**Chapter 2**

**Scientific Background**

**2.1 History & Problems**

**2.1.1 Portability**

**2.1.2 Working anywhere**

**2.1.3 Organizing Versions of a large Project**

**2.1.4 Working Simultaneously**

**2.2 Previous Technologies**

**2.2.1 XMPP Standard Foundation**

**2.2.2 Ignite Real Time**

**2.2.3 Eclipse ECF**

**2.2.4 Google Wave**

**2.2.5 Etherpad**

**2.3 RTC Applications**

**2.3.1 CodeRun**

**2.3.2 Ideone**

**2.3.3 CodePlex**

**2.3.4 ECCO**

**2.3.5 CollabEdit**

**2.3.6 ShiftEdit**

**2.3.7 Gobby**

**2.3.8 Sync.in**

**2.4 Algorithms**

**2.3.1 Real Time Collaboration**

**2.1 History & Problems**

In this chapter we will focus on the history and problems in this field and a solution for each.

**2.1.1 Portability**

Can access files and work anywhere.

*Solution*: Online Storages.

**2.1.2 Work Anywhere**

Work without being chained to specific platform that has your programs and tools installed, just with internet connection can work anywhere.

*Solution*: appeared simple solution for some languages, some in the IDE’s for these languages and other as a web service, some websites provide the ability to code a lot of languages with a syntax highlighting and coloring for reserved words, and then for some languages it can the code in the server side and send back a runnable file like (exe) in windows.

**2.1.3 Organizing Versions of a large Project**

While working in a multi-developer project, it needs frequent updates form and to all developers with the latest updates, logging all changes.

*Solution*: Using SVN (Software Version Number) Servers.

**2.1.4 Working Simultaneously**

This is the feature is a main target in our project, easing the work for developers.

*Solutions*: The RTC (Real Time Collaboration), which give the ability for multi-developers to change simultaneously in a project even in the same page, which can be used in developing or learning as well.

Each problem had a solution for it; our project is to gather all solution in one easy to use service.

**2.2 Previous Technologies**

**2.2.1 XMPP Standard Foundation**

The Extensible Messaging and Presence Protocol (XMPP) is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. The [technology pages](http://xmpp.org/about-xmpp/technology-overview/) provide more information about the various XMPP “building blocks”.

The core technology behind XMPP was invented by [Jeremie Miller](http://xmpp.org/about-xmpp/xsf/xsf-people/" \l "bdfl) in 1998, refined in the Jabber open-source community in 1999 and 2000, and formalized by the [IETF](http://www.ietf.org/) in 2002 and 2003, resulting in publication of the [XMPP RFCs](http://xmpp.org/rfcs/) in 2004 (see the [history page](http://xmpp.org/about-xmpp/history/) for more details).

Although the core technology is stable, the XMPP community continues to define various [XMPP extensions](http://xmpp.org/protocols/xmpp-extensions/) through an open standards process run by the XMPP Standards Foundation. There is also an active community of open-source and commercial developers, who produce a wide variety of XMPP-based software.

**2.2.2 Ignite Real Time**

Open source software, open, collaborative minds are developing so that they would improve the software landscape.

The perspective is that base functionality – the core messaging aspects of RTC should be free and open source.

The community: Igniterealtime is a professionally-led (from [Jive Software](http://www.jivesoftware.com/?source=Website-Ignite)) Open Source community composed of end-users and developers around the world who are interested in applying innovative, open-standards-based Real Time Collaboration to their businesses. We're aimed at disrupting proprietary, non-open standards-based systems and invite you to participate in what's already one of the biggest and most active Open Source communities anywhere.

**2.1.3 Eclipse ECF (Eclipse communication Framework)**

[**ECF**](http://www.eclipse.org/ecf) is a framework for building distributed servers, applications, and tools. It provides a modular implementation of the **[OSGi 4.2 Remote Services](http://wiki.eclipse.org/ECF" \l "OSGi_4.2_Remote_Services)** standard, along with support for [**REST-based**](http://wiki.eclipse.org/REST_abstraction_for_ECF) and [**SOAP-based**](http://wiki.eclipse.org/SOAP-based_Providers) remote services, and [**asynchronous messaging for remote services**](http://wiki.eclipse.org/Asynchronous_Proxies_for_Remote_Services).

ECF/DocShare Plugin:

ECF version 2.0.0 and up includes a plugin called DocShare (**org.eclipse.ecf.docshare**) that implements real-time shared editing.

Real-Time Shared Editing over XMPP:

The [ECF](http://www.eclipse.org/ecf) project has been adding some real-time collaboration functions for dev team support...for example, [sharing selected text in Eclipse editors](http://wiki.eclipse.org/Sharing_Editor_Selections), as well as [real-time collaborative editing](http://wiki.eclipse.org/DocShare_Plugin)...all over the [ECF datashare API](http://wiki.eclipse.org/ECF_API_Docs#Datashare_API)...which means you can use your favorite public IM accounts like [google talk](http://talk.google.com/), or [skype](http://www.skype.com/)rather than be forced to have yet another account. We've also got some new [sharing of Mylyn tasks](http://wiki.eclipse.org/Integration_with_Mylyn) as per [bug 195737](https://bugs.eclipse.org/bugs/show_bug.cgi?id=195737).

