapitalization. 8. not done 9. It is general enough. 10. Extended modifiers with the help of thesaurus.com, but there were so many multi-word modifiers that I couldn't use because our system doesn't support it. Ex: "not at all". 11. Yes, the ones he gave me were a lot better on our linguisticTests.php. 12. Fixed to work on my PC, not tested on our server. 13. Removed it. 14. I made the change in forumThreadProcessor->scrapeThreads(). 15. fixed by changing regexes. Threadless must have changed their code. 16. not done
01/20/09	Winter Break ends	us	-

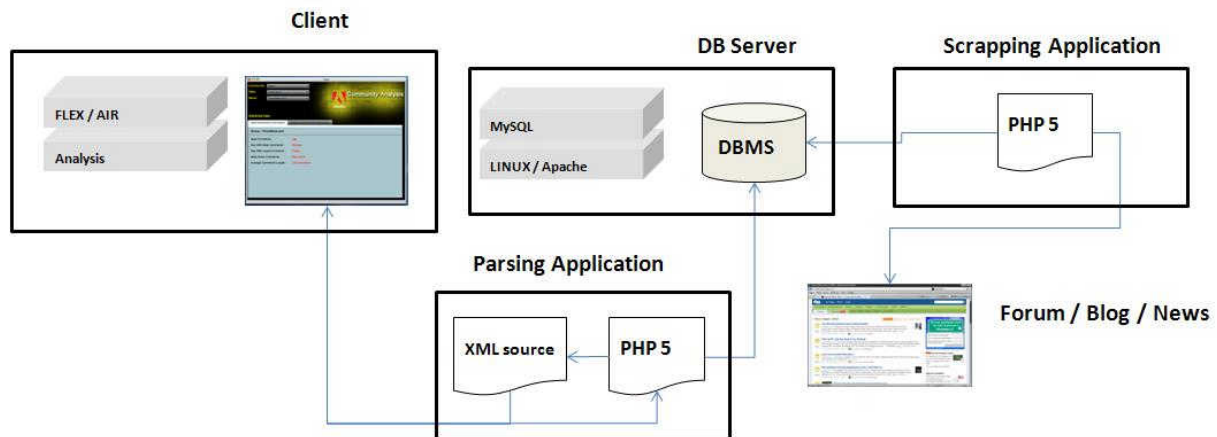
01/23/09	Conference on A/B/C features.	All	
01/30/09	Conference: review progress on workflows and wireframes	All	
02/06/09	Complete workflows and wireframes due;	us	
3/6/09	All A features completed; begin user testing	us	
03/30/09	Final Documentation due	us	-
04/01/09	Hand in to client full documentation.	All	
04/08/09			
04/15/09			
04/22/09			
04/29/09			
05/06/09	Last day of class	us	-

ENGINEERING

VI. Architecture

Entire system

21



Crawling Framework Architecture

22

Link Structure of Threadless

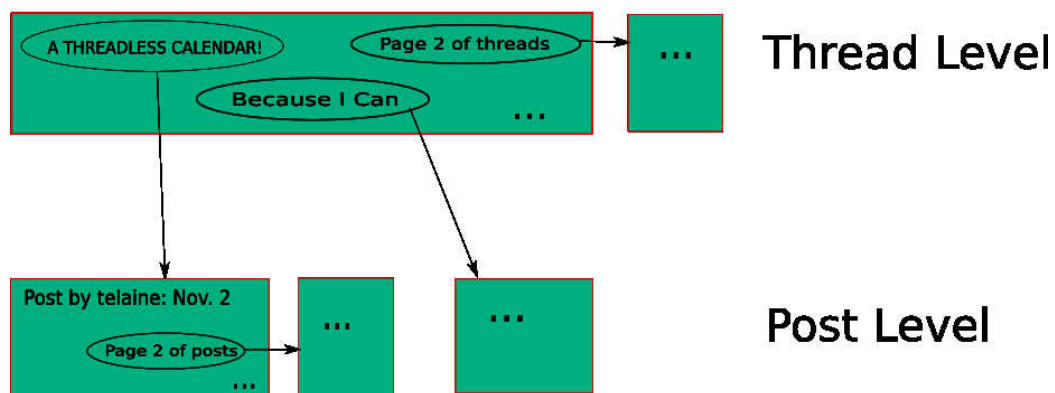


Figure 1: Threadless structure. Ellipses denote links and rectangles denote pages. 23

Link Structure of Ubuntu Forums

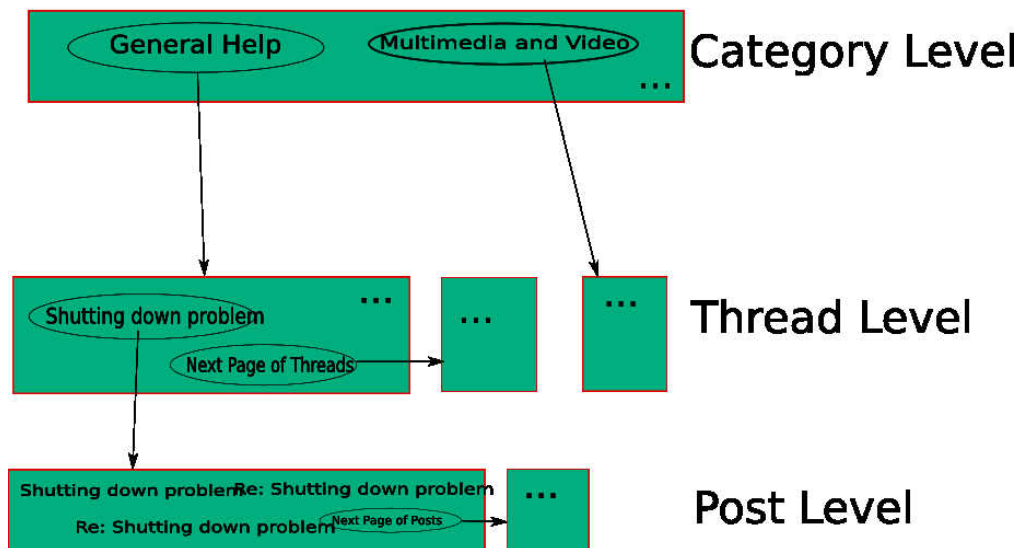


Figure 2: ubuntuforums.org structure. Ellipses = links and rectangles = pages.

