Spring 2012

Link IDE : A Real Time Collaborative Development Environment

Kevin Grant
San Jose State University

Link IDE : A Real Time Collaborative Development Environment

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Kevin Grant

May 2012

SAN JOSE STATE UNIVERSITY

The Undersigned Project Committee Approves the Project Titled

Link : A Real Time Collaborative Development Environment

by
Kevin Grant

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE
SAN JOSÉ STATE UNIVERSITY

May 2012

Dr. Soon Tee Teoh, Department of Computer Science

Date

Dr. Chris Pollett, Department of Computer Science

Date

Prof. Frank Butt, Department of Computer Science

Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research

Date

ABSTRACT

Link: A Real Time Collaborative Development Environment

by Kevin Grant

While working on large scale development projects all software engineers find themselves, at some point, working with a source control system in order to add or revert changes to code. With software projects involving a multitude of programmers this is a crucial part of successful development. When working on a smaller project however, with a tight knit group, setting up and dealing with such a system can become more work than it is worth. To solve this problem a real time collaborative integrated development environment could be used. The IDE's focus would be on providing a collaborative setting for programming teams or pair programming by taking advantage of real time text editing, the ability to build and run code, chat, and various other team and task oriented features. Instead of running into code conflicts at check-in time, users would be able to see conflicts appearing in real-time. This would allow small programming teams to bypass source control, avoid wasting time, and spend more time collaborating. Real time text editing has recently become popular with its appearance in Google Docs. There are a number of open source applications that support real time text editing. Real time editing by multiple users allows not only for excellent collaborative programming but can also be very effective in teaching sessions of programming. Other features such as chatting and task lists would also help to create a fully immersive and organized collaborative environment where users do not need outside tools in order to collaborate.

ACKNOWLEDGEMENTS

I would just like to say thank you to all the people who have helped me with my project. Thank you very much Dr. Soon Tee Teoh. You have been of tremendous help and guidance the past months. I would like to thank Dr. Chris Pollett and Frank Butt for agreeing to lend their time and be a part of my committee. I would also like to thank my friends and family for the encouragement. Thanks to all the random people on the internet who have offered wisdom. I would like to thank the three users who helped me test run this program. God knows they spent a lot of time dealing with the bugs that so inevitably lurk in a program such as this.

Table of Contents

1. Introduction

1.1 Introduction

Distributed computing involves multiple computer systems communicating with each other through a network to achieve a common goal. As distributed computing, also known as cloud computing, becomes more popular more applications are moving towards accessible and flexible network based designs. As applications begin to take advantage of these designs, systems are beginning to cater to the user, offering on demand access to data as well as instant communication with other users. One of the advantages of these types of systems is the ease at which collaboration can occur between users. Developing your application with a cloud based design can eliminate factors limiting collaboration such as distance or time zone [2]. Applications, such as Google Docs, have introduced real time collaborative editing to users and have become widely successful. These new methods collaborative lead to effective teamwork and an improved product. Given these trends it's likely that more developers will not only begin coding more cloud applications but that these applications themselves will be coded in the cloud as well [1]. The focus of my project is on the creation and analysis of an IDE which takes full advantage of this ideology and the collaborative benefits it provides. The program's development title is Link IDE.

1.2 Justification

Although for smaller groups of programmers it may not be ideal, currently, the best way to collaborate on a software project is with the use of a source control system. There are a few ways to setup a source control system and a few options for that. Common choices are SVN, CVS, and a now popular system named Git. CVS and SVN

require setting up a central location for version control info and everyone must connect and write there. Git offers an alternative by having a simple .git folder in your project directory which each user writes to and then can later merge with another users'. The key thing to understand here is that in order for a user to have source control in their project they must understand source control. They have to spend time researching which program is most suitable for them, how to set it up, and how to use it. Of course this is trivial work in a large software development team where the main project is in development for many months. But consider a small group of people, maybe even a pair of programmers. They have been assigned a programming assignment. They each have their own laptop and they want to start coding. If they wish to work effectively as a group their only option, other than merging manually, is a source control system. If it is their first time using a source control system they have to take the time to set it up. This may take a very long time for someone to figure out the first time. And imagine the project is not very large and with not that much coding necessary. Setting up the source control might end up taking almost as much or even more time than the project. But even with the system already in place, time can be lost using these types of systems. Often times conflicts in code can occur. Imagine again the pair of programmers. They are working primarily on one file in the project. They wish to add random implementations here and there as they are working side by side. They may overlap certain sections of code and form conflicts. These conflicts may or may not resolve at the time of checking in. This could lead to potentially frustrating problems and more time spent on source control related issues. The thing once again to understand is that for these two programmers a source control system is not ideal but it is the only option. They spend too much time setting it up, taking occasional breaks to check code in, and possibly spending time re-

solving conflicts. But what other option do they really have? This is where my project would be most useful. In small scale projects setting up a source control system can become more work than it's worth. The constant reminder that you have to check in code to avoid conflicts and to distribute code to teammates can completely disrupt the programming flow. In my program, the pair of programmers would not need to spend any time setting up a system. They would be able to bounce ideas off each other in real time by writing code and seeing each other's edits. Collaboration on a project like this should be a simple process that should only increase productivity and not waste time. There are no conflicts in this system as conflicts arising can be seen in real time by the users.

The use of a real-time collaborative IDE where users can write and build code simultaneously would greatly enhance programming teaching/tutoring sessions. If you have ever tutored programming, especially remotely, you may have already thought about this sort of idea. The tutor and student would connect to each other and start a project. The tutor can showcase some ideas and programming without having to be at or type on the computer with the student. When the student makes a mistake the tutor can quickly fix it while the student continues. This type of instant feedback can greatly increase the effectiveness of a tutoring session.

Aside from how real-time editing can be a valuable substitute for source control and a valuable addition for tutoring, Link IDE's other collaborative features also make it a good development choice. However, one cannot simply add synchronous editing to an IDE and think of it as a collaborative environment. Multiple users editing the same document sounds extremely chaotic and when working with delicate structures like programming projects it can get very messy. To organize the collaboration there are a

number of major features including a task list, personalized user names, network storage of project, and chat features. Web storage of code is especially useful because it gives developers access to code and offers a less expensive build infrastructure and opens the potential for an expandable development [1]. The combination of all these features, which will be explained in detail later, truly forms a collaborative environment which should be a common choice of development for programming projects today.

2. Related Work

As a part of my project it is important to examine the already existing implementations of a collaborative integrated development environment (IDE). I decided to showcase two browser based implementations and then a plug-in that has been developed for the very popular Eclipse IDE.

2.1. Browser Based Collaborative IDEs

2.1.1. eXo Cloud IDE

eXo Cloud IDE is a browser based development environment that allows for the collaborative development of applications that can be deployed directly to a Heroku PaaS environment [11]. Heroku is a very powerful web based platform for Ruby, JavaScript including node.js, and they have recently added Java web app support. They boast that deploying directly within a PaaS environment allows for quick migration from the development stage to deployment [11]. It also contains a real time collaborative editor for use with up to five people.

2.1.2. Cloud9 IDE

Cloud9 IDE is a web based IDE for Javascript and Node.js applications along with HTML, CSS, PHP, Java, Ruby and 23 other languages [12]. Cloud9, like eXo Cloud, is a browser based implementation of a collaborative IDE. Cloud9 IDE is primarily for Node.js and Javascript developers. Cloud9 has many of the features available in popular IDEs such as syntax highlighting, the ability to run and debug code, and keyboard shortcuts [12]. It has formed a fairly large user base by now and is pretty successful.

2.1.3. Comparison to Link IDE

Obviously it is hard to compare my own project which has been worked on by only myself in a short span to implementations that have been in development for years with many developers on staff. However just as some of their features are advantageous to my approach, I do feel that some of the differences between my implementation and theirs can highlight the benefits in my choices. Although browser based solutions are inherently more accessible, I feel many users might find it awkward to work in an only browser setting. There are also various limitations to working in a browser. Browsers can be very limited when it comes to performing computation on a local level and thus rely on servers to process much information [14, 364]. Mark Silver describes the various usability issues with browser based applications in “Browser-based Applications: Popular but Flawed?”. These include ambiguity between browser functions and the application functions (Back button), performance issues regarding screen updates, reliance on page orientation, and the inherent statelessness of web pages [14]. Cloud9 IDE and eXo Cloud IDE may or may not suffer from these problems. One

thing worth noting is that these IDEs focus mainly on web based development. Development of more performance intensive code such as mobile apps, graphic related applications, or even artificial intelligence type programs that train and predict while using intense processing might definitely require a desktop application based IDE which takes full advantage of available computing power. Although the browser based model is useful for web based development I do believe that for many situations a more traditional desktop application might be preferred as the model for a collaborative IDE.

