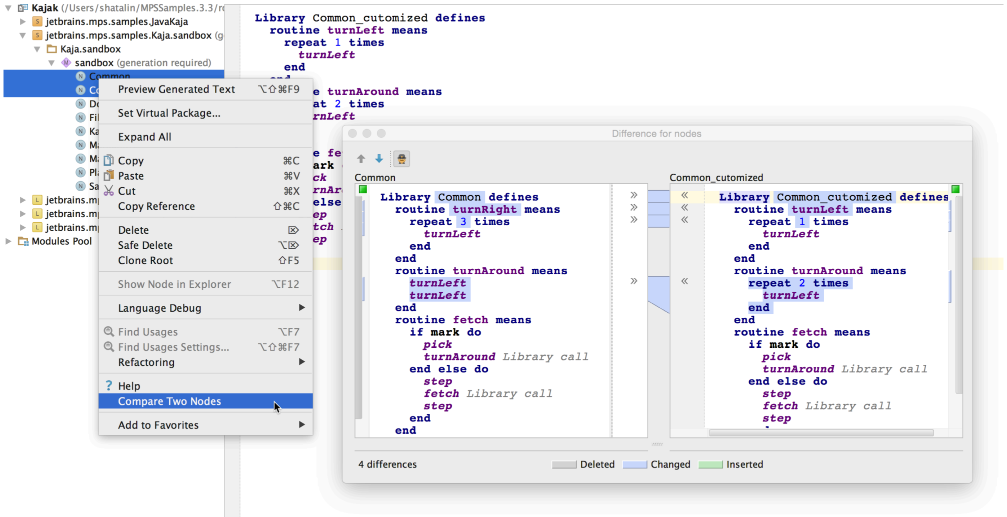
Version Control

Compare two nodes

Any two arbitrary nodes in the Project View tool window can be visually compared:



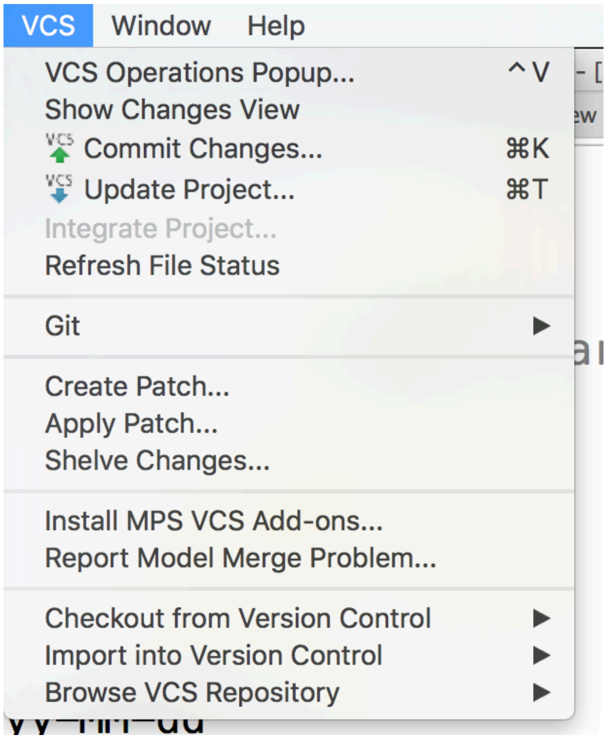
```
Library Common defines
  routine turnRight means
    repeat 3 times
      turnLeft
    end
  end
  routine turnAround means
    turnLeft
    turnLeft
  end
  routine fetch means
    if mark do
      pick
      turnAround Library call
    end else do
      step
      fetch Library call
      step
    end
  end
```

```
Library Common_cutomized defines
  routine turnLeft means
    repeat 1 times
      turnLeft
    end
  end
  routine turnAround means
    repeat 2 times
      turnLeft
    end
  end
  routine fetch means
    if mark do
      pick
      turnAround Library call
    end else do
      step
      fetch Library call
      step
```

The standard VCS comparison dialog shows up that visualizes the mutual differences and allows easy modifications.