In order to make crawling more general for the purpose of code reuse, the crawling model has been reworked. This reworking is based on the observation that different web sites vary greatly in their link structures. The original code developed for *Threadless* (Figure 1) can not easily be adapted for a web site having a different structure like <http://ubuntuforums.org> (Figure 2). In the original code, crawling (finding links and downloading pages) is closely coupled with page processing (gathering statistics and storing them in the database, in this case). Decoupling these two activities as much as possible makes for a cleaner design and should make it easier to define crawl behavior for new sites. While the new model is more general, we still need to make the following basic assumptions to be able to clearly define a crawl:

1. Any crawl may be defined by a finite number of *link types*, which can be thought of as a unique combination of
 - A. a regular expression defining the text representation of the link,
 - B. a class of pages in which this type of link may be found in, and
 - C. the class of page which this link links to.
2. Pages are organized into hierarchical classes. For example, for a site with two classes of pages, pages of threads are above pages of posts because each page of threads provides links to many pages of posts.
3. Processing for a crawl may be defined by an implementation of a *Processor* interface associated with each class of pages. An object of the defined class will execute processing on the html source of the page after the page has been scanned for links.

Given these assumptions, defining the crawl then consists of defining some set of *link types*

and then initiating the crawl from a starting page coming from some specified class of pages. The scope of the crawl can be limited by total pages in the crawl or by the total number of toplevel pages to crawl completely. In the current implementation, the page classes are specified by integers although this is not absolutely necessary. The integers will generally correspond to page classes like pages of posts, pages of threads, or pages of categories.

Once the crawl is initiated, URLs are maintained in data structures to ensure that no page is crawled twice and that pages appearing lowest in the hierarchy are explored first.

In order to define processing on the pages, a processing object can be associated with each page class. This object can maintain state through the duration of the crawl.

It should be straight-forward to apply this crawling method to all types of sites having some hierarchy of page classes.

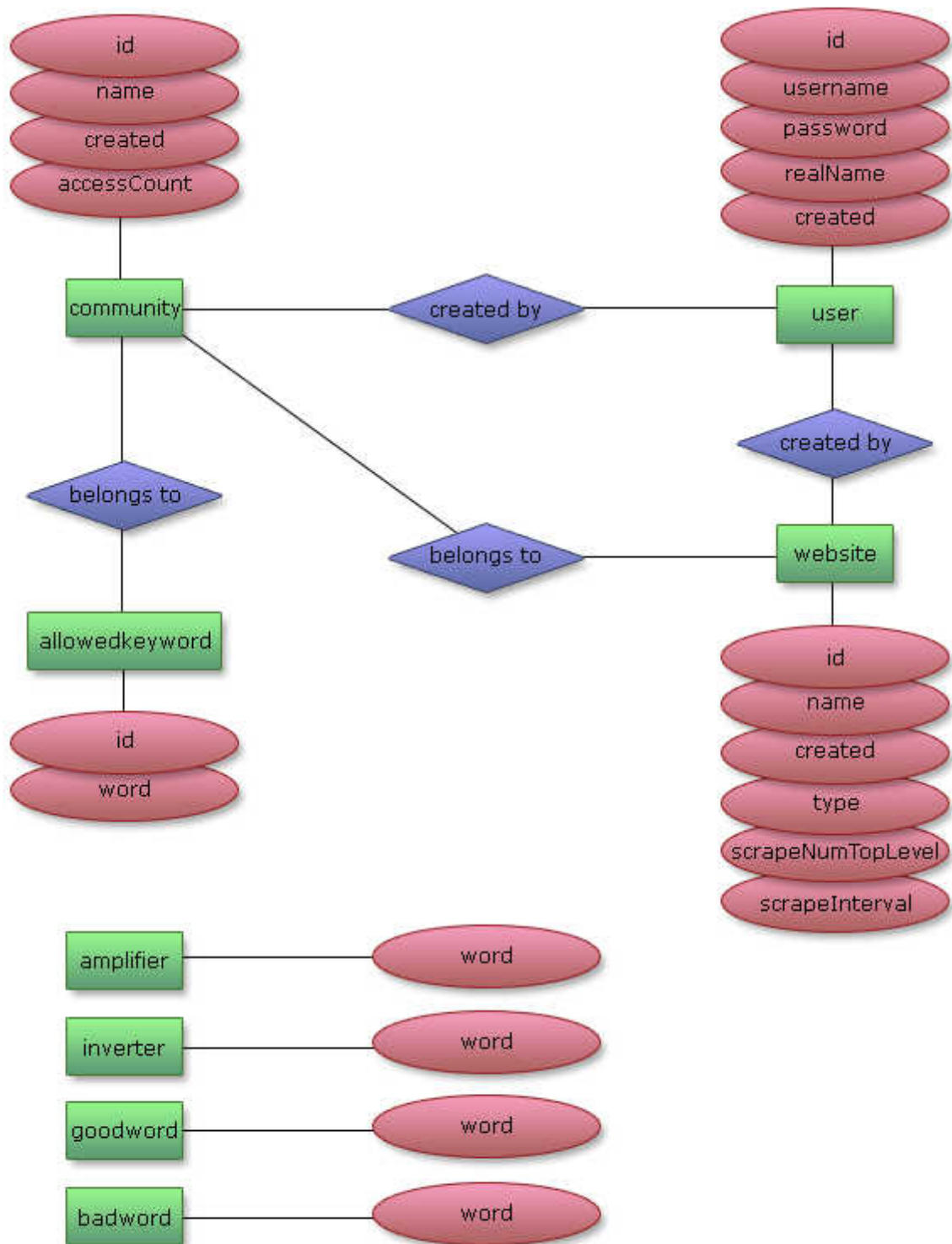
Addition: The scope of the crawl may be defined by the maximum number of pages to scrape for each thread.

Scheduling Web Crawls

We adhered to the KISS principle in designing the scheduling part of the server software. In the current design, users specify both the period between crawls and the scope of a crawl (in number of *top-level pages*) and perpetually running Python script periodically wakes up, makes a database query to determine which websites have reached the end of their period between crawls, and runs the PHP scripts to crawl these websites. Since each crawl is a different request to the Apache server, if multiple crawls are initiated at the same instant they are all run on different threads. This was a positive side effect of our design that was not initially considered.