2.2. Eclipse Communication Framework Project

The Eclipse Communication Framework project (ECF) is a project aimed at creating distributed applications for Eclipse. One part of this specifically is the Cola plug-in which allows for synchronous editing between users in Eclipse [13]. The Cola plug-in can be seen as one very important step in creating a collaborative IDE. Eclipse is already such a developed IDE that millions of people enjoy using for development. It is an incredible step forward to begin to integrate these types of features into Eclipse. The feature is at this time only a plug-in that must be downloaded, installed, and configured to work with Eclipse. Although it is a very useful plug-in it is only one step in the process of collaboration. I included features such as a task list and chatting in my program along with the synchronous editing so that users could organize their work in an efficient way. Having a task list allows the collaboration to live on even when there are not multiple users currently signed in. I think the Cola plug-in is a remarkable tool but

it does not necessarily qualify the Eclipse IDE as a collaborative environment.

3. Research Topics

3.1 Real Time Collaborative Editing

A key feature of the my project's implementation was allowing users to simultaneously edit a document and see each other's edits as they appear. This is known as real time collaborative editing (RTCE). RTCE has a long history and actually appeared first in 1968. For over 40 years it was largely overlooked as an important tool because of performance issues. With the advent of Web 2.0 and web applications such as Google Docs, RTCE has become a very beloved feature among teams. Google docs is the most successful real time collaborative editor so far. It has revolutionized the way students work on group reports, excel sheets, or presentations. This type of collaborative editing has been slow to be adopted by programmers. Recently the code behind Etherpad, the company who implemented the technology that serves as the basis for Google Docs' RTCE, was made open source. This, along with many other advances in the field, has opened the door for the development of new RTCE applications. As a result more and more applications dealing with word processing have added this feature [1]. The two most common and efficient means of implementing RTCE are two methods known as Operational Transformation and Differential Synchronization.

3.1.1 Operational Transformation

Operational transform is the breaking down of each action inside a text editor (insert character,delete,tab,enter) into a series of operations which are

each transformed to conform to the operation preceding them [4,472]. To illustrate this, the following rather simple example can be used, See Figure 1.

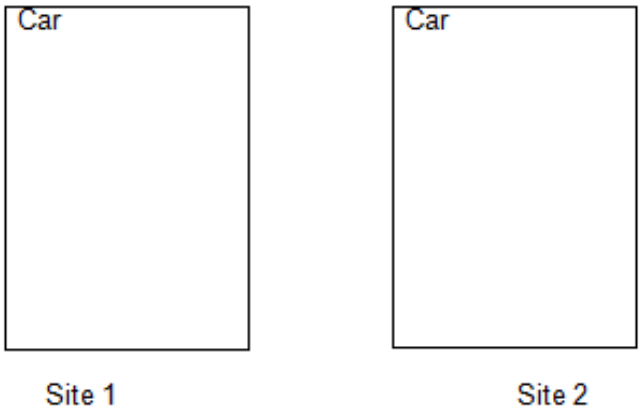

Figure 1 – Before Edits

Suppose that two people at Site 1 and Site 2 are working in a collaborative setting with a text editor that currently contains the word “Car”. The user at Site 1 inserts the character ‘s’ as the first character at the same time the user at site 2 deletes the letter ‘r’. In operational transform this would yield the following two transformations sent to the server copy.

1. Insert[0,’s’] - Insert character ‘s’ at position 0
2. Delete[2] - Delete character at position 2

Figure 2 – After edit with no transformation

Suppose the server receives the 1st operation and then the 2nd operation. Inserting the 's' into the 0 position and deleting the character at position 2 will lead to the string sCr. This is obviously wrong because the second user wished to delete the letter ‘r’. The idea of operational transform is that each operation be transformed with respect to previous operations so that concurrency can be achieved. In other words after the server parses the insertion of ‘s’ at position 0 it will increment the position of each operation according to their position.

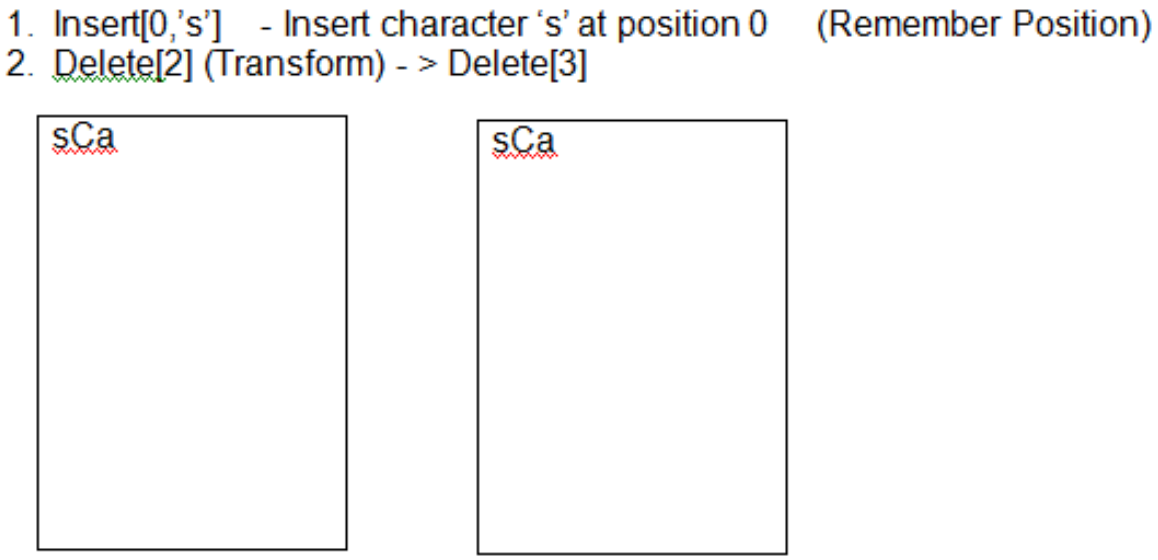

Figure 3 – After operation transformed

Getting the server to correctly transform the operations is however more complicated than incrementing indices. In order to maintain consistency among the documents at all the sites collaborating on a single document there are a number of different models with underlying properties that must be followed.

An Example of the base consistency model is the causality convergence model.

Causality Convergence Model

- Causality Property : Makes sure all edits which are causally dependent produce the same effects as was intended in the collaboration process[4,475].
- Convergence Property : Ensures all copies of a single collaboration document are equal (Operations have been applied to all sites) [4,475]

To implement operational transform it is important to know that it requires a system of components. The two main parts of an operational transformation system are the transformation control algorithm and the basic transformation operations.

1. Transformation Control Algorithm
    a. Generally determines the order of transformations
2. Transformation Operations
    a. Insert,Delete,Enter

There are generally 2 different types of operational transformation control functions.

1. Inclusion Transformation: IT(O1, O2) or $T(O_1,O_2)$, transforms O1 with respect to the effect of O2. [15]
2. Exclusion Transformation: ET (O1, O2) or $T^{-1}(op_1,op_2)$, transforms O1 without respect to the effect of O2. [15]

This means each operation that is received by the server is processed in pairs in a sorted order according to when they were received. Concurrent operations must be analyzed and possibly transformed with the inclusion function. To determine if two operations need to be transformed one must decide if they conflict. According to Du Li and Rui Li. in their paper on operational transformation [4],

two concurrent operations are conflicting if they possess one of following characteristics

(1) It is a deletion operation operating on the same character.

(2) One of the operations is deleting the character and the other is updating it.

(3) Both operations update the same attribute of the same character [4,477].

There are various implementations of the inclusion transformation and it generally varies based on the type of file you wish to collaboratively edit.

OT transform is by far the most popular method in solving the real time collaborative editing problem. The following is a list of software that currently uses operational transform.

Collaborative plain text editors

- Subethaedit (commercial)
- Ace (free, open-source)
- Gobby (free, open-source)
- MoonEdit(free for non-commercial use)

Web-based applications

- Google Docs & Google Wave.
- EtherPad (Purchased by Google)

3.1.2 Differential Synchronization

Another interesting method of implementing real time collaborative editing is with differential synchronization. Differential synchronization is a symmetrical algorithm created by Neil Fraser at Google as a part of the Google Mob Write project.

