**Collaborative Edit:**

A Collaborative Package for the Atom Text Editor

**Prepared For**: Michel Barbeau, COMP 3203

**Authors**: Eric Adamski, 100873833 and James Roose, 100852693

**Date**: November 22, 2014

**1. Introduction**

* 1. Problem Statement

Creating a program of significant size often requires the work of a team. In order to work most efficiently, members of a team must be able to collaborate on tasks and issues, even when unable to physically gather all at once. Therefore, in order to increase the efficiency of the team, collaborative software is necessary to coordinate the asynchronous contributions of individual team members in an organized fashion. We aim to solve a small subset of this larger problem, by creating a collaborative environment in which multiple developers can simultaneously work on a task or issue within a document.

1.2 Project Description:

The goal of our project is to create a package for the Atom text editor that will allow the use of collaborative editing within a document. This package is intended to allow for fluid transfer between a local and collaborative editing environment that is capable of supporting any number of users while maintaining document structure. Therefore, features such as syntax highlighting and IDE quick editing capabilities should remain intact even while multiple users interact with a document. Additionally, should time allow, we aim to add several ‘flavor’ features to enhance the usability of our final product (for example, the ability to see each other’s cursor within the shared document).

1.3 Background/Context:

Collaborative editing has played a large role in the success of the programming world through the use of version controls, such as Git, or the use of other online collaborative tools, such as Google Docs. These tools are sought out because they remove the need for a group to assemble and increase the time efficiency of a project by allowing individuals to asynchronously and seamlessly combine their efforts.

So, how does collaborative editing work? The implementation of collaborative software is actually very complicated due to factors involved in network communication (ex. latency) as well as local hardware communication (ex. read/write speed of a HDD or SDD). However, each of these issues have known solutions that fall under a set of algorithms know as Operational Transformation (OT) algorithms.

OT is a set of frameworks that encompass multiple algorithms that are meant to handle the quick storage and retrieval of information. These frameworks include schemas on data precedence, function precedence, permissions and much more. Essentially, it can be represented as a Git or SVN repository that executes fetches, pulls, commits and pushes every few seconds. This process allows for a collaborative setting in which the most recent changes made to a document are regularly made available to all users.

1.4 Tools For The Job:

Programming today is rarely about building from the ground up; why reinvent something that has already been done? With this philosophy in mind, we wanted to create a meaningful solution to a relevant problem. In order to accomplish this, we made use of three important technologies.

1. NodeJS:

NodeJS is an open source and cross-platform runtime environment for server side and networking applications. It is a Javascript application that provides an event-driven architecture and a non-blocking I/O API aimed at optimizing an application’s throughput and scalability. NodeJS serves as the foundation upon which ShareJS, our next important technology, builds upon

1. ShareJS:

ShareJS is a Javascript implementation of an OT Protocol as a simple server/client library. ShareJS takes advantage of the event-driven and non-blocking nature of NodeJS to allow users to collaboratively edit documents and arbitrary JSON data in real time.

1. Atom:

Atom is a completely open source and ‘hack-able’ text editor that allows the use

of custom packages. With the implementation of ShareJS, we aim to create a custom package for Atom that will enable collaborative editing within a document.

In addition to the tools listed above, we utilize web sockets to connect each part and create a single usable package for local-collaborative editing. Lastly, the socket API we use is a javascript wrapper for C system sockets.

Result Summary:

In summary, we feel we have succeeded in achieving our goals. The Collaborative-Edit package for the Atom text editor allows multiple users to simultaneously interact with a document with minimal issues. However, some bugs do still exist, which we speak towards in section 2.3 of this document. We have also succeeded in implementing a few quality of life features, such as the ability to see other user’s cursors in a document.

TODO: Better Summary

**2. Project Analysis**

2.1 Implementation:

TODO

2.2 Analyses and Discussion:

TODO

2.3 Problems and Improvements:

TODO

**3. Conclusions**

TODO

**4. References**

TODO