RT Shared Editing:

Motivation for the RT Shared Editing, which is dubbed “**Cola**” (**Col**l**a**bode) for now, is supposed to be a tool enabling developers to reap the benefits of pair programming within the Eclipse IDE.

The term *pair programming* describes an activity in which two developers simultaneously work on a single development machine.

**2.2.4 Google Wave**

[Google Wave](http://wave.google.com/) is a real-time collaboration system based upon operational transformation approach to replicated state synchronization. With the Cola System (DocShare), ECF has been using operational transformations, for some time now. I will implement a provider to allow an equinox+ecf based web server to inter-operate with Google Wave.

With an announcement on 08/04/2010 Google [discontinued](http://googleblog.blogspot.com/2010/08/update-on-google-wave.html) the development of Google Wave as a standalone product. Anyway, an open source implementation of the [Google Wave](http://code.google.com/p/wave-protocol/) server is available and the protocol is also [open](http://www.waveprotocol.org/). The Google Wave (aka Federation One) server comes with an implementation of the operational transformation algorithm, which allows you to build real time shared editing applications from the comfort of your eclipse RT / ECF environment using this provider.

Implementation of a Google Wave ECF provider that people can use to build their own wave applications on top of ECF. The provider will handle the basic wave-protocol operations like managing waves, contacts and documents. Also the provider will provide an API which allows users to add listeners to wave changes and implement real time shared editing applications.

**2.2.5 Etherpad**

Etherpad is a hosted web service that allows really real-time document collaboration for group of users.

All editing of the document is instantly visible on the screens of all participating users, enabling new and productive ways to collaborate on text documents.

Etherpad is useful for meeting notes, drafting session, education, team programming, and more.

EtherpadLiteDotNet:

An implementation of the Etherpad Lite API for .NET written in C#

It implements the Etherpad Lite API, [more infomation on the API can be found on the Etherpad Lite wiki](https://github.com/Pita/etherpad-lite/wiki/HTTP-API).

It matches the API in structure as closely as it can.

It parses the returned JSON into strongly typed objects to allow the use of intellisense and reduce the use of magic strings.

The library is written in C# and uses .Net Framework 4.

**2.3 RTC Applications**

**2.3.1 CodeRun**

A free, Cross-Platform browser-based IDE

CodeRun Studio is a cross-platform Integrated Development Environment (IDE), designed for the cloud. It enables you to easily develop, debug and deploy web applications using your browser.

CodeRun Studio can be used instead or alongside your existing desktop IDE. You can upload existing code in order to test it in the cloud or for sharing with your peers.

**2.3.2 IDEOne**

Ideone is something more than a pastebin; it's an online compiler and debugging tool which allows  
to compile and run code online in more than 40 programming languages.

[ideone.com](http://ideone.com/) is designed mostly for programmers (but, of course, common plain text can also be uploaded). You can use it to:

* share your code (that's obvious - it is a pastebin, isn't it? :)) in a neat way,
* **run your code** on server side **in more than 40 programming languages** (number still growing)
* and do it all **with your own input data!**
* ideone.com also provides free [Ideone API](http://ideone.com/api) which is availabe as a webservice. Its functionality allows you to build your own ideone-like service!
* for logged in users Ideone offers possibility to manage their codes, publish multiple submissions at one go, and more.

All codes can be accessed through convenient hash links. Source code pages provide information about the code and its execution: memory usage, execution time, language and compiler version, code itself, input uploaded by the user, output generated by the program and error messages from compilers and interpreters.   
  
It is in great measure your contribution that this service exists - it is you who share your ideas with us and report bugs and suggestions. We hope it is still going to be like this - as long as this site works! Thank you!

**2.3.3 CodePlex**

CodePlex is Microsoft's free open source project hosting site. You can create projects to share with the world, collaborate with others on their projects, and download open source software.

**2.3.4 ECCO**

is an online programming development environment with file manager, console interface for compiling and execution of code and an editor with syntax highlighting, tabs and support for many languages which is accessed using a browser.

Main goals

* E-learning + interactive tutorials  
  Students can easily learn new languages through examples or interactive tutorials;  
  Students can learn basic linux commands;  
  Teachers can track students progress;
* Instant development and code   
  Begin to code without spending time with configurations and environments;  
  Test your code instantly;
* Access your code from anywhere  
  As an online service, it can be accessed from anywhere;
* Must be easy to use

**2.3.5 CollabEdit**

Collabedit is an online code editor that lets people collaborate in real-time.

It works in your web browser so no installation is needed.

Try it now, no account necessary, just click [here](http://collabedit.com/new).

Features:

* text editor
* chat
* document history
* syntax highlighting for programming languages

Collaborative Coding:

Email and instant messaging do not work well for sharing code. They don't preserve whitespace, the fonts aren't monospaced, the spell check gets in the way, etc.