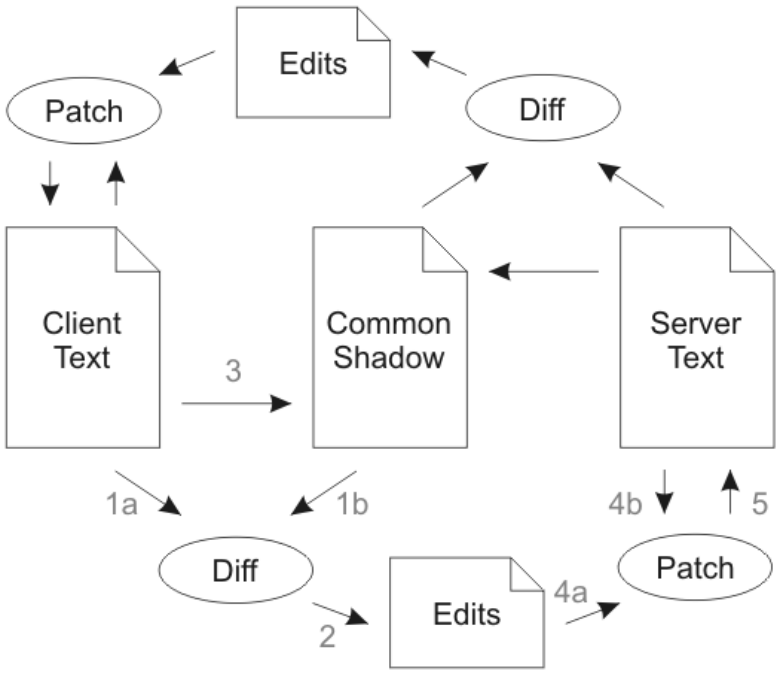

Figure 4 – Overview of Differential Synchronization [5]

Neil Fraser gives the visualization in Figure 4 and steps to go along with how exactly differential synchronization works.

1. Client Text is diff'd (get difference) against the Common Shadow.

2. This returns a list of edits which have been performed on Client Text.

3. Client Text is copied over to Common Shadow. This copy must be identical to the value of Client Text in step 1, so in a multi-threaded environment a snapshot of the text should have been taken.

4. The edits are applied to Server Text on a best-effort basis.

5. Server Text is updated with the result of the patch. Steps 4 and 5 must be atomic, but they do not have to be blocking; they may be repeated until Server Text stays still long enough.[5]

Differential Synchronization is a very interesting topic and alternative to OT. The algorithm does not have as much conceptual theory behind it as operational transformation as it is fairly new method composed of a few already established parts. Neil Fraser, the protocols designer, gives a wonderful Google Tech Talks lecture in the following video. http://www.youtube.com/watch?v=S2Hp_1jqpY8

3.3 Cursor Preservation

As multiple people type text into the same text box, cursor behavior becomes important to control. This may not be obvious at first but consider you are typing at cursor position 20 on an editor and someone comes along and inserts five characters starting at cursor position zero. The position you were previously editing will have moved up position 25 but your cursor would have stayed at 20 and been left behind. This is necessary to be fixed before a collaborative editor is usable. There are two correct ways to sufficiently implement cursor preservation.

3.3.1 Absolute Referencing

Absolute Referencing, based on storing character and cursor offsets, is the most popular technique for cursor preservation [3]. The start and end characters of each users' cursors are stored. If an insertion is received that starts at or before one of these points then the offsets are incremented by the length of the edit. If a deletion is received that starts before or at one of these points then the offsets are incremented by the

length of the deletion. Any deletions and insertions received after these points have no effect on the cursor. The following is a simple example whose form was taken from [3] showcasing absolute referencing.

The cursor is currently at offset 24, just before the word "filthy":

```
`Door sailiig, and the ^filthy doves
```

The following edits arrive from a remote user (strike-through represents a deletion, underline represents an insertion):

```
`Door sailiig, and& the filthy doves
```

Three characters were deleted and one was added. If there were no deltas made to the cursor offset, the cursor would shift by two characters:

```
`Door sailiig, & the fi^lthy doves
```

By subtracting three and adding one to the cursor location, the cursor is moved to the expected location:

```
`Door sailiig, & the ^filthy doves” [3]
```

3.3.2 Context Matching

Re-Positioning a cursor to its previous location after a remote edit by finding its previous context is known as context matching. In context matching a variable number of characters before and after the cursor start and end location are remembered. The location of the start and beginning points are also remembered. When a remote edit is received the text is patched and a fuzzy match algorithm is executed in order to find the location in the text that most closely matches the previously stored context. The algorithm by default will take into account both the difference in context and the offset between the old cursor and possible new locations and use this as a tie-breaker for equally probable context matches. The following example whose form was taken from [3] can accurately portray context matching.

Consider the the following text where ^ denotes the cursor position

```
`Door sailiig, and the filthy toves
  Did gyre and ^gimble in the wabe:       CONTEXT = S:yre and  E:gimble i
All simsy were the porogoves,
  And the moth waths outgrobe.
```

And the following edit is received.

```
`Door sailiig, and the filthy toves
All simsy were the porogoves,
  Did yre and& gGimble in the Wabe:
  And the moth waths outgrobe. [3]
```

Following the edit the algorithm will begin searching the new text for the context at around position 52. Eventually it will find the position of the new cursor at position 80 with a Levenshtein distance of 4 compared to the old context.

Context matching performs fairly well although there can be problems when there are duplicate lines. Google Docs uses context matching as a part of its real time collaborative editing.

4 Designs and Implementation

4.1 GUI Decisions

4.1.1. GUI Framework

One very large consideration I had when starting my project was which GUI framework to use. This is a very important choice for a number of reasons. First off the choice of GUI framework basically dictates the coding style. How complex will the code be? Do new languages need to be learned to adopt the framework? These are two

questions to consider when choosing a framework. Secondly, each GUI has different strengths and weaknesses. Lastly there are always factors to consider such as compatibility with various operating systems. I was able to conglomerate the following possible choices for the GUI.

Name	**Windows**	**OSX**	**LINUX**	**Familiar Language**
Silverlight	Yes	Yes with Mono-light	Yes with Mono-light	1/2
WPF	Yes	Not at this time	Not at this time	1/2
GTK+	Yes	Yes	Yes	Yes
Java Swing	Yes	Yes	Yes	Yes
WinForms	Yes	Yes with Mono	Yes with Mono	Yes

However this table is obviously inadequate in making a decision. I eliminated WinForms and Java Swing right away as their age renders them basically obsolete. GTK+ is the standard linux GUI toolkit. Its compatibility across all popular operating systems makes it very worthy of attention. However I did find its available tools and controls and complexity rather dated. I was also disappointed to learn that its most recent version GTK+ 3.0 was only available with C++,python, and various other functional programming language wrappers. At the end of the day I was looking at two main choices for my environment.

a. Silverlight

Microsoft Silverlight is a framework specified at creating applications that have heavy internet integration. Like WPF, Silverlight uses the Extensible Application Markup Language (XAML) programming language to describe frames and windows of the GUI. Although Silverlight applications can run outside a browser the vast majority of Silverlight applications run inside a browser. Silverlight is obviously the choice when building an internet application. Its cross compatibility with Linux and OSX when you use the ported version Mono-light is also a big plus.

b. WPF

Windows Presentation Foundation is a graphical subsystem for rendering user interfaces in Windows. WPF like Silverlight uses the XAML language to declare user interface objects and dynamically link these objects with items in the code. Each WPF file has an XAML file and a code behind file. WPF's most native language is C#. Using C# and XAML it is simple to create very dynamic, nice looking, and animated GUIs. The main downfall of WPF, although this may change, is that it is only compatible with windows at this time.

I ended up choosing WPF as the framework I would work in. This was mainly due to the fact that I felt more comfortable using it and felt Silverlight, although attractive with its cross compatibility, its browser driven platform might not be suitable for this project.

4.1.2. Docking Panels

The docking system of an IDE is very important as it gives users the freedom to decide what their user interface looks like and how they interact with it. To handle dock-

ing I once again looked to an open source control for docking called AvalonDocking. It behaves almost identically to Visual Studio 2010 which was in fact coded in WPF. Panels can be docked to the bottom, right, left, top and outside of the program itself as a stand-alone window.