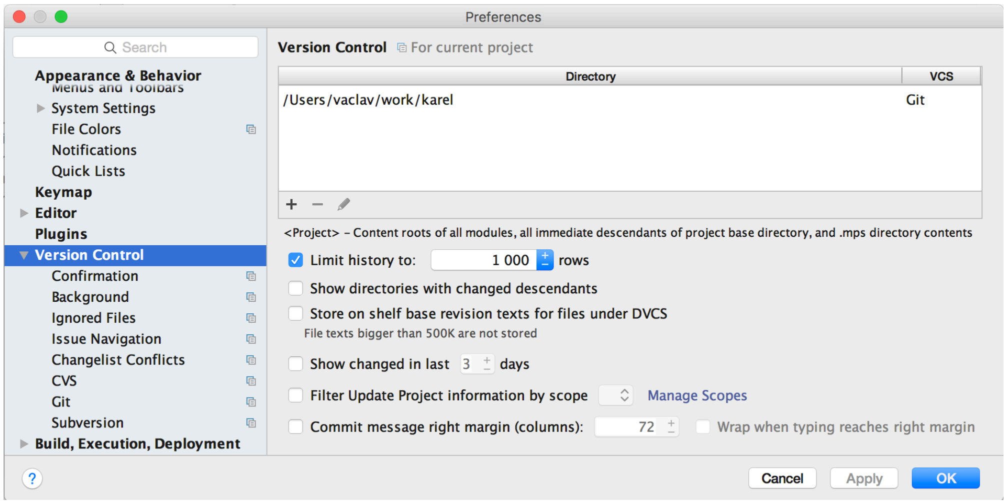
VCS menu

The VCS menu contains commands and configurations related to version control:



VCS configuration

In order to configure VCS for your project, open the Preferences (Control + Alt + S or Cmd + ,) and choose the Version Control item:



Version Control For current project

Directory	VCS
/Users/vaclav/work/karel	Git

<Project> – Content roots of all modules, all immediate descendants of project base directory, and .mps directory contents

☑ Limit history to: 1 000 rows

☐ Show directories with changed descendants

☐ Store on shelf base revision texts for files under DVCS
File texts bigger than 500K are not stored

☐ Show changed in last 3 days

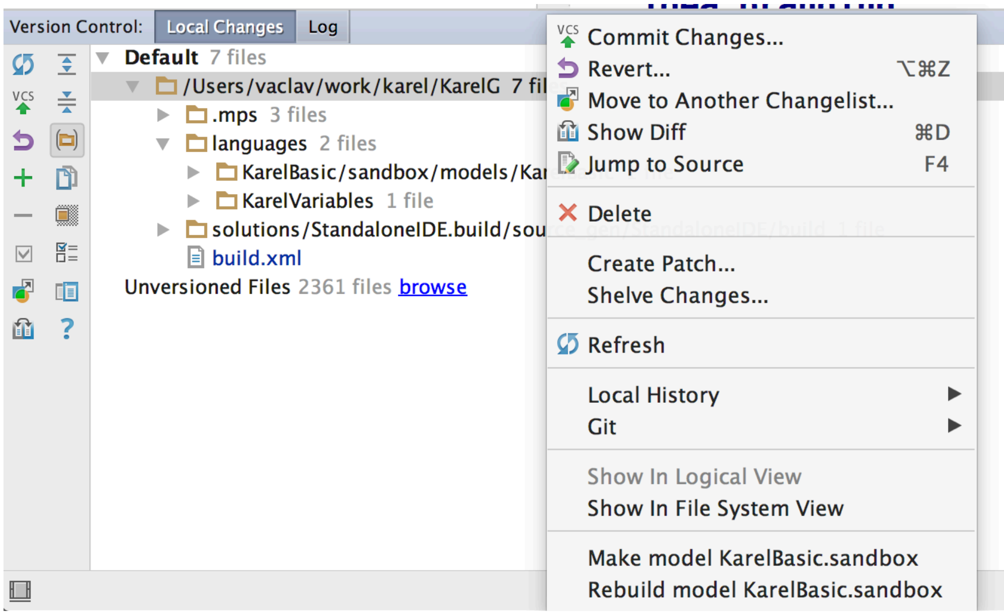
☐ Filter Update Project information by scope Manage Scopes

☐ Commit message right margin (columns): 72 ☐ Wrap when typing reaches right margin

Among other things you must configure the project roots for the individual version control systems used.

