Team #4
Product Backlog
8/30/2013

Authority-Based Publishing Platform

Andrew Mack, Christina Atallah,
Jason Salter, Nanthaniel Cherry,
James Miller, Mitchell Mounts

Team #4
Product Backlog
8/30/2013

Problem Statement

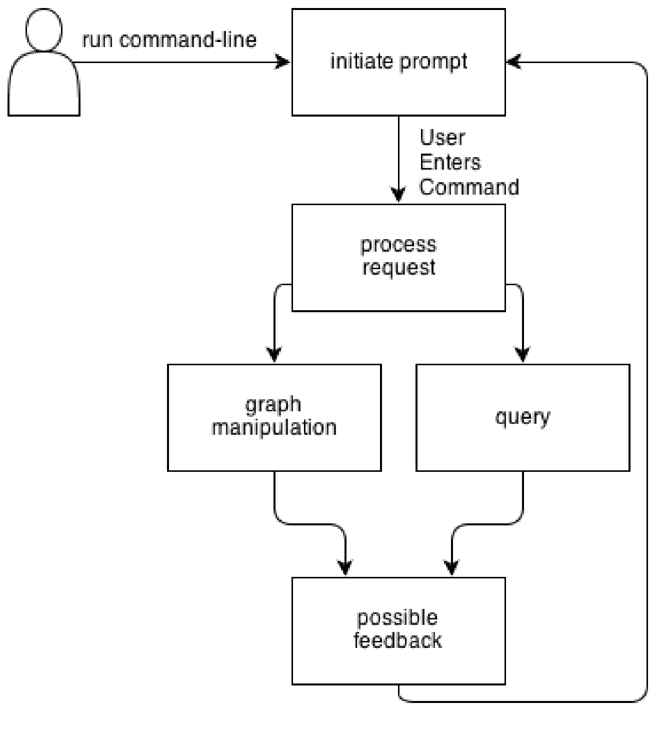
Our project involves developing an authority based publishing system that focuses on author involvement and credibility in a distributed environment. When having academic discourse on certain topics, it's important that the author and subsequent discussion have some minimum level of credibility. In turn, having authors of any topic openly display their credibility and authority enables humans and machines to evaluate both the discussion and authors familiarity with a subject.

Background Information

There is a lot of information on the internet, and it is easy to get caught up in the world of mass information. Often times it is easier to trust a stranger on his word, then to check his sources. The ability to reference the validity of a person's words simultaneously with the content would be a huge benefit. In our project, we hope to make first steps to solving this problem. The ability to build credibility networks is as valuable as it is practical.

Environment and System Model

The library will be written in Java and accessible as a web service. The graph will be maintained through the use of Neo4j. Users will interact with the library through command-line API requests, as described by the figure below.