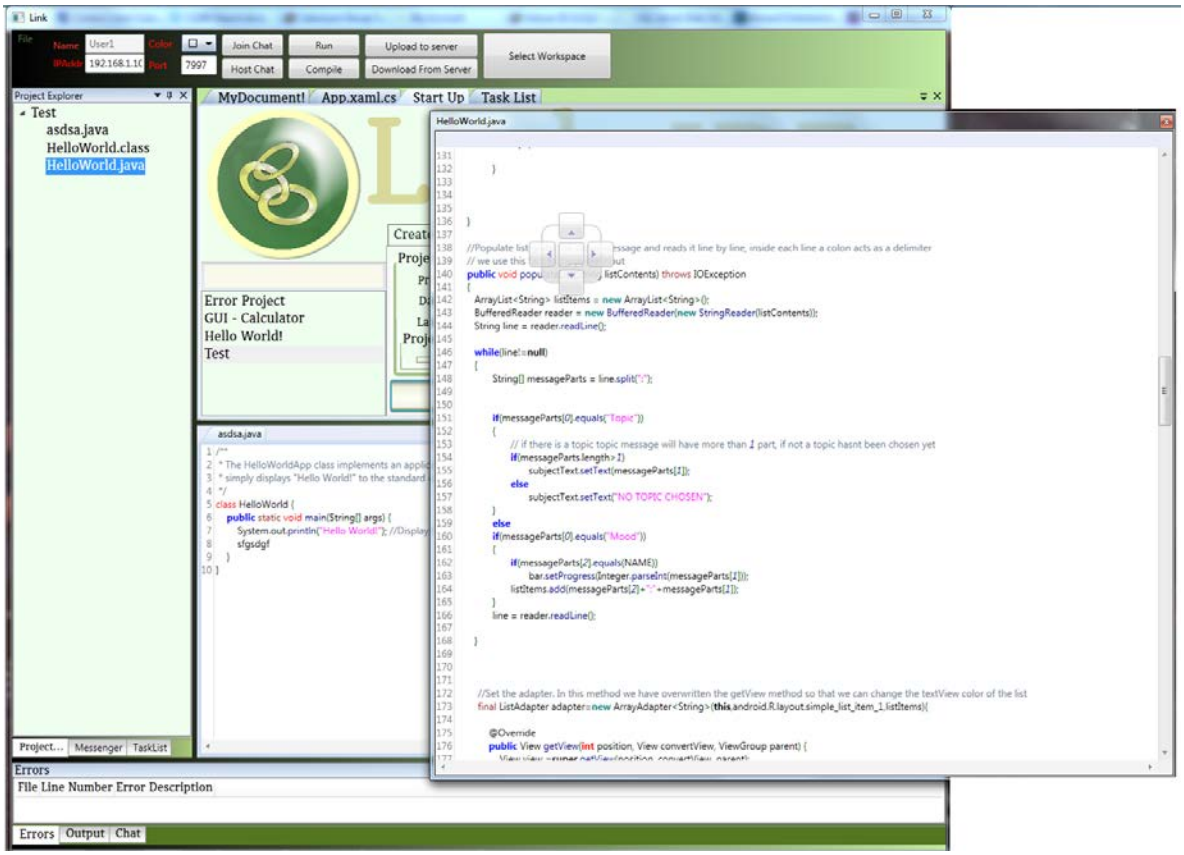

Figure 5 – Docking Capabilities

4.1.3. Code Editor

The text editor of an IDE is one of its most important members. A capable programming text editor is however quite complicated. It must handle syntax highlighting, line numbering, and in many cases code completion. In this project the text editor component has added complexity due to the aspiration for text editing to be collaborative

between many people in real time. This means the text editor is the essential component of this project.

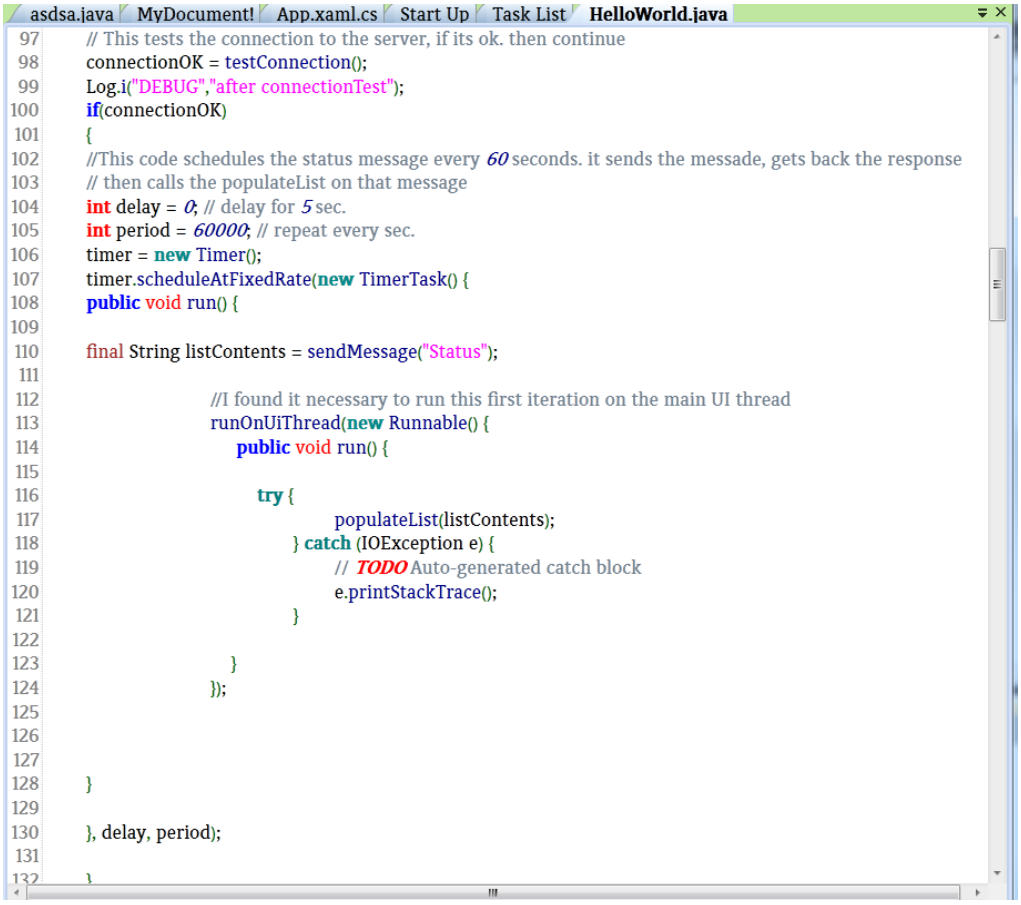

Figure 6 – Code editor

The process of creating a line numbered and syntax highlighting text editor can be time consuming. Instead of trying to mimic current implementation I decided to use a popular open source “code style” text editor called Avalon Edit. Avalon Edit supports syntax highlighting in multiple languages such as XML,Java,C++,and C#. It supports line numbering and has a built in mechanism for implementing a code completion. I felt this text editor was perfectly suited to my needs. I did however change much of source code in order to work with the networking aspect of the program

4.2 Persistence

To keep track of individual projects, users, tasks, and other user data my project takes advantage of many popular methods for persistence.

4.2.1. SQL Database

When a user first opens the Link IDE, he or she will see the project data screen. In this screen users can either create a project or resume an existing project. To resume an existing project they double click the project name as it appears in the left. At this time the user will be prompted to enter the project password. After they have entered the correct password they will reach a user login screen for that project where they must login as a specific user. Each project that is created is stored in a web-hosted SQL database in a table named Projects. Here the password is stored. Each project's info is loaded from SQL when the program starts up. The usernames for that project are also stored in a SQL table named Usernames with a foreign key pointing to the project they belong to. The usernames are loaded once the user enters the correct password for the project. Using a SQL server for this information is an efficient method of storing project and user information.

Figure 7 Sql Projects Usernames and create new project Page

Users can also create tasks for other users or themselves. These tasks designate work that needs to be done. In order to have constant access to the list of tasks each task and its information is stored in a SQL table ProjectTasks with a foreign key pointing to the project they belong to.

