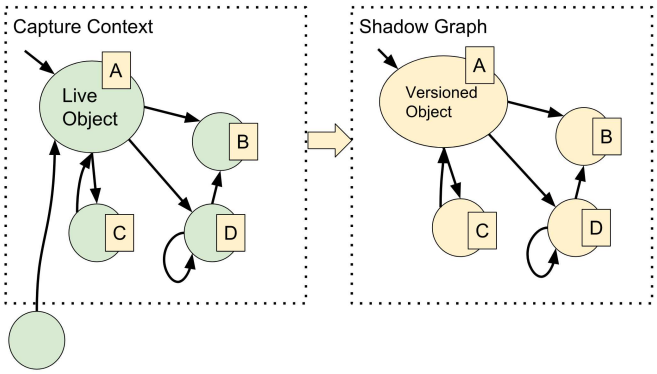
Merging in Object-centric Version Control Systems

Master Project Proposal, Winter Term 2017/18
Software Architecture Group, Prof. Dr. Robert Hirschfeld

In this project, we will design and implement means and tools to merge version controlled objects in interactive, object-based programming systems. In particular, we will look into tools to visualize and interactively resolve differences between object graphs. The main challenge will be to account for complex and domain-specific merge conflicts. Further, to allow programmers to work with services such as Travis CI, we will investigate ways to integrate service configurations into object-oriented abstractions within the programming system. All tools will be developed based on the object tracking framework Squot and its bridge to Git called Squit.

Object-centric Version Control



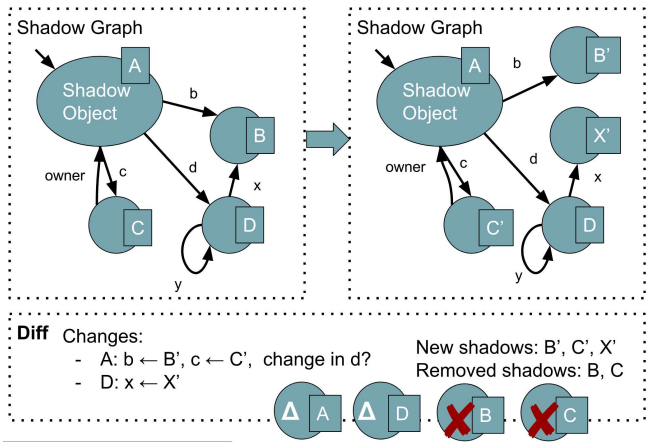
Most of present day version control systems, such as Mercurial or Git, are used to manage files containing text. As a result, the corresponding tools and infrastructure, such as the Git command line tool or the GitHub platform, are designed for working with files.

These tools and services do not work well with object-based programming systems in which parts of the application are defined through long-living objects, for example in multi-media projects or user interface themes in Squeak/Smalltalk. Similar issues arise in model-driven or projectional development environments that represent the application logic in non-linear graphs.

An *object-centric version control system* provides abstractions and means to put object graphs under version control. While programmers store and load object graphs in such a version system, in the end, this graph can be serialized to files, so that the graph can also be shared through services working with files, for example GitHub.

Merging Text versus Merging Objects

One prominent feature of version control systems is the support of parallel strands of development, for example through branches in Git. As a result, however, programmers regularly have to manually merge conflicting changes to the source code. Again, tool support for merging, through automated merge algorithms or specialized editor views, is limited to textual representations of source code stored in files.



Tool support for merging changes to object graphs would need a fundamentally different design. The main challenge for merging object graphs is the granularity of changes. While merging of text happens at the granularity of lines, merging of object graphs happens at the relatively finer granularity of single assignments and the creation of new objects. This results in a large number of changes per version and makes manual merges tedious.

Text Files and the Object Graph

Besides the merging functionality, objects as the unit of a version control system also impact the integration of a project with external services. Some online services for software development, such as TravisCI, are configured through textual configuration files in repositories of a version control system. This requirement conflicts with the interface of an object-centric version control system that manages object graphs and not single streams of text.

This conceptual difference results in a variety of issues for managing text files from within the object-based environment. In particular, configuration files have to be put at specific locations in a tree which is not easily possible in a graph (for example the TravisCI configuration has to be in the root folder). Further, the objects in a graph are mostly unnamed while textual configuration files often have to be stored under specific names, for example .travis.yml for configuration files.

Contact

Prof. Dr. Robert Hirschfeld, Patrick Rein, Tobias Pape, Fabio Niephaus, Jakob Reschke
Software Architecture Group, Potsdam
http://www.hpi.uni-potsdam.de/swa, hirschfeld@hpi.uni-potsdam.de