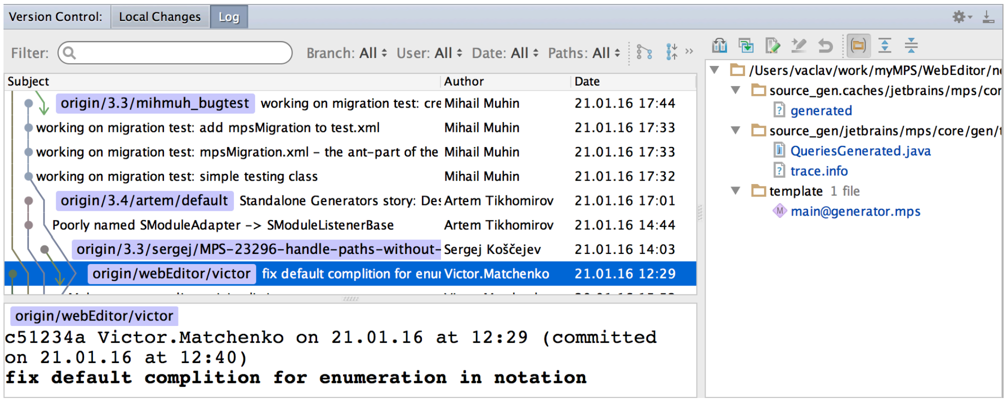
Changes View

The Changes View tool window at the bottom (Alt + 9 or Cmd + 9) lists all files/models that have been modified. The view can be configured using the buttons on the side of the window. A context pop-up menu provides quick access to frequently used VCS-related actions applicable to the selected items:



```
▼ Default 7 files
  ▼ /Users/vaclav/work/karel/KarelG 7 fil
    ▶ .mps 3 files
    ▼ languages 2 files
      ▶ KarelBasic/sandbox/models/Kar
      ▶ KarelVariables 1 file
    ▶ solutions/StandaloneIDE.build/sou
      build.xml
Unversioned Files 2361 files browse
```

The Log tab of the Changes View visualizes the commit history:



Subject	Author	Date
origin/3.3/mihmuh_bugtest working on migration test: cre	Mihail Muhin	21.01.16 17:44
working on migration test: add mpsMigration to test.xml	Mihail Muhin	21.01.16 17:33
working on migration test: mpsMigration.xml – the ant-part of the	Mihail Muhin	21.01.16 17:33
working on migration test: simple testing class	Mihail Muhin	21.01.16 17:32
origin/3.4/artem/default Standalone Generators story: De:	Artem Tikhomirov	21.01.16 17:01
Poorly named SModuleAdapter -> SModuleListenerBase	Artem Tikhomirov	21.01.16 14:44
origin/3.3/sergej/MPS-23296-handle-paths-without-	Sergej Koščejev	21.01.16 14:03
origin/webEditor/victor fix default complition for enu	Victor.Matchenko	21.01.16 12:29

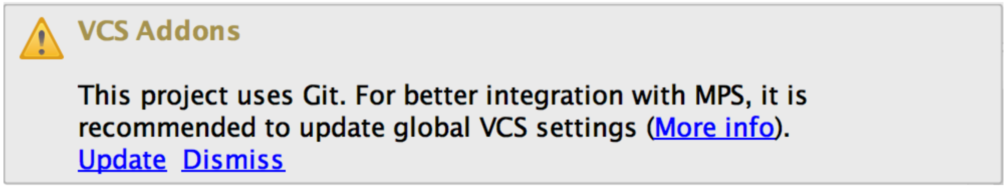
origin/webEditor/victor

```
c51234a Victor.Matchenko on 21.01.16 at 12:29 (committed
on 21.01.16 at 12:40)
fix default complition for enumeration in notation
```

```
/Users/vaclav/work/myMPS/WebEditor/n
  source_gen.caches/jetbrains/mps/cor
    generated
  source_gen/jetbrains/mps/core/gen/t
    QueriesGenerated.java
    trace.info
  template 1 file
    main@generator.mps
```

VCS Add-ons

When you first open MPS with version control or add VCS mapping for existing project, it offers you installing some global settings and install so called VCS Add-ons (they can also be installed from main menu: Version Control Install MPS VCS Add-ons).