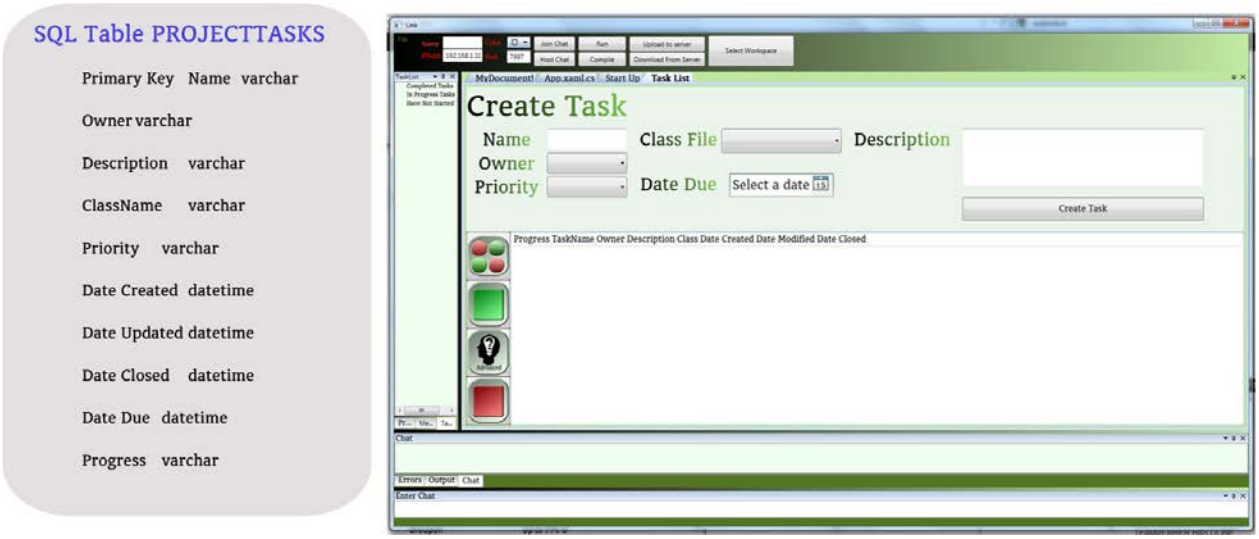

Figure 8 – SQL table and Create Task List

4.2.2. FTP Server

The storage of Link project files on the web is handled by a dedicated FTP server. When a user selects an existing project and enters the correct authentication then the files are downloaded from the folder on the FTP server with the name of the project. It will then create a folder for this project in the selected Link project workspace. On the creation of a new project a folder is created on the FTP server for that project name and a folder is created in the user's workspace. The class that handles the FTP operations runs the FTP requests in a background thread so as to not to disturb the UI.

Figure 9 – FTP relationship for projects

The presence of the FTP server is very useful because it allows users to have their code backed up without having to worry about it themselves. It also allows them to pick up where they left off no matter what computer they happen to working with.

4.2.3. Storing local data

Local data such as where the workspace for projects is stored are written to an isolated storage file. Isolated storage files are kept in a computer's hard drive memory and persist. This is useful because the user should have to select a new workspace upon using the program on a new computer.

4.3 Compilation and Building of Code

As of right now Link IDE supports only one language, java, and runs via the user's java compiler. When I started the project I realized that I should probably focus on one language and Java happened to be the one language I knew most about. A typical IDE does not usually come packaged with a compiler. It will expect users to have the required compilers installed on their systems in order for them to develop inside Link. Not including a compiler reduces complexity and reduces installation size to keep the program quick and responsive.

With that being said, the compilation and running procedures are actually very simple. As files are added to the project, the program tries to recognize which files are for code and add them to a sort of list to be compiled. The user must then mark the main class by right clicking and selecting "Set As Main" in the context menu that pops up. Then upon pressing compile the code is compiled and outputted to the output pane at the bottom. Any errors will be found in the error tab at the bottom in the same pane as the output (Figure 10).

In order to compile code, the java compiler exe is started with the arguments being the source code of the current project. Similarly, to run the code the Java exe is ran with the main file as the argument. Both of these actions are done via Processes in C# (See Code Example 1).

```
System.Diagnostics.ProcessStartInfo compileStartInfo =
new System.Diagnostics.ProcessStartInfo("javac");

System.Diagnostics.ProcessStartInfo runStartInfo =
new System.Diagnostics.ProcessStartInfo("Java");
```

Code Example 1 – Starting Java and Javac via Processes in C#

Errors received at run time are simply printed to the output window. Errors at compile time however are more complex to deal with. The compile error output stream must be read and parsed to separate errors and store their specific information. After the errors have been parsed they are stored in a collection that is shown the error listview which has details such as Name, Line Number, and Error Description (Figure 10)

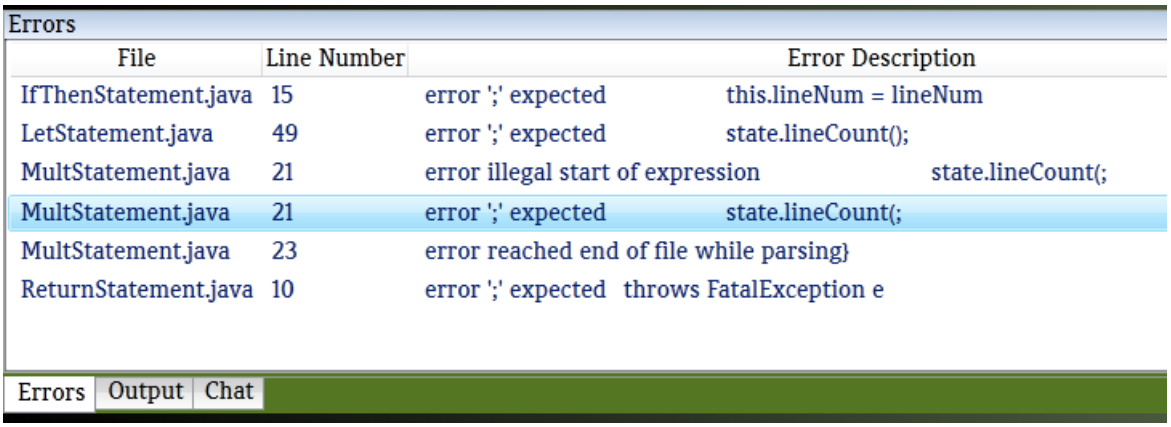

Figure 10 – Error list

Double Clicking on an error will automatically open or switch focus to the file containing the error.

4.4 Collaborative Implementations

4.4.1. Project/User Authentication

Requiring project authentication based on a project password and usernames offers obvious security benefits but also identifies each user and allows the system and users to analyze and assign work respectively. Because one is logged in when one makes changes, the system could potentially track user activity and efficiency. This might include graphs or bar charts showing the number of edits each user has done. This would be useful to tell who might not be doing enough work or who is

doing too much. Most likely a feature like this would influence people to work more to stay in balance with the others. More importantly assigning users each a personal name allows users to communicate more effectively whether it is by chatting or by delegating tasks.

4.4.2. Task List

The inclusion of a task list is a necessity to organize the otherwise chaos synchronous editing on the same document. Upon logging into a project users can start delegating tasks to other users by creating them in the create task menu. A task has a Name, Owner, Class File where task is to be implemented, Priority, Date due, and a Description. After a task is created, a signal is sent to the server and each user is triggered to refresh their task list. Tasks for the current user will then appear in the left pane along in the tab titled TaskList (Figure 11).

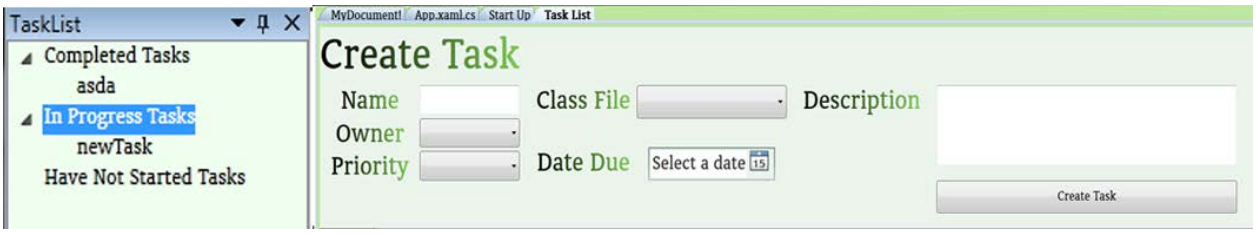

Figure 11 – Create Task

Tasks for the entire project are also visible via a task overview list. This will show each task's information. This operates in two different views, a list view and a tile view. The tasks are color coded depending on their progress. Finished tasks are green, in progress tasks are yellow, and not started tasks are red.