Database

The master database has the following tables and columns:

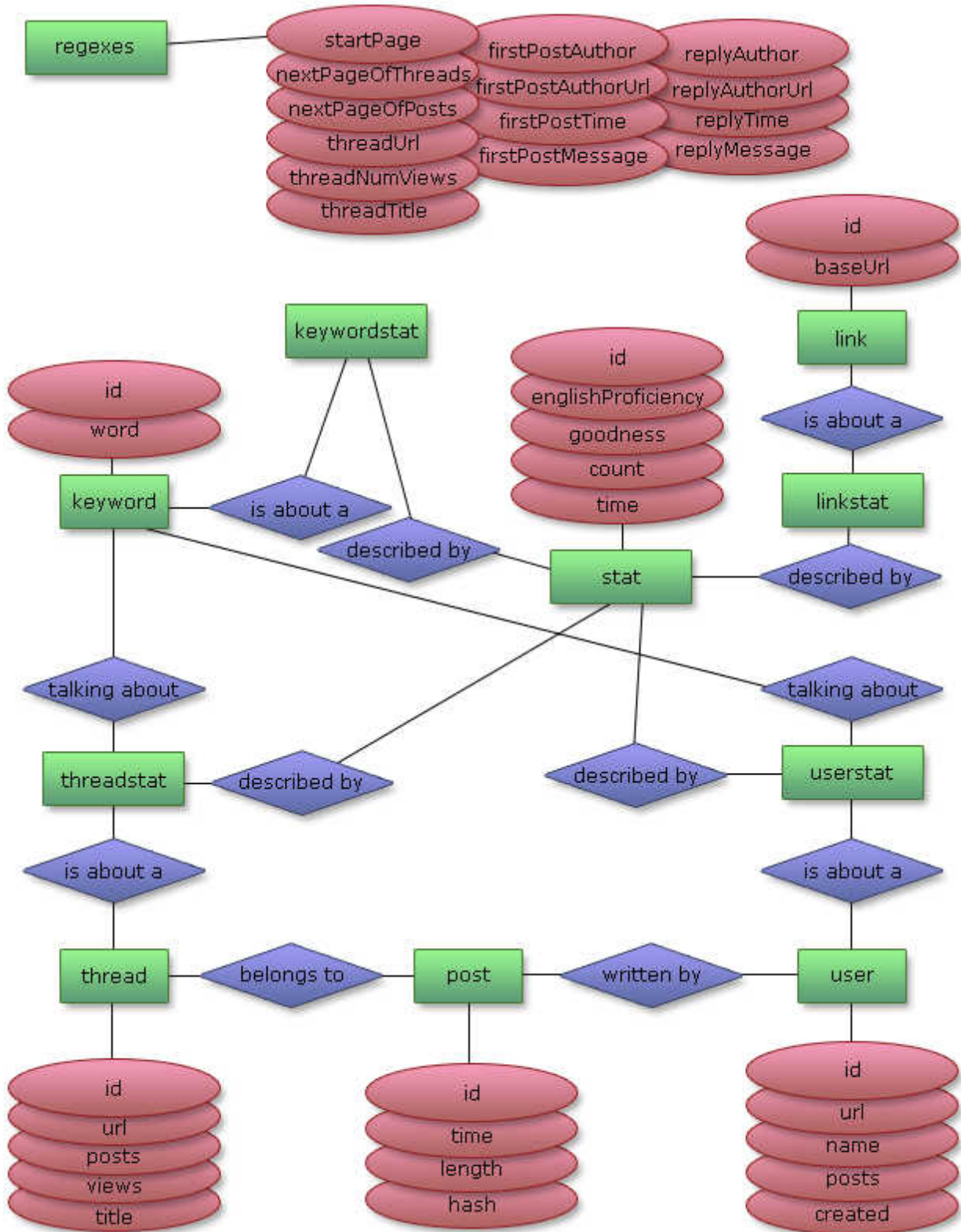


- **communities**
 - the community groups

- id - primary key
- name - name of community
- user - ID of user that created the community
- created - Unix timestamp when this community was created
- **websites**
 - the websites which fall under certain community groups.
 - id - primary key
 - community - id of community a website belongs to
 - user - id of user that created this website
 - created - when this website was added to its community
 - type - the type of the website: forum | blog | news
- **goodwords**
 - Synonyms of "good". Used for linguistic keyword analysis. Examples: love, great, hot, celebrate, decent, worthy...
 - word - a good English word
- **badwords**
 - Synonyms of "bad". Used for linguistic keyword analysis. Examples: hate, sucks, shit, murder, dirty, wetback ...
 - word - a bad English word
- **allowedkeywords**
 - per community, we specify a list of keywords we want to collect stats for
 - id - primary key
 - community - id of community group that scrapes for the word
 - word - the word to be scraped for
- **users**
 - keeps track of who created a certain community group or website
 - id - primary key
 - username - email address of the user
 - realName - full name of the user
 - password - encrypted password
 - created - Unix timestamp when user registered for Monarch

There will be one database per website. This is better for performance and scalability. In the future, databases can be distributed across multiple servers.

Each websites's database must have the following tables:



- **keywords**

- a list of all the keywords that this site has seen so far. This is an improper subset of

- all the keywords that the website's community scrapes for.
- id - primary key
- word - the keyword that was seen by the scraper
- **keywordstats**
 - Shows the graph plot points for a keyword.
 - keyword - foreign key to the keyword
 - stat - foreign key to the stat
- **links**
 - We convert all links encountered to their base. For example, *www.threadless.com/folder/page.html* is converted into *threadless.com*. This allows many links to map to the same base URL, thereby giving us better aggregate stats.
 - id - primary key
 - baseUrl - domain name + domain suffix. Example: yahoo.com
- **linkstats**
 - Shows the graph plot points for a link.
 - link - foreign key to the link
 - stat - foreign key to the stat
- **stats**
 - common table for link and keyword stat entries.
 - id - primary key
 - time - time which we initiated this particular scraping session
 - count - number of times the keyword or link was seen in the scrape session.
 - goodness - accumulated goodness of the keyword or link in the scrape session. Centered around 0; 0 is neutral.
 - englishProficiency - accumulated average of spelling, capitalization, and punctuation of the keyword or link in the scrape session. Divide by count to put in the 0.0 - 1.0 range.
- **posts**
 - Posts under threads. Includes the thread starter's post (or original article if this is not a forum).
 - id - primary key
 - userId - foreign key to the user that wrote the post
 - time - Unix timestamp of when this post was made
 - length - number of characters in this post. Currently, not used in the GUI, but definitely has the potential to be an interesting stat.
 - thread - foreign key to the thread that this post belongs under
 - hash - Many website's store a coarse granularity of the post times. For example, if a website stores times only to minute precision and some user posts twice within the same minute, the crawler will have trouble determining if it already scraped those two posts. To resolve this issue of duplicate post detection, an md5 hash value of the entire post body is stored.
- **threads**
 - We give URL's to threads so that users can read the thread that spoke about their keyword.

