Tree-based Version Control in Envision

Project Description and Schedule

Balz Guenat
guenatb@student.ethz.ch

January 15, 2015

Background

State of the art version control systems track changes to the granularity of lines in files. This only partially corresponds to how program code is structured semantically. Envision [1], an experimental IDE that is being developed at ETH Zurich, aims to aid the programmer by providing a visual representation of the program structure. Envision grants us the opportunity to explore how version control could work on the basis of a program's abstract syntax tree. Much initial work on this idea was done by Marin Otth during his master thesis [2]. The goal of this project is to complete and extend Otth's work by providing stronger correctness guarantees, designing good interfaces for the system's features and enabling users to customize the behavior of *diff* and *merge*.

Core Goals

The *diff* and *merge* algorithms are major components of a version control system and as such, should provide some guarantees of correctness and completeness. A first core goal is to define these guarantees in an exact manner to then systematically analyze the *diff* and *merge* algorithms. This analysis will guide the necessary revisions of the algorithms and underlying data structures in order to reach the established guarantees. After these proposed conceptual changes are deemed reasonable, they will be implemented.

Another core goal is to develop a suitable way to visualize the *diff* and *merge* functionalities for the user to enable an intuitive and effective interface. This includes *diff* visualization, manual merging and histories of individual AST nodes. Again, these interfaces will be implemented and integrated into Envision.

An evaluation of the revised algorithms will be performed to confirm that the desired guarantees are established and that the overall correctness of the system could be improved. Special attention will be given to cases that previously led to incorrect results.

Additionally, advantages of the developed system over existing line-based version control systems will be highlighted by comparing how common use cases are handled.

Possible Extensions

It would be interesting to provide a plugin interface to the *diff* and *merge* algorithms. This would allow users to add custom behavior in order to enhance *diff* information and/or to improve the outcome of a *merge*. In his work, Otth proposes some ideas on how renaming of entities could be handled by the version control system. This functionality could be designed as a plugin for the *merge*. If sufficient time remains after achieving the core goals, such an interface and client could be implemented and tested.

Schedule

This is a rough outline of the planned progress of the project. Especially the time allocated for extension work is flexible.

Deadline	Week	Task
Feb 1	0	Start of project
Feb 15	2	Familiarize with codebase and understand Otth's work
Feb 22	3	Define what guarantees *diff* and *merge* should provide
Mar 8	5	Identify weaknesses of current *diff*, *merge* and/or provide reasoning for correctness
Mar 22	7	Find conceptual solutions
Apr 19	11	Implement said solutions
Apr 26	12	Begin writing report about work up to this point
May 17	15	Come up with good interfaces for *diff*, *merge* and history
May 31	19	Implement said interfaces
Jun 7	20	Evaluation of work up to this point
Jun 21	22	Conceptual work for extensions
Jul 12	25	Implementation and evaluation of extensions
Aug 1	28	Finish report, end of project

References

[1] D. Asenov and P. Müller. Envision: A fast and flexible visual code editor with fluid interactions (overview). In *Visual Languages and Human-Centric Computing (VL/HCC)*, pages 9–12, 2014.

[2] Martin Otth. Fine-grained software version control based on a program's abstract syntax tree. Master thesis, ETH Zurich, 2014.