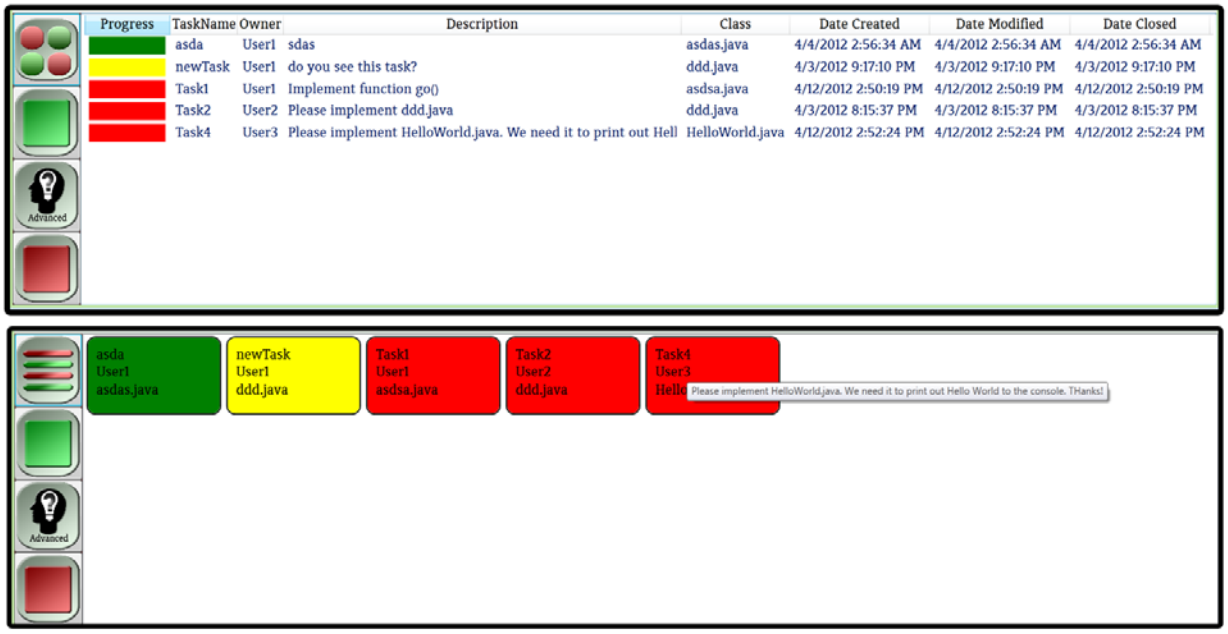

Figure 12 – Two task list view versions

Double clicking a task will open a window where that task can be modified and updated (Figure 13). Updates are simultaneously received by all users at the point of update. Left clicking a task will bring up a context menu where the task can be deleted (completed task and by owner only) and where the update menu can be found again (Figure 13).

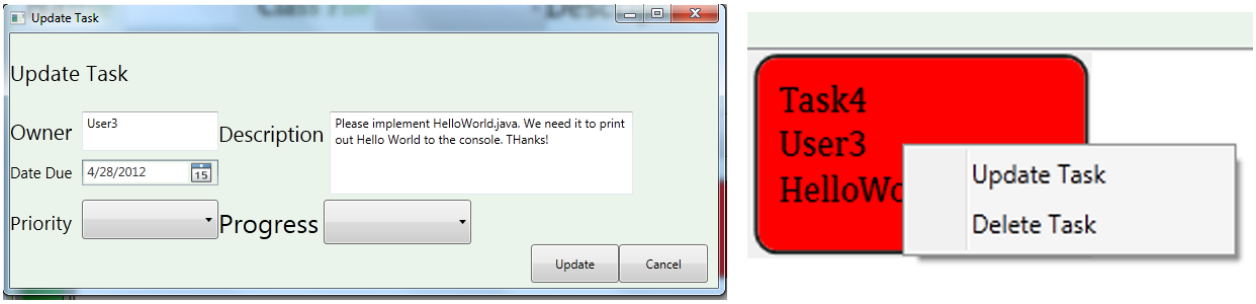

Figure 13 – Update Task

4.4.3. Chatting

Chatting through a global chat room and instant messaging is an important tool of communication between users that increases collaboration. In link there are two window panes which are vertically bounded at the bottom. One of these is to enter a chat message and the other is to view all the chat messages of the users (Figure 14). Docked to the left side of the window along with the task list and project explorer tree is a Messenger tab. This tab contains all the names for the users currently connected to the project. Double clicking a username will open an instant message window where one can hold a private conversation with only that user. Having this feature allows users to coordinate work on tasks that they may share without typing it directly into the real-time editor.

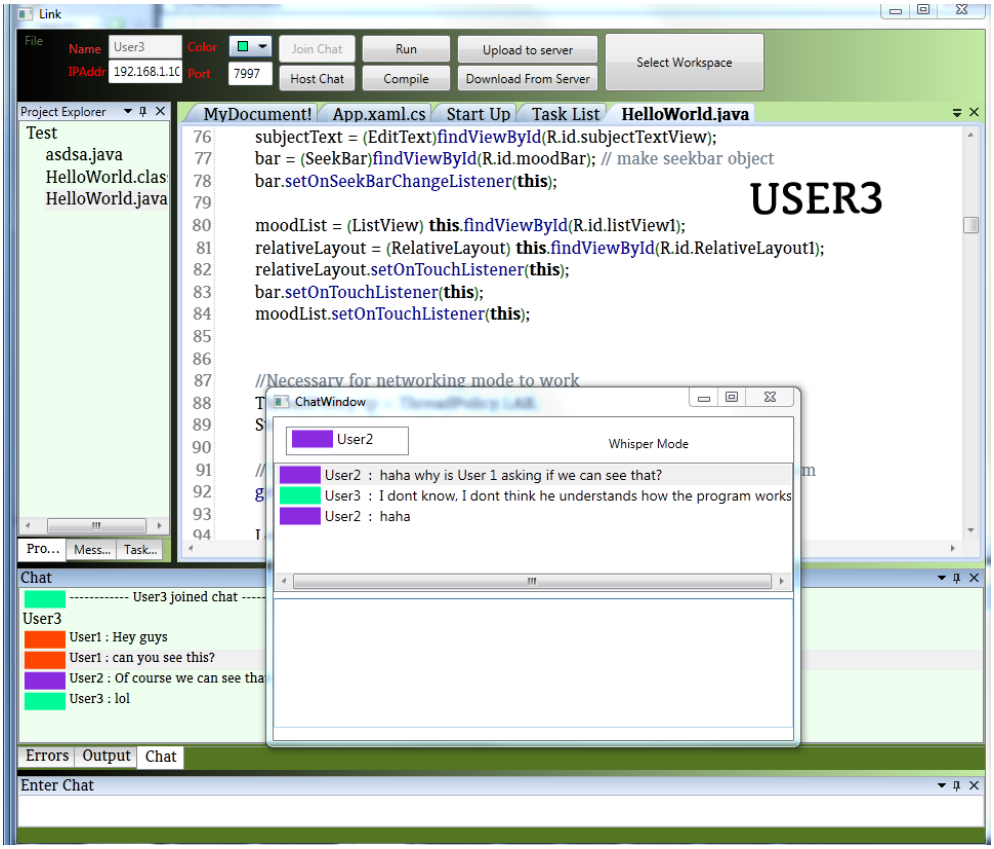

Figure 14 – Chat Functionality

Chatting with users operates through the same server as the real time collaborative editing. The server used for this was a Windows Communication Foundation (WCF) based server. WCF is a service oriented API for .Net that has excellent compatibility with WPF applications. The chat service implemented a few different methods relating to chat functionality.

```
public void Say(Message msg)
{
    lock (syncObj)
    {
        foreach (IChatCallback callback in clients.Values)
        {
            callback.Receive(msg);
        }
    }
}

public void Whisper(Message msg, Client receiver)
{
    foreach (Client rec in clients.Keys)
    {
        if (rec.Name == receiver.Name)
        {
            IChatCallback callback = clients[rec];
            callback.ReceiveWhisper(msg, rec);


        }
    }
}
```

Code Example 2 – Say and Whisper message server side.