VCS Addons

This project uses Git. For better integration with MPS, it is recommended to update global VCS settings (More info).

Update Dismiss

What are VCS Add-ons

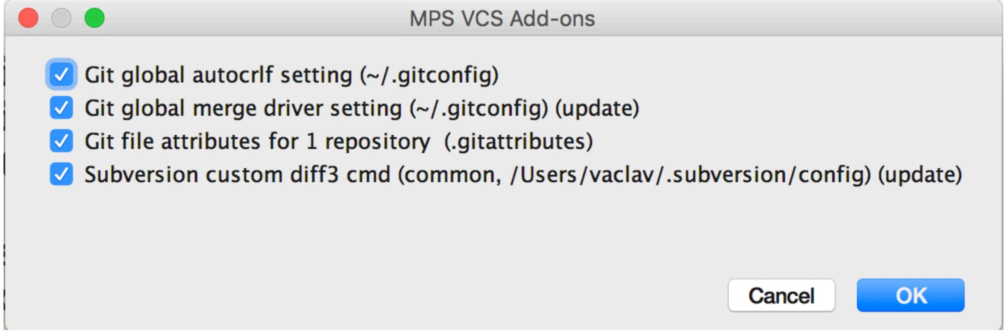
VCS Add-ons are special hooks, or merge drivers for Subversion and Git, which override merging mechanism for special types of files. In case of MPS, these addons determine merging for model files (*.mps) and generated model caches (dependencies, generatedand trace.info files, if you store them under version control). Every time you invoke some version control procedure (like merging branches or Git rebasing) which involves merging file modifications, these hooks are invoked. For models, it reads their XML content and tries to merge changes in high level, "model" terms (instead of merging lines of XML file which may lead to invalid XML markup). Sometimes models cannot be merged automatically. In that case, it stays in "conflicting" state, and it can be merged in UI of MPS.

In some cases during work of merge driver, there may happen id conflicts, situations when model has more than one node with the same id after applying all non-conflicting changes. In this situation, no automatic merging is performed, because it may lead to problems with references to nodes which are hard to find. In this case you should should look through merge result by himself and decide if it okay.

For model caches merge driver works in a different way (if you store them under version control, of course). Generator dependencies (generated files) and debugger trace caches (trace.info files) are just cleared after merging, so you will need to regenerate corresponding models. Java dependencies (dependencies files) which are used on compilation are merged using simple union algorithm which makes compilation possible after merging.

Different VCS Add-ons

Look at the dialog:



☑ Git global autocrlf setting (~/.gitconfig)
☑ Git global merge driver setting (~/.gitconfig) (update)
☑ Git file attributes for 1 repository (.gitattributes)
☑ Subversion custom diff3 cmd (common, /Users/vaclav/.subversion/config) (update)

There are several types of VCS Add-ons which can be installed. It is recommended to install them all.

- Git global autocrlf setting. Forces git to store text files in repository with standart Unix line endings (LF), while text files in working copy use local system-dependent line endings. Necessary when developers of your project use different operating systems with different line-endings (Windows and Unix).
- Git global merge driver setting. Registers merge driver for MPS models in global Git settings so they can be referred in .gitattributes files of Git repositories (see below). It only maps merge driver name (in this case, "mps") with path to actual merge driver command.
- Git file attributes for repositories. Enables MPS merge driver for concrete file types (*.mps, trace.info, etc) s in Git repositories used in opened MPS project. This creates or modifies .gitattributes file in root of Git repository. This file usually should be stored under version control so these settings will be shared among developers of the project.
- Subversion custom diff3 cmd. Registers MPS merger in config file of Subversion. MPS may use its own config folder for Subversion, so there are two different checkboxes. One updates global config used when you invoke Subversion procedures from command line or tools like TortoiseSVN. Another one modifies config only for MPS Subversion plugin. By the way, directory for Subversion config used in MPS can be defined in Subversion settings.