A Collaboration Based Community to Track Idea Diffusion Amongst Novice Programmers

Orion Taylor, Reilly Butler, Greg Edelston, Jazmin Gonzalez-Rivero, Derek Redfern, Brendan Ritter, and Ursula Wolz

Franklin W. Olin College of Engineering
Olin Way
Needham, MA 02492

Abstract. Computer science is a collaborative effort, as evidenced by the rise of distributed version control systems and the proliferation of code-sharing services such as SourceForge and Github. We believe modern computer science educators should introduce students to programming in an environment where collaboration is encouraged. Sharing work should be more natural to the next generation of programmers than working individually.
We are prototyping an integrated development environment for this paradigm in the form of a community of practice, such as that presented by Etienne Wenger [EW1]. Our IDE is tied to a remote database to provide continual cross-device synchronization of students' work and to remove the complexity of collaborating with remote students. This IDE provides students with tools for direct collaboration on projects on the server, for giving feedback to peers, and building on others work.

1 Community

Creating comfortable and natural interactions with other users is the core goal of our prototype. Students need to feel safe posting their work. We developed a system of privacy controls designed to create a safe sandbox where students could interact with peers they trust.

1.1 Groups

In order to organize the numerous users that a server will support, we separated the user base into groups. These groups could consist of courses of novice programmers, students who want to collaborate on a coding projectgtgt or any group of people who share a common coding goal.

Groups can be as small as one member or as large as the user base of a server. They can contain multiple projects, allowing development of many different programs within the same group. Furthermore, this allows the user base to be sorted based on interest, allowing a user who wants to learn more or help with a certain project able to find the necessary information easily.

While only the members of a group can edit code in projects affiliated with the group, anyone can view and comment on the groups projects, encouraging open interaction between groups. If users wish to work with another groups source code, they can create a copy of a project in their own group. The new fork cites its parent project in its metadata, but is completely independent, changes made in a child project cannot be synchronized with its parent and vice versa.

New projects default to public visibility but it is possible to block a project from public view. Possible reasons for hiding a project could include if the project has sensitive information or if the project is an exam for a coding course. These rare cases aside, projects should be open to the public, allowing anyone to view and comment on the development of code. This will help foster an open and responsive community.

1.2 Citations

Traditional collaboration schemes track code forking and sharing, but often neglect sharing code snippets—little segments of code, which can be anything from the core of an algorithm to an obscure call to a library function. We wish to encourage students to borrow snippets from others' code. This is vastly different from the paradigm today of learning programming through individual projects hidden from other students. Most work in today's computer science field involves using, implementing and changing other's work rather than creating code from scratch—students learning how to code should reflect this type of collaborative development.

To create an environment where students are comfortable with borrowing snippets and letting others borrow their snippets, we propose a mechanism to simplify attributing snippets to their original authors.

The IDE will implement this by overriding the copy and paste functions of the interface toolkit. The client will track the origin of the contents of the system clipboard. If the snippet does not originate from one of the files belonging to the project the user is editing, such as lines from other projects on a server or documentation on the internet, it will offer to add a citation to the metadata of the file. The citation will indicate the the group that authored a snippet and where it came from. If the snippet came from outside the application, it will offer to cite a URL where the snippet can be found.

2 Design Considerations

Our primary goal was to make the user experience as straightforward as possible. The IDE is designed to be cater to experience from email and word processors. Functionality unique to our project is to be implemented so that it is intuitive easily discoverable.

2.1 User Interface

In order to make the project easily accessible, we are implementing a graphical user interface for interacting with code that integrates the whole user experience, from editing code to exploring projects and groups to commenting to revision control. The user is able to explore projects and view links between them in the form of borrowed code or forks. This allows the user to find projects that they are interested in, become part of those projects, view code that other members have already created, and work on code themselves.

For users that are already part of a group, it is very simple to view one another's code as well as leave helpful comments, which can be targeted generally at files or towards specific lines. Furthermore, users can visualize a project's revision history and citation in the form of a node map showing how a project moved and grew as groups forked it and how it borrowed snippets from other projects.

The IDE implements a minimalistic code editor which allows beginners to dive into the language and to create programs without worrying about the complex set up required in some IDE's while still including essential options found in modern IDE's such as syntax highlighting, auto-indentation, and integration with the Python interpreter.

The user interface is written in Python using GTK+. While the Linux desktop is our current development target, we plan to port the client software all desktop operating systems, while maintaining simple installation and high quality desktop integration.

2.2 Revision Control

We are developing a linear revision control system to reduce the entry barrier to working with revision control. Current revision offer powerful functionality for large development teams and long term projects at the price of a steep learning curve. We chose to sacrifice the more advanced functionality in favor of consistent and immediately apparent behavior.

Revisions represent the state of the project at a point in time that has been locked in, so that it can be viewed statically in the future. Viewing previous revisions allows users to see how the code has changed over time. Comments included with those previous revisions indicate why changes were made and where the ideas for changes came from.

The revision system is structured such that when a user edits a file, it is automatically synchronized in a head revision. Even in a public only members of a project's group can see files in the head revision. Revisions are intended for saving work at a largely stable point. Once the project has reached an important point, users can commit these changes to create a revision.

Any member of a group is able to make a new revision as long as files are not being edited at the time. Although real time collaboration is not currently planned—only one person can edit a file at a time—separate files in a project can be edited at the same time and users can view changes to a file as it is being edited.

3 Tools for Instructors

One of the goals of our project is to give computer science educators better insight into how students learn and communicate ideas. Instructors can make effective use of the revision control tool in order to observe the developing thought process of their students. This will allow instructors to better teach to the specific competencies of their students—if, for example, they see that students are solving problems only by means of 'for' and 'while' loops while shying away from recursion, then they know to focus more on recursion in the future.

3.1 Visualization

Our project is built on the premise of opening up the flow of ideas. One user may write a snippet of code with the intention of using it on their own project, but the significance of that code grows once it has been shared to the world. Other users can pick it up and add to it, share it with their friends, and even fork it and make it their own. With this in mind, we are developing a graphical tool that allows users to follow chains of code forks and borrowed snippets—in other words, exploration of how ideas spread. The map of shared snippets will provide insight into learning styles and help teachers tailor lessons to individuals.

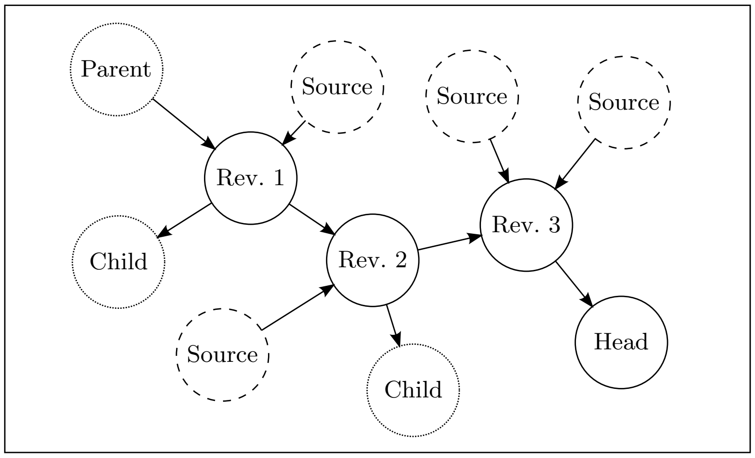

Fig. 1. This revision map shows how a project might grow over time, originating as a fork of a project, spawning child projects and borrowing code from others.

References

[EW1] Etienne Wenger introduced the term 'Community of Practice' to CS in 1998, even though the concept has existed for many years. See http://www.ewenger.com/ for his works.