- id - primary key
- posts - number of replies in this thread
- views - number of views of this thread. Many websites don't display a view count. This column will be 0 for these websites.
- title - thread title
- url - URL of the first page of the thread
- **threadstats**
 - Tells you how a whole thread is talking about each keyword. Useful for linking back to interesting threads. Unlike keywords and links, this is not a time based trend. A thread will only have one stat per keyword it contains.
 - thread - foreign key to the thread
 - keyword - foreign key to the keyword spoken within this thread
 - stat - foreign key to the stat describing how this thread cumulatively talked about this keyword.
- **users**
 - Members posting on the website.
 - id - primary key
 - posts - the total number of posts this user has made (which Monarch is aware of).
 - name - username
 - created - time of the earliest post of the user that Monarch has analyzed of this user.
 - url - URL to the user's profile page.
- **userstats**
 - user, keyword, stat
 - Tells you how each individual user is talking about each keyword. Useful for linking back to powerful user's profile pages. Unlike keywords and links, this is not a time based trend. A user will only have one stat per keyword they say through the entirety of their existence.
- **Regexes** - shows us how to crawl the website. Contains the following columns:
 - startPage - (String) url of initial page
 - nextPageOfThreads - (regex) matches a link to "next" page of threads
 - nextPageOfPosts - (regex) matches a link to "next" page of posts
 - threadUrl - (regex) matches a link to a different (or new) thread
 - threadNumViews - (regex) matches a count of the number of views of this thread
 - threadTitle - (regex) matches the title of the thread (text)
 - firstPostAuthor - (regex) matches the text of the first post's author
 - firstPostTime - (regex) matches the text of the first post's time
 - firstPostMessage - (regex) matches the text of the first post's message body
 - replyAuthor - (regex) matches text of the reply author
 - replyTime - (regex) matches the text of the reply time
 - replyMessage - (regex) matches the text of the reply message body

Each website that wants to participate in our project will need to fill out the regex (Perl style) table in their community. This table allows us to parse the necessary data and crawl between pages given their link structure.

Using the regular expressions, PHP first examines the first page of threads and fills out all columns in the threads table. We go into the thread to parse the replies. Per each reply, we check to see if the author is already listed in our database. If so, we update his join date (best of our knowledge) and increment his post count. If not, we create a new entry in the users table. We also need to detect if this reply was scraped in a previous session, skipping duplicate replies. Per each reply we scan through all of the body's links and keywords and update its stats with the following metrics:

Sentiment

Huge word lists were inserted by hand into our database because no dictionary company has ever classified words into the following categories:

- good words
 - adjectives: good, sexy, efficient, ...
 - verbs: help, embrace, love, favor ...
 - exclamations: wow, rad, cool, ...
- bad words
 - crimes: murder, theft, suicide, ...
 - curses: fuck, shit, damn, bitch, fag ...
 - racial slurs: wetback, nigger, chink, ...
 - adjectives: barbarous, stupid, dirty, ...
- inverters
 - negatives: don't, never, not, ...
 - wishes: should have been, wish he had, ...
 - adverbs: hardly ever, rarely, seldom, ...
- amplifiers
 - so, very, substantially, ...

Using these gigantic word lists, our team developed a sentiment algorithm. It originally started out as a joke, but critics were shut up as soon as they saw it perform flawlessly on [Tupac](#) lyrics, [New York Times](#) articles, [Youtube](#) posts, and many more texts.

$$\text{sentiment} = \text{Sum over all goodWords} \{ 1 / \text{distance from keyword} * \text{modifier} \} - \text{Sum over all badWords} \{ 1 / \text{distance from keyword} * \text{modifier} \}$$

modifier = -1 if adjective appears after an inverter: "don't", "not", ...

modifier = 2 if adjective appears after an amplifier: "very", "so", ...

Basically, if the keyword appears near good words, it will get a higher goodness rating.

If it appears near bad words, it's goodness decreases. Take for example this sentence:

"I *don't* *hate* Adobe. It is *so* great."

legend

- keyword
- adjective
- modifier

goodness of Adobe = $1/(\text{distance to great}) * (2) - 1/(\text{distance to like}) * (-1)$

goodness of Adobe = $1/3 * 2 - 1/1 * -1$

goodness of Adobe = 1.666

It is true that sometimes when a *badword* is said near a *keyword*, it doesn't mean that the keyword was said negatively. Take for example "it *sucks* that my mom forced me to uninstall *Microsoft* Windows." Here, the keyword *Microsoft* appeared near the bad adjective *sucks*, but the speaker was speaking positively about Microsoft. We can't account for all such twists on *badwords* or *goodwords*, but according to Monte Carlo distribution, we should be able to converge to the true positiveness / negativeness of a keyword as we increase the amount of scraping and the size of the *goodwords* and *badwords* table increases.

english proficiency

englishProficiency = (spelling + punctuation + capitalization) / 3

An average of our custom spelling, punctuation, and capitalization (in detail below). We are assuming that people who rush when typing make more English mistakes and are less trustworthy. Their contribution to the keyword's hotness should be decreased.

We have not alienated foreigners with this metric. Grammar was specifically left out as it is the most difficult subject of English. Most countries which use a Roman alphabet have similar punctuation and capitalization rules as in English.

spelling

spelling = words spelled correctly / total number of words

An entire English dictionary is loaded into an array. We scan through each word one by one to verify if it is in the dictionary. Since PHP's array search did linear search, we programmed our own binary search to logarithmically cut down run times.

punctuation

punctuation = # dangling quotes and parentheses /
 (# dangling quotes and parentheses + # properly closed q's and p's.)

