Homework #1

CECS 530

Dr. Ragade

January 21, 2014

Team Wikiviz

Sarah Mullins

Everett Rush

Ashley Revlett

**Task I. Team Experience**

Ashley Revlett: In my experience as a web designer and project manager, I usually followed the Iterative Waterfall method. I would gather requirements at the beginning of the project, flesh them out as completely as I could at the time, and my team would proceed with design and development like a traditional Waterfall model. When we realized certain design aspects or functionality were not workable as intended because of technical, usability or budget reasons, or when the client changed their mind about a feature or design, we would revise the requirements, then go back to revise a part of the design, then recode whatever necessary. As these revisions started to add up, the requirements document was not always updated in lock-step and quickly became out-of-date, as the developers were more focused on coding than documentation. In the end it was a challenge to ensure all requirements were met because the requirements were always evolving, even during development, and not necessarily well documented.

Everett Rush: I have not had an experience with a software lifecycle. A SRS sheet was a new term to me when I first encountered it in this course.

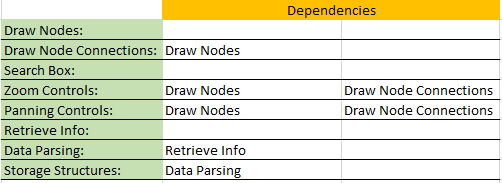
Sarah Mullins: I also do not have experience with a software lifecycle.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **None** | **Iterative Waterfall** | **Agile** |
| **Ashley** |  | X |  |
| **Everett** | X |  |  |
| **Sarah** | X |  |  |

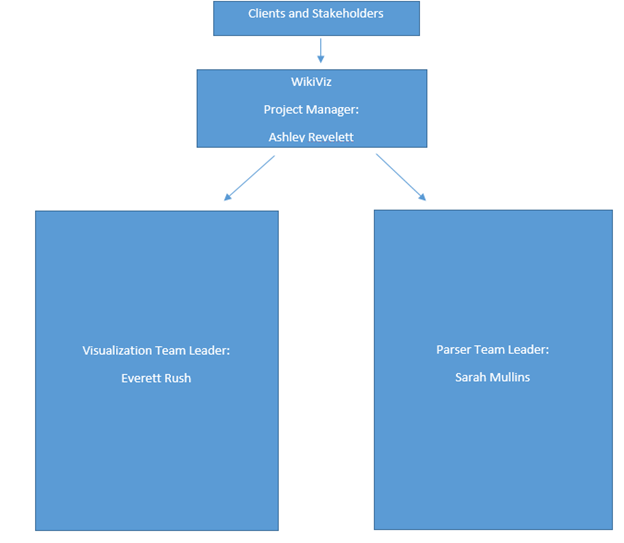
**Task II. Preliminary Research**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name and Link** | **Description** | **Release Date** | **Platform** | **Pros** | **Cons** |
| [**WikiNodes**](http://www.idea.org/WikiNodes.htm) | Visually-driven, interactive graph-based Wikipedia | 2011 | iOS (iPad) | Well-designed and currently active. Full featured (multilingual, annotations, bookmarks, sharing) | iPad only.  Occasional slow response times in the UI. |
| [**Wikistalker**](http://sepans.com/sp/works/wikistalker/) | Visualization of the meta-structure of Wikipedia articles | 2011 | Web (HTML5/JS) | Novel visualization of relationships between articles. | Interesting way to explore, but not very useful as article content isn’t shown, only meta-structure. |
| [**The Web Stalker**](http://www.visualcomplexity.com/vc/project_details.cfm?index=7&id=7&domain) | Experimental web browser that uses graph-based representation of web pages and links | 1997 | Downloadable Program (Windows, Mac) | Interesting visualization of information. One of the first in this area. | Outdated. Non-wikipedia related. |
| [**Ask Ken**](http://datavisualization.ch/tools/ask-ken/) | Node-Link diagram that allows to visually navigate through interconnected topics provided by the Freebase Service | 2010 | Web (Ruby, JS) | Unique “ring chart” visualizations. Beautiful design. | Buggy. Not usable on Mac. Not based on Wikipedia content. |
| [**IndyWiki**](http://indywiki.sourceforge.net/index.html) | Visual Wikipedia Browser | 2008 | EXE (Python, QT) | Good concept. Runs on all OSes. | Terrible design. Looks outdated. |
| [**WikiMindMap**](http://www.wikimindmap.org/viewmap.php?wiki=en.wikipedia.org&topic=Wolfgang_Amadeus_Mozart) | Interactive mind map visualization of Wikipedia | 2007 | Web | Mind map visual is useful. | No graphics, no pictures. Just text. Limited ability to click down into articles. |
| [**Liveplasma**](http://www.liveplasma.com/#/artist/Wolfgang_Amadeus_Mozart/255b884c-25e9-4da2-99e9-b104ed637eae/US) | Graph-based visually-driven search engine | 2004 | Web (Flash/Flex) | Good use of animation, audio, and interactivity. | Can search only music. Non wikipedia-based. |