These two methods implemented by the chat server are called by the user when they want to send a message to the chat box. When a say message is sent to the server it is then broadcasted to the entire list of users connected. When a whisper message is sent to the server it is sent only to the recipient. For the server to broadcast these messages

it needs to have the callback object call its Receive and ReceiveWhisper messages. The callback object is each client that has subscribed to the service and is implementing the service's interface. For chat functionality this interface requires that the user implement the following functions (Code Example 3) [17].

```
[OperationContract(IsOneWay = true)]
void RefreshClients(List<Client> clients);

[OperationContract(IsOneWay = true)]
void Receive(Message msg);

[OperationContract(IsOneWay = true)]
void ReceiveTaskChangeSignal();

[OperationContract(IsOneWay = true)]
void ReceiveWhisper(Message msg, Client receiver);
```

Code Example 3 – Call back methods on server side to be implemented by client

On the client side these implementations are fairly trivial and involve simple adding the messages received to UI components such as list boxes.

4.4.4. Real Time Collaborative Editing

In order to make the code editor inside the Link IDE a real time collaborative editor I roughly followed and implemented the Google Mobwrite protocol. Implementing this required me to have the three main functionalities of the protocol present. These functionalities include a version of differential synchronization, a difference and patching algorithm, and a cursor preservation method.

4.4.4.1. Difference and Patching algorithm

Neil Fraser, the designer of the Mobwrite protocol, has open sourced much of his work and included in that are the difference and patching algorithms for Mobwrite. I used his C# implementations as a part of the WCF server that controls most of the real-time collaborative functions in my project. Inside the service there are two methods that needed to form a patch and apply the patch (Code Example 4). First a diff_match_patch() object is created. This object can then call the function patch_make(String old, String new) in order to find the difference between the two strings and create "patch" object. This patch object can then be applied with function patch_apply(Patch p, String toBePatched). The following example shows how this would work.

```
                diff_match_patch dmp = new diff_match_patch();
                 String toBePatched ="This is to be patched";
dmp.apply(dmp.patch_make(toBePatched,"This was to be patched"),toBePatched);
          Console.WriteLine(toBePatched) => "This was to be patched"
```

Code Example 4 – Creating a diff and patching it into text

Using this I created a function diffAndPath(String newText, String filename) inside my program which return an Array object containing the patched text and the edit length (Code Example 5).

```
 private Object[] diffAndPatch(String newText, String fileName)
{
   String oldText = textFile[fileName];
   Object[] results = dmp.patch_apply(dmp.patch_make(oldText, newText),    oldText);
   textFile[fileName] = (String)results[0];
   int editLength = ((String)results[0]).Length - oldText.Length;
   results[1] =  editLength;
   return results;
  }
```

Code Example 5 – Server side function for finding difference and patching it

The Edit Length is especially important for cursor preservation

4.4.4.2. Differential Synchronization

The implementation of differential synchronization in my project varies in a few ways from how it is implemented in the Google Mobwrite protocol. Neil Fraser initially designed the algorithm with dropped packets on the internet in mind and thus it does some extra work to ensure consistency among all parties text. For use in my project I simplified the protocol.

The differential synchronization process runs on the same WCF server as the project's chat functionality and new task signaling. The first step in the implementation was to put a listener on the code editor of the program. Opening a file will automatically establish that file in the server. Establishing a file means that the server will now track its contents and allow multiple users to edit those contents. Upon entering any text in the editor this method is called (Code Example 6).

```
//NOT EXACT SAME CODE AS IN PROGRAM, MODIFIED FOR SIMPLICITY FOR PAPER
  private void editorTextInput(object sender, EventArgs e)
  {
     if (this.localClient != null)
     {
      SendTextChanges(Editor.Text.ToString(), Editor.Tag.ToString(),
      this.localClient, Editor.CaretOffset);
     }
  }
```

Code Example 6 – Listener placed on text editor to send text changes to server

This method will send the text changes, along with the files name and the cursor position of the edit to the server. Upon receiving the message the following method will be executed on the server side (Code Example 7).

```
public void SendText(String text, String fileName,Client client,int
carretPositionOfEdit)
 {
    Object[] newTextAndLength = diffAndPatch(text, fileName);
    foreach (Client sender in clients.Keys)
    {
      if (sender.Name != client.Name)
      {
         IChatCallback callback = clients[sender];
       callback.ReceiveText((String)newTextAndLength[0], fileName,
      (int)newTextAndLength[1], carretPositionOfEdit -
      (int)newTextAndLength[1],carretPositionOfEdit,false);

       }
     }
}
```

Code Example 7 – Server side response to text changing

This method first forms the diff between the new and old text, which is stored on the server, and then goes on to apply the path (Code Example 4). It then goes on to broadcast the changes to each client except for the sender by sending the callback message ReceieveText. The ReceiveText is then executed on the client side.

```
public void ReceiveText(String text, String fileName, int editLength,
int carretPositionOfEditStart, int carretPositionOfEditEnd, bool tag)
{
   int newCarretOffset = 0;
   bool changeCarret = false;
   if (openedDocuments.ContainsKey(fileName))
   {
      if ((openedDocuments[fileName]).Equals(editor.SelectedItem) && !tag)
      {
     int differenceBetweenCarretPositions = carretPositionOfEditStart –
     openedDocuments[fileName].Content).CaretOffset;
     if (differenceBetweenCarretPositions < 0)
     {
```

```
        newCaretOffset = openedDocuments[fileName].Content).CaretOffset + edit-
        Length);
         changeCarret = true;
       }
     }

//Remove the Text editor listener before adding text and then re-apply the listener
 openedDocuments[fileName].Content.TextChanged -= editorTextInput;
 openedDocuments[fileName].Content.Text = text;
 openedDocuments[fileName].Content.TextChanged += editorTextInput;

//If carret should be changed then change it
if (changeCarret)
       (openedDocuments[fileName].Content).CaretOffset = newCaretOffset;

 }
}
```

Code Example 8 – Client method for receiving text and adjusting cursor

Receiving the text and changing the editor to represent the new text changes is the easy part in this method. The complicated handling in this method comes from trying to preserve the cursor.

4.4.4.3. Cursor Preservation

Cursor Preservation as explained earlier is the necessity to move the cursor algorithmically every time an edit is made to the text in a collaborative editor. Without controlling this behavior in the correct manner, cursor locations might jump and move and the editor becomes basically unusable. For the purposes of my project I implemented an Absolute Reference cursor preservation method that is similar to the method used in the retired Google Wave.

In Code Example # 6 you can see that when a text change signal is sent to the server the cursor position of the edit is also sent as well. Following that the server determines the length of the edit and relays this information along with the cursor position

to each receiver. It is important to know the position of the edit and the length because we want to determine if the edits full encompassing length and position intersect with any other cursors in the program. If they do intersect their cursors are moved up according to the length of the edit. This is an efficient and clean method because it works for deletion just the same as additions.

5. Usability Testing

In a further step to validate my work on the Link IDE I found it imperative to carry out a usability test with the input of a few real life users. I called on the help of 3 users to test the program. User 1 is a graduate student of biomedical engineering. User 2 is a middle aged high school computer science teacher. User 3 is a graduate student in computer science. I tested each User separately three different times using three different tests where I would be present each time so that we could work together in a pair programming model. After these tests I scheduled a test with all three users and myself present and created a sample project. This final step was recorded on video and is available as a link referenced in this paper [16].

5.1. Test one – Project Creation, Resume project, Build Project

Test one involved asking the user to first create a project and add a few files. Next the user would exit the program and then resume the project. Finally the user would be asked to write some sort of simple program (ie Hello World) and then build and run it.

5.1.1. User 1

User 1 opened the program and initially did not know how to begin. Before using the “Create New Project” page he tried to do a “File - > New Project” route but found out it did not exist. After a short while he noticed the new project page and began filling out the necessary information. He had no trouble creating the project and creating his username. He found adding new files or existing files easy. There was no problem resuming the project or writing code but building the project was at first confusing. User 1 needed help in order to realize you had to set the main class of your program before running it.

5.1.2. User 2

User 2 had no problems creating a project. Upon resuming he realized he must have mistyped his password and could not get back in. After making a new user and exiting the program, he was able to login to the new user. Writing the program was very swift but he also ran into problems when attempting to run the program as he had not set the main class yet.

5.1.3. User 3

User 3 had little problems with the entire test. Because he spent time right clicking the project files he saw the context menu option to set a file as main and thus figured you must do that. He did however comment that he could see how that could be overlooked.

