Concept - Undo and redo

Rationale

Being able to undo an action is a feature found in most user interfaces that gives the user an easy way to undo a mistake. It also gives the user a sense of safety, its not critical if you do a mistake or change your mind cause there's always a way back. This makes the user more confident and more inclined to explore the application.

Here's a quote from http://foruse.com/articles/instructive.pdf

> Facilities to undo and redo actions are at the core of explorability. These facilities must be ubiquitous and absolutely consistent. Users who encounter even a single isolated case in which the Edit | Undo option is mysteriously disabled will become more cautious and hesitant to try new things. A single level of undo is insufficient to fully support free exploration. In principle, infinite-level undo is needed, but in practice, a dozen or so levels is as good as infinite because users are seldom able to make effective use of more levels.

Design

Implementation undo functionality is usually done using the command pattern. Every action invoked by the user causes the execution of a command. Each executed action adds a command that reverses the changes made to the undo stack. To support redo functionality the command also adds itself to the stack.

Step

- A step has a representative label presented in the UI
- A step has a command that reverses the effect of an executed command
- A step has a command the redos the executed command

Stack

- The stack maintains a list of steps that can be undone and a separate list of steps that can be redone
- The stack has a maximum number of steps it can store

Command

- A command mutates the subject in some way
- An undoable command adds a step to the history
    - Any redo steps are then dropped from the stack
- A command that cannot be undone clears the stack completely

Collaboration

- Each user has a stack of his own
- If two users edit the same subject and their changes are not kept local to each user until they save then their undoing and redoing can interfere

UX remarks

- How many steps are necessary to keep?

        - some say 10 is enough and 12 more than enough
- The history can be useful also as a reminder of what you've recently done
- Should it be possible to choose a step further back causing all steps back to it being undone?
- Commands that cannot be undone might need to alert the user of this fact

Implementation remarks

- Undo commands need to keep information necessary to restore the subject to its previous state, this can be done in a number of ways:
    - By taking a complete copy of the subject before and after
    - By taking a copy of the part of the subject to be changed before and after
    - Using a granular operation specific to what was done
        - Such as change-property from x to y