We have not implemented detection of proper nesting yet.

capitalization

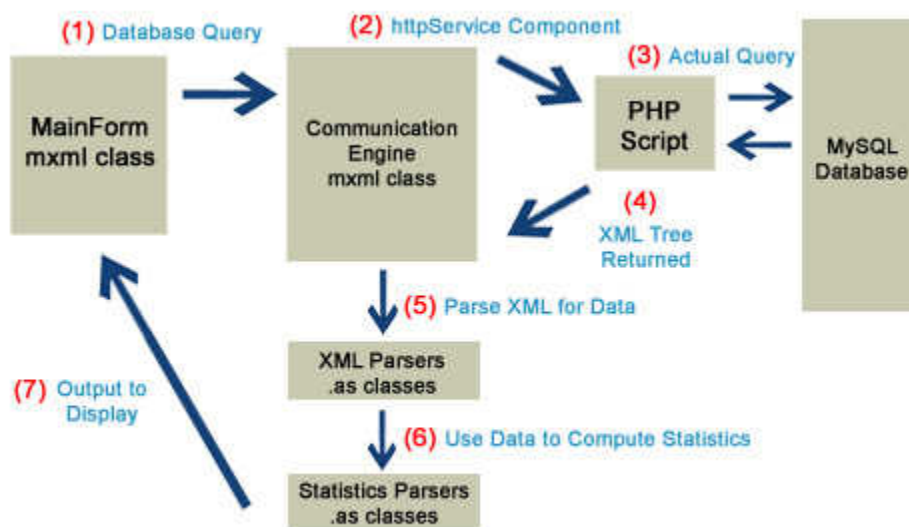
capitalization = # of capitalized sentences / total # of sentences

Only checks if first word of every sentence is capitalized. We have not implemented checking for capitalization of proper nouns yet.

Frontend

Database Communication Scheme:

26

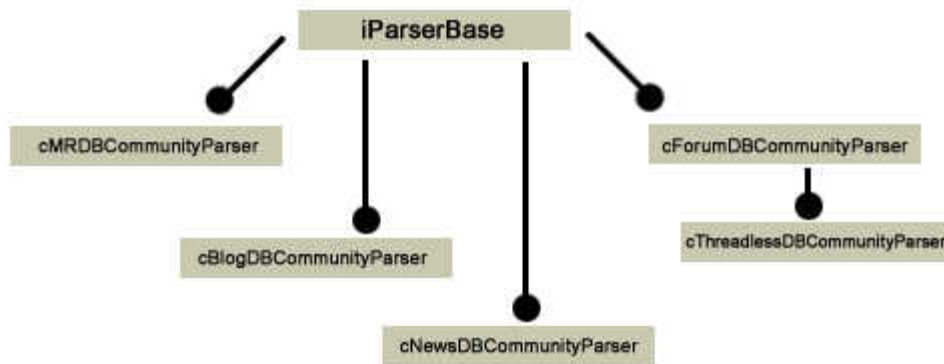


The client application structure is designed in a three part form. The three parts are as follows:

1. the MainForm - responsible for all GUI functionality within the main window
2. the Communication Engine - responsible for all most communication between the client application and the database
3. the XML and data parser classes which extract raw data and arrange it in an output format.

27

XML Parser Hierarchy:



The parsers are responsible for extracting raw data from an XML object. Each parser will inherit from `iParserBase` which acts as an interface class for all the derived parser objects. Each derived parser object contains an internal collection of data pertaining to the specific parser. The `cMRDBCommunityParser`, for example, will contain a dictionary with the keys being the community types and the values being dictionary collections of communities and their entry classes.

Parsers such as `cForumDBCommunityParser` will parse an XML for basic raw data pertaining to a specific community type - in this case forums. Each derived class from these parser classes will parse for additional raw information which might only be specific to a particular community.

Statistical Parser Hierarchy:

[Diagram [Coming Soon](#)]

28

The statistical parsers will take in raw data as input and output computed data based on the type of parser. The structure and implementation of the parsers is yet to be determined.

A Note on Server-Client Communications

For the following, let *send communications* refer to the sending of a message from the client application to the server and let *receive communications* refer to the messaging that originates at the server and is received by the client.

Server-client communication is needed in all of the following situations:

Send and receive communications:

- User submits information to register for a new account
- User attempts to log in to an existing account
- User selects a community group for viewing
- User creates a new community group

- User edits a community group by adding keywords or changing the icon
- User adds a new website to an existing community group
- User begins the edit process for an existing website
- User submits an edit an existing website
- User selects a website to view additional stats for

Every one of these can be thought of as "send and receive" communications. Under the cover, the every user-initiated action causes the client to submit a request via HTTP to the server to run a PHP script. This is the send message. The server then runs the PHP script and the output of this script is returned to the client. This output is the received message. Every "send" message uses the HTTP GET method to specify the values that the server-side PHP script expects to receive. Since the values of all GET arguments are specified as text and the length of each argument is also stored in the message, there are no problems with strange characters in these arguments. For "receive" formats, XML is used because this seems to be the encouraged data transfer format in Flex. XML manipulation is very easy, once some initial pitfalls are overcome, and this method of communication is very productive. However, there can be issues if the values of certain arguments cause the resulting XML to be malformed. This results from the use of '<', '&', and '"' which happen to be fairly common characters in the regular expressions which are retrieved from the server when a user chooses to edit a website. To avoid complicated find and replace routines needed to escape these misbehaving characters, we instead invented an ad hoc data transfer method used specifically in this situation. Using this method, the receive message is a string of variable-value pairs delimited by semi-colons structured as follows:

```
"{variable name #1}={value #1};{variable name #2}={value #2}; . . . ;{variable name #n}={value #n}"
```

When the client receives this message, it matches it against a regular expression and captures the values.

Is this method fool-proof? No. The arguments could be parsed incorrectly if the user happened to know what the names of our variables were or if they accidentally used one of the variable names in a regular expression. However, this is not likely as the variable names tend to be domain-specific and unlikely to occur in the text of a website. The only person that would be harmed by such an action would be the user which committed it, as their website would end up not having any useful statistics.