5.1.4. Analysis of Test 1 results

The main problem users encountered in the first test was that they did not realize that you had to set the main class that will be run. This is an obvious shortcoming of my project and it was something that I did not get to in the time I had. Because the program was still functional without it I left it as is. In a future version this will be fixed. User 2's interesting dilemma with his password being lost made me think I might want to add some sort of password recovery service.

5.2. Test two – Synchronous editing

In this test I asked the users to start a server and have me connect. Then they would disconnect and connect to a server I would start. Then we would both open the same document. The user would add a print statement with the variable "x" and I would initialize the variable and then we would run the program.

5.2.1. User 1

User 1 had problems figuring out his own IP when attempting to host the collaborative server and had to be helped. Connecting to my hosted server went without problem. The editing worked nicely.

5.2.2. User 2

User 2 also had some issues at first when hosting his own server but quickly understood. Connecting to my server was no problem. The editing was interesting at first because we were both editing in same area but quickly we adapted.

5.2.3. User 3

User 3 had no problem hosting or connecting to the server and found the synchronous editing to be responsive and impressive.

5.2.4. Analysis of test

This test was short and was mostly just to familiarize the user with the basic synchronous testing. Based on the confusion with IPs, it might be useful to add an automatic local IP recognizer.

5.3. Test three – Communication, Task Lists, Overview

The third experiment involved having users test the chat functionality as well as the task list. The test was also kind of an overview test because it incorporated most of all the functions of the application. Users would be asked to host a server for myself to connect to, complete the task I assign them and assign me a task to complete. During this test I would not be in the room with the user so that the chat functionality had to be utilized.

5.3.1. User 1

User 1 did not have a problem resuming the project from earlier and remembered how to host a server this time. User 1 had a bit of an issue seeing which tasks where assigned to him but after receiving instructions via the in applications chat window it was not a problem. Assigning a task was not an issue.

5.3.2. User 2

User 2 had no problems resuming the project and found the task screen right away and assigned myself a task. I assigned him a task and he was able to find it by exploring the applications window panes.

5.3.3. User 3

User 3 did not encounter any issues resuming his project. He was able to add tasks easily and realized quickly where the tasks showed up as they were added. User 3 had little problems with test 3.

5.3.4. Analysis of test

Test 3 was more of an introduction for the users to the task list and chat functionalities than a usability test. It was important to lay the groundwork and establish the understanding for the final step of usability testing.

5.4. Final Test

For the final step of usability testing I scheduled a time where all 3 users could be online at the same time so that a project could be worked on collaboratively together. The program assignment would be a simplified GUI calculator. The GUI would be composed up of 3 text fields and 4 buttons. There would be two text fields for the left and right numbers and one text field for the result. Each of the 4 buttons would compute the result of the addition, subtraction, multiplication, and division of the two numbers and display them in the result box.

After letting the users deliberate and decide how they were going to program I told them to record their work on the project. The video turned out to be an excellent way to see how efficient work could be done using this tool. This video is available in the link referenced in [16].

6. Future Work / Implementations

This section is dedicated to ideas I have had during my time working on the project that I did not have enough time to implement. If I ever make a real push to get Link IDE ready for a release then some of these will be a necessity.

6.1. Synchronous Locking

It may be useful to include some sort of mechanism for locking a specific document in the project. This would allow team members who do not wish to be disturbed by other users to work alone on a certain class file.

6.2. Concurrent Performance

It would be useful to parallelize all the code in the program in order to receive performance boosts. Network operations such as downloading project files from FTP could also be parallelized in order to increase the latency. Having separate servers for chatting and real time collaborative editing may increase robustness and speed as well.

6.3. Multiple Cursors Visible

One thing my program does not contain is the ability to see where each user's cursors are at and what their current selections are. This type of behavior would work similarly to Google Docs and would be beneficial to see possible conflicts in positioning edits.

6.4. Fully Web Hosted Servers

As of right now I have set up a single web hosted server in my own home to hold examples for outside of network. In the future I would want to deploy a web hosted version of the server so that users would not have to open ports to work with people on outside networks.

6.5. Fully Designed GUI.

Although the GUI is roughly designed at this point it is not complete. Interesting and aesthetically pleasing elements still need to be added. Certain UI elements can still be modified to have custom styles so they do not look like common WPF user interfaces. However this is something that would be done near the end of development when all the functionality is complete and mostly bug free

6. Conclusion

The Link IDE has been useful to showcase the possibilities of real time collaboration inside an IDE. Having organized user and project authentication along with web storage for projects allows for work to be continued at any location. Real time collaborative editing can be a new fresh air approach to source control that may suit some parties better. Combining this type of synchronous editing with a task list that users can easily access and follow gives project members a good idea of what needs to be done and allows work to move forward in an organized manner. There simply is no reason that there shouldn't be an IDE that allows users to work together in such a close environment. In the next few years it is inevitable that more and more software moves to real time collaborative formats including IDEs [7]. The most popular and actually used forms of existing implementations of collaborative IDES including Cloud9 and eXo are browser based. Although browser based applications are naturally more accessible than traditional desktop applications they are also limited in their processing power. Certainly graphics oriented, data intensive, or mobile applications could at this time not be developed in a browser. For this reason I think that traditional desktop collaborative IDE's like Link IDE will also become popular in the coming years. It is also necessary that future collaborative IDEs focus not only on being web accessible but also offering tools for organizing collaboration outside of the cloud such as task lists. The Cola plug-in for Eclipse to allow for synchronous editing is a mind blowing tool but it would be nice to see a product similar to Eclipses quality or even a version of Eclipse that supports this and other collaborative features out of the box. In the future I hope to work more on Link IDE and possibly with the help of other developers make it ready for a release.

7. References

[1] Binstock, A. (2012). Developers love IDEs. *InformationWeek,* (1322), 36-36. http://search.proquest.com/docview/922481313?accountid=10361

[2] Collaboration 2.0. (2009). *Library Technology Reports*, *45*(4), 16.

[3] Cursor Preservation, Neil Fraser
http://neil.fraser.name/writing/cursor/

[4] Li, Du, and Rui Li. "An Operational Transformation Algorithm And Performance Evaluation." *Computer Supported Cooperative Work: The Journal Of Collaborative Computing* 17.5/6 (2008): 469-508. *Academic Search Premier*.

[5] Sun, D.; Chengzheng Sun; , "Context-Based Operational Transformation in Distributed Collaborative Editing Systems," *Parallel and Distributed Systems, IEEE Transactions on* , vol.20, no.10, pp.1454-1470, Oct. 2009
doi: 10.1109/TPDS.2008.240
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4668339&isnumber=5226185

[6] Neil Fraser, Differential synchronization, Proceedings of the 9th ACM symposium on Document engineering, September 16-18, 2009, Munich, Germany

[7] Riemer, Kai, and Frank Frößler. "Introducing Real-Time Collaboration Systems: Development Of A Conceptual Scheme And Research Directions." *Communications Of AIS* 2007.20 (2007): 204-225. *Business Source Complete*.

[8] Understanding and Applying Operational Transformation, Code Commit
http://www.codecommit.com/blog/java/understanding-and-applying-operational-transformation

[9] Avalon Edit
http://wiki.sharpdevelop.net/AvalonEdit.ashx

[10] Avalon Dock
http://avalondock.codeplex.com/

[11] eXo Cloud Ide Website

http://www.cloud-ide.com/about.html

[12] Cloud9 IDE Website

http://c9.io/

[13] Eclipse Communication Framework Project Cola Project

http://live.eclipse.org/node/543

[14] Silver, Mark. "Browser-Based Applications: Popular But Flawed?." *Information Systems & E-Business Management* 4.4 (2006): 361-393. *Business Source Complete*. Web. 15 Apr. 2012.

[15] C. Sun, X. Jia, Y. Zhang, and Y. Yang, "Reversible inclusion and exclusion transformation for string-wise operations in cooperative editing systems," In Proc. of The 21st Australasian Computer Science Conference, pp. 441-452, Springer-Verlag, Perth, Feb. 1998.

[16]Usability Test Video
http://www.youtube.com/watch?v=pHKuf6Rt5MQ&feature=youtu.be

[17] A WCF-WPF Chat Application By Islam ElDemery | 15 Apr 2008 | Article
http://www.codeproject.com/Articles/25261/A-WCF-WPF-Chat-Application#xx0xx

8. Figures and Code Example Page Reference

8.1. Figures / Images

8.2. Code Examples