Paste bins are better, but they are too static if you're looking for collaboration. Also they don't have a good editor, they're only good for copy and paste.

Collabedit has what you need - a good editor, syntax highlighting, real-time collaboration, chat and versioning.

Teach Programming:

Teaching demands a fine balance between letting your students do, and showing them how.

Collabedit allows student and teacher to share a common workspace without the distractions that other tools impose.

You don't need the hassle of exchanging emails with bad formatting. Whiteboards do not work very good for code, neither does Power Point. The pain of saying "closing parenthesis" out loud is no longer necessary.

Now you can teach more effectively with collabedit.

**2.3.6 ShiftEdit**

ShiftEdit is an online IDE for developing PHP, Ruby, Python, Perl, HTML, CSS and JavaScript via FTP, SFTP and Dropbox

"The online IDE is one of the final frontiers of apps ported to the web. I would like to be able to develop from any computer or operating system and have the same experience without having to install software or create site definitions." - Adam Jimenez (founder)

**2.3.7 Gobby**

Gobby is a free collaborative editor supporting multiple documents in one session and a multi-user chat. It runs on Microsoft Windows, Mac OS X, Linux and other Unix-like platforms.

It uses GTK+ 2.6 as its windowing toolkit and thus integrates nicely into the GNOME desktop environment.

**2.3.8 Sync.in**

Sync.in is a web based word processor for people to collaborate in real-time.

When multiple people edit the same document simultaneously, any changes are instantly reflected on everyone's screen. The result is a new and productive way to collaborate with text documents, useful for meeting notes, brainstorming, project planning, training, and more.

And more (ex: CakePHP, CodeAcademy).

**2.4 Algorithms**

**2.4.1 Real Time Collaboration**

In real time collaboration we faced a problem in the way we will compile the code

We are now synchronizing the code between users simultaneously using one of these ways:

1. Every time slice (each 3 seconds for example)
2. Every number of character changes (every 10 chars changes – add or delete)
3. Every number of character changes relative to time (send each #chars changed reached the previously specified number, else send in time slice specified as well)

All previous ways are independent from what every user is writing, which will raise a problem in debugging the code, cause if user (A) is writing in the top of the page

1. With every block change:

If writing a new block wait for it to take the whole struc

**2.4.1 Collaborative Editing with semantics**

Collaborative editing of source code files presents the particular problem that program errors introduced by one collaborator may disrupt the progress of their colleagues. Source files must conform to a particular programming language syntax and semantics, but it is impossible to expect users to maintain semantic validity throughout the coding process: almost any partially written expression, statement, block, method, class, or program is likely to contain errors according to the parser or compiler. Structured editors [17] address this problem only at the significant cost of requiring convoluted steps to make certain simple changes [28], and most programmers do not use them.

To see how errors introduced by one user can make it more difficult for another user to continue, consider the following simple program:

**class** Hello {

**int** x;

**void** a() { x = **true**; }

}

This contains an error, since the integer field x cannot be assigned a boolean value. However, suppose another user begins defining a new method:

**class** Hello {

**void** b() {

**int** x;

**void** a() { x = **true**; }

}

At this point the compiler will report instead that “x cannot be resolved to a variable,” masking the true error because x now appears to be a local variable of method b. The particular set of failure cases and their likelihood and severity will depend on the compiler. Compiler attempts to prevent common problems such as unbalanced braces from generating opaque errors elsewhere, but it cannot always succeed. The problem is much worse in the case of running the program: although Compiler will allow the programmer to launch an error-containing program, any attempt to execute an errorcontaining line will result in failure. This has the potential to prevent collaborating programmers from writing and testing their code incrementally, and we have observed this problem during pilot studies with both Python and Java programmers.

Rather than constrain collaborating programmers to agree on and reach points at which the program is sufficiently errorfree for them to run it and test code in progress, we instead account for the semantic content automatically in our synchronizing behavior.

|  |
| --- |
|  |
| Figure 2: Alice and Bob are working concurrently to define methods a and b. (a) The union of their changes is displayed in the editor, shown here from Bob’s perspective: his unintegrated change is in yellow, with an outgoing arrow; Alice’s is in gray, with an incoming arrow. (b) Alice’s working copy and (c) Bob’s working copy contain only their own changes. (d) The disk does not reflect their workinprogress, since both of their changes still contain errors. |

Collabode addresses this issue by first giving each programmer a separate, persistent working copy of the program, and then maintaining two versions that integrate programmers’ changes from their working copies: a disk version and a union version. The union version is the text that users see and manipulate, and it contains all edits applied by all users, with highlighting and icons to indicate provenance. So if Alice has begun defining method a and Bob is writing method b, the union version contains both in-progress methods, and both methods appear in the user interface (Figure 2a). As long as their methods contain compilation errors, the working copies of Alice and Bob will each contain only their own method (2b-c) and the disk will contain neither (2d). Once their methods compile, their edits will be shared both with their collaborator’s working copy, and with the disk version, which corresponds to the content on disk. It is this disk version that is run when either programmer elects to run the program. This version is always free of compilation errors.