In addition to this invented message format, we also had to filter some of the text being returned in XML prior to packaging it because it is possible for the XML-unfriendly characters to also appear in thread titles, user names, URLs, and other data that is returned by the server. The filter routines simply remove the problematic characters and this has not produced any usability issues in testing so far.

VII. Graphical User Interface

The following two GUI's were presented to our liaisons. In December, we came to a consensus that the hierarchy was not correct. Instead of having "actions" as the main navigational tabs, they should have been "nouns". In other words, "communities" or "websites" should be the top level tab navigation and the things you can do to them should be included on the page after you navigate to them. Over the Christmas break, we must redesign our GUI structure to meet the liaison's needs.

31

VIII. All Project Files

Note: Directories in **red**.

The displayed hierarchy is the same as in our SVN repository hosted by Google Code.

- **monarch-client** - *Flex GUI*
 - **AdobeAnalyticsTool**
 - **bin-debug**
 - * AdobeAnalyticsTool-app.xml
 - * AdobeAnalyticsTool.swf
 - * cEngine.swf
 - **Documentation**
 - * Features
 - **images** - *These images are used in the GUI. Hint: search for "embed" in source files*
 - * add.png - *add a keyword to a community group*
 - * application_add.png - *add a website*
 - * application_cascade.png - *add a community group*
 - * arrow_left.png - *previous regex*
 - * arrow_right.png - *next regex*
 - * clock.png - *sort community groups by date added*
 - * door_in.png - *login*
 - * door_out.png - *logout*
 - * eye.png - *sort community groups by most viewed*
 - * help.png - *brings up regex screenshot help window*
 - * information.png - *detailed website stats*
 - * Internet.png - *view other people's community groups*
 - * magnifier_zoom_in.png - *decrease Y-axis extrema of line graph*
 - * magnifier_zoom_out.png - *increase Y-axis extrema of line graph*
 - * Misc.png - *my community groups*
 - * monarchLogo.png - *logo in upper left*
 - * pencil.png - *edit community group or website*

- * ps.png - *default community group icon*
- * text_uppercase.png - *sort community groups alphabetically*
- * thumbnailBorder.png - *border around all community group thumbnails*
- * user_add.png - *register for an account*
- **libs**
- **skin** - *kingnare skin downloaded from www.scalenine.com*
 - * **src**
 - **assets**
 - \$ **icons**
 - ¶ ai.jpg - *image used in sample, but we never use it*
 - ¶ air.jpg - *image used in sample, but we never use it*
 - ¶ fl.jpg - *image used in sample, but we never use it*
 - ¶ fw.jpg - *image used in sample, but we never use it*
 - ¶ fx.jpg - *image used in sample, but we never use it*
 - ¶ ps.jpg - *image used in sample, but we never use it*
 - \$ **skins**
 - \$ **xml**
 - ¶ listData.xml
 - Kingnare.mxml
 - **style**
 - \$ kingnarestyle.fla - *source for sample*
 - \$ kingnarestyle.swf - *sample*
 - \$ style.css - *stylesheet*
 - **src**
 - * AdobeAnalyticsTool-app.xml
 - * AdobeAnalyticsTool.mxml
 - * cAccordionKeywordGraphicCanvas.mxml
 - * cAccordionKeywordGraphicEvent.as
 - * cAccordionLinkGraphicCanvas.mxml
 - * cAccordionWebsiteGraphicCanvas.mxml
 - * cAccordionWebsiteGraphicEvent.as
 - * cAccountInformation.as
 - * cAdditionalWebsiteInformationEvent.as
 - * cAllPurposeEvent.as
 - * cAnalysisDataReadyEvent.as
 - * cAnalyticsParser.as
 - * cCommGroupGraphicClickedEvent.as
 - * cCommGroupLargeGraphic.mxml
 - * cCommunityGroupBrowserCanvas.mxml
 - * cCommunityGroupEditorCanvas.mxml
 - * cCommunityGroupEditorEvent.as
 - * cCommunityGroupEntry.as
 - * cCommunityGroupHomeCanvas.mxml
 - * cCommunityGroupSelectionCanvas.mxml

- * cCreateAccountEvent.as
- * cCreateAccountWindow.mxml
- * cCreateCommunityGroupCanvas.mxml
- * cCreateCommunityGroupEvent.as
- * cCreateWebsiteEvent.as
- * cCreateWebsiteWindow.mxml
- * cEditWebsiteEvent.as
- * cEngine.mxml
- * cInformationGraphic.mxml
- * cKeywordAnalyticsEntry.as
- * cKeywordEntry.as
- * cLabelEntryGraphic.mxml
- * cLineChartCanvas.mxml
- * cLinkAnalyticsEntry.as
- * cLoggedInCanvas.mxml
- * cLoggedInEvent.as
- * cLoggedInHomeCanvas.mxml
- * cLoggedOutCanvas.mxml
- * cLogginEvent.as
- * cNewCommunityEvent.as
- * cNewKeywordEvent.as
- * cNewWebsiteEvent.as
- * cNewWebsitesEvent.as
- * **com**
 - **kingnare**
 - § **skins**
 - ¶ **ComboBox**
 - † ComboBoxSkin.as
 - ¶ **NumericStepper**
 - † NumericStepperDownSkin.as
 - † NumericStepperUpSkin.as
 - ¶ **PopUpButton**
 - † PopUpButtonSkin.as
 - ¶ **util**
 - † DrawUtil.as
 - SampleTitleWindow.mxml
 - * Constants.as
 - * **controls**
 - CanvasButton.as
 - Highlighter.as
 - § **textClasses**
 - ¶ Finder.as
 - ¶ StringBoundaries.as
 - * cRegexHelpWindow.mxml