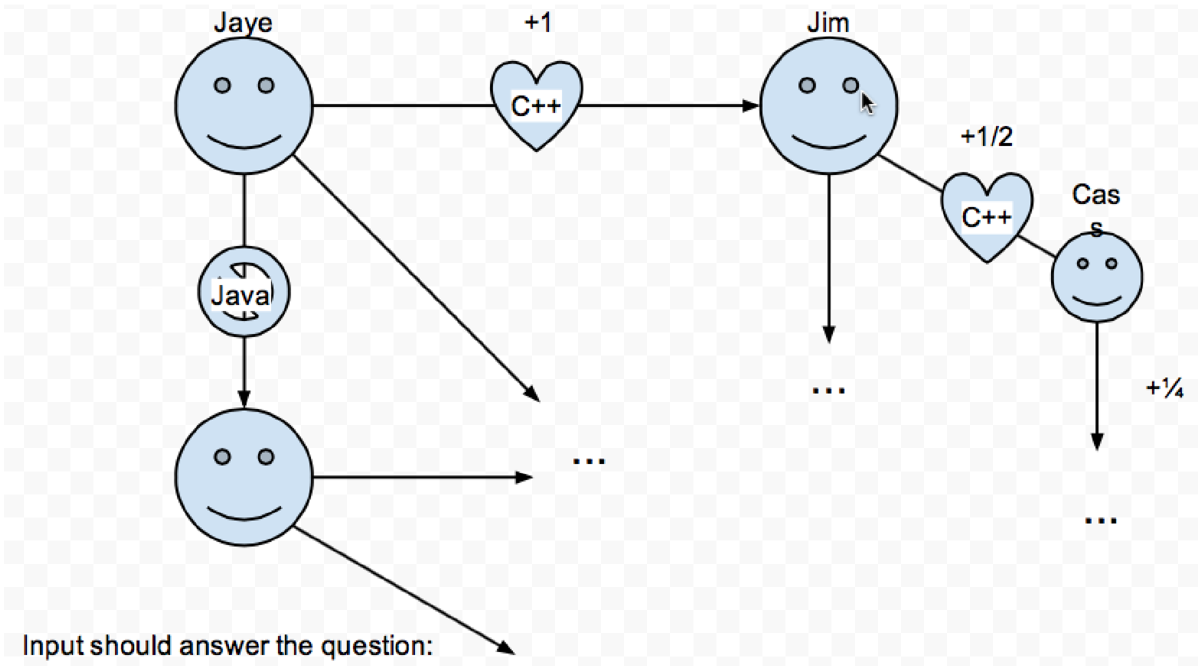


Team #4
Product Backlog
8/30/2013

<u>Functional Requirements</u>

1. **Interaction between user and command-line**
    a. ability to start and stop prompt
    b. ability to enter commands
    c. useful post-processing feedback

2. **Interaction between user and graph database**
    a. ability to sign-up and register
    b. ability to add and remove email address
    c. ability to login and logout
    d. ability to make changes to the graph via endorsements
    e. ability to query users by identifier
    f. ability to view networks



Input should answer the question:

<u>Jaye</u> trusts **<u>Jim</u>** about **<u>C++</u>** because of **[<u>citations</u>]**

____ trusts ___ about ____ because of [____, ______, _____....]

Team #4
Product Backlog
8/30/2013

Non-Functional Requirements

1. **Reliability**
   The graph should update to reflect the changes made to it accurately and without issue. It should also update in a timely manner in order to avoid one user from receiving incorrect information while another user is updating it. We will be using Neo4j as a base for creating our graph in order to accommodate these concerns.

2. **Scalability**
   The graph should be able to grow without using up too many resources. When the graph grows and users are endorsed, we will have a trickle down effect. In order to keep the amount of time needed to reflect changes down, we will institute a lower limit for the trickle effect to continue. We will also want to avoid potential infinite loops from occurring.

3. **Usability**
   The API should be intuitive. Any prompt feedback should be coherent and descriptive.

Use Cases

1. **User wants to sign-up**
   a. user calls the sign-up function with email
   b. sends user token via email

2. **User wants to register**
   a. user calls the register function with username, password, and token
   b. database adds the user

3. **User logs in**
   a. user calls the login function with username and password
   b. user is given an active session

4. **User wants to add an email**

   a. user calls the add email function
   b. adds email to user

5. **User wants to remove email**
   a. user calls the remove email function
   b. removes email from user

6. **User wants to add citation to portfolio**
   a. user calls the add citation function with new citation
   b. adds citation to user portfolio

7. **User wants to remove citation from portfolio**
   a. user calls the remove citation function with citation
   b. removes citation from user portfolio

8. **User endorses someone**
   a. user calls the trust function
   b. database updates graph to reflect the user's endorsement

9. **User de-endorses someone**
   a. user calls untrust function
   b. database updates graph to reflect the user no longer endorses that person

10. **User logs out**
    a. user calls the logout function
    b. user session ends

11. **User wants to see a network**
    a. user asks to view network by function name
       i. subjective network vs public network
    b. database checks for active session and returns appropriate network