**Task III:**



**Task IV: Organizational Hierarchy and Responsibilities**



The project manager is the team leader’s leader. As such she is responsible for the success of the software project. She shall hold team leaders accountable for meeting deadlines. She shall apply specific metrics to evaluate the project at each sprint. Then she will provide specific feedback to each team.

Additionally, she is the principle mediator between the team leaders, stakeholders, and the clients. Therefore, she is responsible for ensuring each meeting occurs, and that all parties involved are present. She is responsible for ensuring each meeting is productive and on task with a clear agenda. Upon meeting with clients, she shall immediately inform the proper teams on the way the project should evolve.

Each team leader is responsible for the documentation, for module cohesion, and for the testing of all code under their supervision. Team leaders are expected to make sure their team is on track to meet deadlines. Team leaders are expected to communicate with the project manager at least on a weekly basis. A team leader must report each substantial problem to the project manager as quickly as possible.

b) We will be using an Agile Lifecycle to produce our software. Each two week sprint will have a design phase, a development and implementation phase, a validation phase, and an evolution phase. We will prioritize the functionalities from most important to least. The first meeting in the two week sprint will cover evolution from the previous week’s sprint and design for the next sprint. The second meeting will be used to review the results of testing the implementation and for receiving client feedback.

c) Obviously WikiViz is a small team of three. Accordingly, if one member of the group is unavailable for even a short period of time, that inconveniences the other group members. We will keep all of our documentation on Google drive so that every member has access to the work of every other member. We will keep all code on Git. Depending on the nature of the missing member/s unavailability, the remaining team member/s will attempt to cover the responsibilities of the missing member/s.

d) We will measure the code by module cohesion and cyclomatic complexity. That is we will be measuring our code by how well code is grouped together. Code that is relevant to other code must be logically grouped together somehow.

We will measure the success of the user interface by its ease of use, aesthetics, reliability, and correctness. Ease of use is defined by the ability of a user with no prior experience with the software being able to navigate the application with ease. Aesthetics is defined by a visually compelling product with a clear picture. Reliability means the software should not crash. Correctness is a measure of the information presented being the exact same as its source.

**Task V: Schedule and Estimate**

Our schedule includes:

* 1 two-week initial research and “big picture” design phase
* 4 two-week development sprints, focused on four increasingly refined iterations of the application
* 2 two-week testing sprints, for refining the optimizations made during development, and complete component, integration and end-to-end testing

We used an agile method of allocating story points to form a time estimate. Our total estimate for this project is 191 hours. For a detailed list of user stories, business priorities, tasks, responsibilities, and hourly estimates per task, please see the attached Project Schedule spreadsheet.

**Task VI: Version Control**

The group chose to use Git for version control. We are using Git because everyone is familiar with the services that Git offers and because it is free, fast and reliable. Git is accessible from any computer with an internet connection. It does not require any software beyond an internet browser, and it does not have to be configured beyond the read/write permissions. While it is easy to use, it does not have all the features of Bugzilla or other version control software. But we do not anticipate needing a version control system for anything beyond open access to the code. So we chose Git because we could get the version control system working immediately, and then we could get to coding quicker.

The version control software, Bugzilla, has many features that make communication easier and increase productivity, but using Bugzilla would require a lot of overhead. No one in the group is familiar with Bugzilla. We would need everyone in the group to install the software and figure out how to set up the database. Then we would have to form a team to administer the database. The scope of our project and expertise makes Bugzilla a poor choice for version control.