- * cRegularExpressionEditorCanvas.mxml
- * cSignInEvent.as
- * cSignInWindow.mxml
- * cStatOptionsAccordion.mxml
- * cStatOptionSelectedEvent.as
- * cWebsiteAdditionalStatsBrowserCanvas.mxml
- * cWebsiteDynamicLargeGraphic.mxml
- * cWebsiteEntry.as
- * cWebsiteGraphicClickedEvent.as
- * cWebsiteGraphic.mxml
- * cWebsiteLargeGraphic.mxml
- * cWebsiteStaticLargeGraphicClickedEvent.as
- * cWebsiteStaticLargeGraphic.mxml
- * cWebsiteStatsBrowserCanvas.mxml
- * RegexUtilities.as
- **monarch-server**
 - **Client** - *server scripts called by the client*
 - **database** - *unsure whether these files are even used anymore*
 - * website.txt
 - * website_temp.txt
 - **images** - *any link to an image that the GUI dynamically generates needs to be placed on the server because Flex's stupid security sandbox setting.*
 - * regexFirstPostAuthor.jpg - *screenshot of Threadless' first post author*
 - * regexFirstPostMessage.jpg - *screenshot of Threadless' first post message*
 - * regexFirstPostTime.jpg - *screenshot of Threadless' first post time*
 - * regexNextPageOfPosts.jpg - *screenshot of Threadless' button to see next page in thread.*
 - * regexNextPageOfThreads.jpg - *screenshot of Threadless' button to see next page of threads*
 - * regexNotFound.jpg - *shown when there is no regex help available for this step*
 - * regexReplyAuthor.jpg - *screenshot of Threadless' reply author name*
 - * regexReplyMessage.jpg - *screenshot of Threadless' reply message*
 - * regexReplyTime.jpg - *screenshot of Threadless' reply time*
 - * regexStartPage.jpg - *screenshot of Threadless URL in Firefox*
 - * regexThreadNumViews.jpg - *screenshot of Threadless' number of thread views*
 - * regexThreadUrl.jpg - *screenshot of Threadless' link to a particular thread*
 - * heart.png - *fluff shown on the homepage*
 - * homeGraph.png - *fluff shown on homepage*
 - * homeWebsiteScreenshots.png - *fluff shown on homepage*
 - 404error.html - *when you click on a link in the GUI that wasn't correctly loaded by a server script.*
 - ForumBaseSingle.php
 - RecordsInit.php
 - changeData.php

- communityIncrementView.php - increments community view count
- detailStats.php - general website stats and links to pages that talk about the keyword
- getAllWebsites.php
- getAnalytics.php
- getWebsiteSettings.php
- insertCommunity.php
- insertExistingWebsites.php
- insertWebsite.php
- login.php
- register.php
- setWebsiteSettings.php
- xmlWriter.php
- **database** - database generation and querying
 - Database.php - interface for querying any database in our system
 - communityanalysis.txt - SQL to generate master database
 - designbyhumans.txt - backup SQL for DesignByHumans
 - gizmodo.txt - backup SQL for Gizmodo
 - googleuspolitics.txt - backup SQL for Google US Politics
 - nytimespolitics.txt - backup SQL for NYTimes Politics
 - reset.txt - SQL to safely reset the master database or website database
 - sarahpalintwitter.txt - backup SQL for Sarah Palin Twitter
 - threadless.txt - backup SQL for Threadless
 - website.txt - SQL to generate a website's database
- **englishDictionary** - for our spell checker
 - englishDictionary.txt - all words of the English dictionary on separate lines
 - fileToDatabase.php - inserts all words of the English dictionary into a Database (not used anymore because we just read directly from the txt file now).
- **scheduling**
 - scrapeOrderer.py - a script that runs forever. It calls other PHP files to fill in stats for all websites.
- **setup** - an AJAX website version of the GUI. Scrapped because Client wanted a Flex solution
 - **images** - have no functional effect; only makes the website look good
 - * background.jpg - the smoky futuristic background and logo
 - * formTextBackground.png - gives text boxes a concave look
 - * loading.gif - load bar when waiting for AJAX roundtrip
 - * spacer.gif - clear GIF for tweaking positions of elements
 - **scriptaculous** - Javascript animation library (contents not shown)
 - ajax.js - requests for server text without reloading the website
 - base.css - basic styling common to all websites
 - createCommunity.php - create a community group
 - home.php - page after you sign in
 - index.php - allows you to sign in or register
 - insertCommunity.php - inserts a community group into database
 - insertRegexes.php - store regexes in database

- insertWebsite.php - insert a website under a community group
- loadPostsHtml.php - loads a thread's HTML contents
- loadThreadsHtml.php - loads a page of multiple thread's HTML content
- login.php - logs in user into the website
- master.css - specific style of the website
- matchRegex.php - loads the HTML that the regex matches
- myAccount.php - edit your communities and websites
- regexEditor.php - asks for various regexes to specify the adapter for a website
- register.php - register for an account before you can login
- reset.css - browser compatibility styling
- submitCommunity.php - parses the HTML form for creating a community
- submitWebsite.php - parses the HTML form for creating a website
- **tests** - testing that our code is valid
 - binarySearching.php - verify that BinarySearch works correctly on English dictionary.
 - blogArticleTest.php - verify that sentiment metric works on Vera's Flex articles.
 - linguisticTest.php - verify that sentiment and English proficiency work on short snippets of text.
 - sentimentTester.php - allows you to enter any text and keyword you want and have the sentiment value be calculated.
 - talkTest.php - shows the top 5 threads/users that talk positively/negatively about a keyword.
 - test.php
 - test1.php
 - testcrawl.php
 - testcrawl2.php
 - yat.php
- BinarySearch.php - logarithmic scanning of the English dictionary, good words, bad words, inverters, and amplifiers.
- ForumPostProcessor.php - analyzes a page of posts
- ForumThreadProcessor.php - analyzes a page of threads
- Linguistics.php - sentiment and English proficiency algorithms
- imageUpload.php - upload a thumbnail for a community
- PhotoManip.php - generic library for manipulating an uploaded image
- Processor.php
- RedditProcessor.php
- StructuredCrawl.php
- Url.php
- constants.php - various crawling settings and common regexes
- **design** - Photoshopped layouts
 - **wireframes** - mockups sketched by hand
 - 1.png - for user test #3
 - 10.png - for user test #3
 - 11.png - for user test #3
 - 12.png - for user test #3

- 13.png - *for user test #3*
- 14.png - *for user test #3*
- 15.png - *for user test #3*
- 16.png - *for user test #3*
- 17.png - *for user test #3*
- 18.png - *for user test #3*
- 19.png - *for user test #3*
- 2.png - *for user test #3*
- 20.png - *for user test #3*
- 21.png - *for user test #3*
- 22.png - *for user test #3*
- 23.png - *for user test #3*
- 3.png - *for user test #3*
- 4.png - *for user test #3*
- 5.png - *for user test #3*
- 6.png - *for user test #3*
- 7.png - *for user test #3*
- 8.png - *for user test #3*
- 9.png - *for user test #3*
- AdapterEditor-finalScreen.PNG - *for user test #2*
- AdapterEditor-finalScreen.xml - *for user test #2*
- AdapterEditor-openingScreen.png - *for user test #2*
- AdapterEditor-openingScreen.xml - *for user test #2*
- AdapterEditor-transitionScreen.PNG - *for user test #2*
- AdapterEditor-transitionScreen.xml - *for user test #2*
- AdapterEditor-viewSelected.png - *for user test #2*
- AdapterEditor-viewSelected.xml - *for user test #2*
- Mariusz's_Revised.psd - *for user test #1*
- adapterEditor.doc - *all of user test #2's adapter editor images on one page*
- individualFrames.ppt - *all of user test #3 images in one powerpoint*
- process.psd - *arrows drawn between user test #3's slides showing what takes you where*
- wireframes.png - *exported version of wireframes.psd*
- wireframes.psd - *revised version of process.psd*
- wireframes2.png - *exported version of wireframes2.psd*
- wireframes2.psd - *revised version of wireframes.psd*
- adobe logo.ai - *vectorized Adobe logo*
- layout graphics.ai - *background and logo graphics for original website*
- master ER diagram.psd - *master database's entity-relationship diagram*
- mockup1.psd - *layout of the original website*
- mockup2.psd - *layout of a proposed Flex GUI*
- monarch butterfly.ai - *vector graphic of the final Monarch logo*
- monarch butterfly.psd - *final Monarch logo with gradient and drop shadow*
- userTestingSuggestions.txt - *suggestions from Adobe representatives on user test #3*
- website ER diagram.psd - *entity-relationship diagram of a website's database.*

XI. Building the solution

32

Server Solution: since all server software is written in PHP, an interpreted server-side language, there is no traditional build process. However, it is required that the server have Apache installed and running as well as PHP. Also, the server software must be in the correct path.

Client Solution: the easiest way to build the Monarch client from source is to acquire Flex Builder, install the Subclipse plugin (see how here: <http://www.flexer.info/2008/04/30/how-to-install-subclipse-on-flex-builder-3/>), and checkout the project. You may then click the run button or the debug button to build and run the application.

- 1 correct me if my neuroscience is wrong.
Ryan Lin May 10, 2009 11:04 PM
- 2 WARNING: Ryan wrote this paragraph, not me!
Andrew Spencer May 11, 2009 10:41 PM
- 3 this paragraph is very wordy
Ryan Lin May 12, 2009 1:43 AM
- 4 wasn't this stuff said in the "working remotely" section above?
Ryan Lin May 12, 2009 1:51 AM
- 5 TODO
Andrew Spencer May 12, 2009 4:15 PM
- 6 not dramatic enough. Needs more action and military terms.
Ryan Lin May 12, 2009 6:57 PM
- 7 Mariusz, explain why you couldn't do this and tell them which file to edit to complete this feature.
Ryan Lin May 8, 2009 2:36 PM
- 8 We should mention files to modify in this part too.
Andrew Spencer May 11, 2009 11:18 PM
- 9 fix
Ryan Lin May 3, 2009 11:04 PM
- 10 todo
Ryan Lin May 3, 2009 11:16 PM
- 11 Andrew, clarify this.
Ryan Lin May 3, 2009 10:40 PM
- 12 I don't know where, don't know how.
Ryan Lin May 3, 2009 11:00 PM
- 13 Okay. For this part I think I want to try to setup everything we need from a bare RedHat install.
Andrew Spencer May 5, 2009 4:44 PM
- 14 fix
Ryan Lin May 3, 2009 11:21 PM
- 15 clarify
Ryan Lin May 3, 2009 11:21 PM
- 16 need full path
Ryan Lin May 12, 2009 2:02 AM
- 17 are we ever going to do this?
Ryan Lin May 12, 2009 2:03 AM
- 18 we did not finish this
Ryan Lin May 12, 2009 2:04 AM
- 19 It looks like our dictionary contains many forms of the same word (punch, punches, punching,...). But definitely not our goodwords / badwords. That's why we need to stem the "adjectives" to match out good/badwords databases.
Ryan Lin Jan 12, 2009 6:47 PM

- 20 Since most of the keywords will be proper nouns, stemming should be mostly unnecessary. There are exceptions (I just 'googled' blah blah blah) but I don't think we need it.
Andrew Spencer Dec 11, 2008 11:56 PM
- 21 "scrapping" misspelling.
Ryan Lin May 10, 2009 7:58 PM
- 22 I have this working as I described. Using the callback to define processing seems to be powerful enough to do whatever we want to do. Separating crawling from processing like this should make it easier to use multithreading, pipes, or something else to overlap network latencies and the stats-gathering/ database operations in the future.
Andrew Spencer Nov 2, 2008 7:41 PM
- 23 Changed from original design:
Instead of using a callback we implement a Processor interface. This allows us to maintain state.
Andrew Spencer Dec 11, 2008 2:37 PM
- 24 label ER diagram lines
Ryan Lin May 11, 2009 1:19 PM
- 25 label ER diagram lines
Ryan Lin May 11, 2009 1:19 PM
- 26 I think you could find this to be easier if you use AMF and strongly typed classes instead of XML as the data transfer. That is, assuming the analysis work is done on the backend (pre-Flex).
Matthew Chotin Oct 20, 2008 6:09 PM
- 27 Using it for all communication tends to be cumbersome and it makes the code hard to follow as you have to jump from file to file up the event stack to see how the event finally arrives. In theory, it is a good idea but in practice it didn't really work.
Andrew Spencer May 7, 2009 9:13 PM
- 28 This was phased out because the final design only needed one "parser"/data class.
Andrew Spencer May 7, 2009 9:15 PM
- 29 These statistical parsers are client-side? What might one do?
Andrew Spencer Oct 27, 2008 11:54 AM
- 30 make it come soon
Ryan Lin May 10, 2009 11:20 PM
- 31 This needs to be revised to discuss our final GUI design.
Andrew Spencer May 10, 2009 1:03 PM
- 32 move this to client installation section
Ryan Lin May 12, 2009 0:39 AM