

---

# ScribesTools Documentation

*Release 0.6.1*

**escribis**

April 10, 2017



<b>1</b>	<b>Tools</b>	<b>3</b>
1.1	GanttProject . . . . .	3
1.2	Modelio . . . . .	19
1.3	UseOCL . . . . .	34
1.4	KMADe . . . . .	46
1.5	Pandoc . . . . .	48
1.6	SchemaSpy . . . . .	49
1.7	SchemaCrawler . . . . .	51
1.8	CheckStyle . . . . .	53
1.9	PyLint . . . . .	54
1.10	Diigo . . . . .	55
1.11	Assembla . . . . .	57
1.12	Git(Hub) . . . . .	58
1.13	ScribesGit . . . . .	65
1.14	ReadTheDocs . . . . .	71
1.15	Selenium . . . . .	73
1.16	TravisCI . . . . .	73
1.17	Java . . . . .	73
1.18	Python . . . . .	74
1.19	PyCharm . . . . .	77
1.20	Django . . . . .	80
1.21	SQLite . . . . .	81
1.22	MySQL . . . . .	82
1.23	Graphviz . . . . .	82
1.24	Sphinx . . . . .	83
<b>2</b>	<b>Platforms</b>	<b>93</b>
2.1	Windows . . . . .	93
2.2	Ubuntu . . . . .	93
<b>3</b>	<b>Misc</b>	<b>95</b>
3.1	Le jeu du promoteur . . . . .	95



Welcome to ScribesTools documentation. You will find here some indications on how to install and use some useful software development tools. If you are interested in quality aspect you may also be interested in [ScribesQuality](#).



## GanttProject

GanttProject is a free project management tool allowing to *edit gantt models* but also *view pert models* and *resource allocation models*. As this tools is written in java it runs on all platforms.

### Features

GanttProject allows to create *milestones* and hierarchies of *tasks* related with *dependency constraints*. Different *fields* can be attached to *milestones* and *tasks*. This includes for instance “priority”, “cost”, “start date”, “duration”, etc. *Custom fields* can also be added.

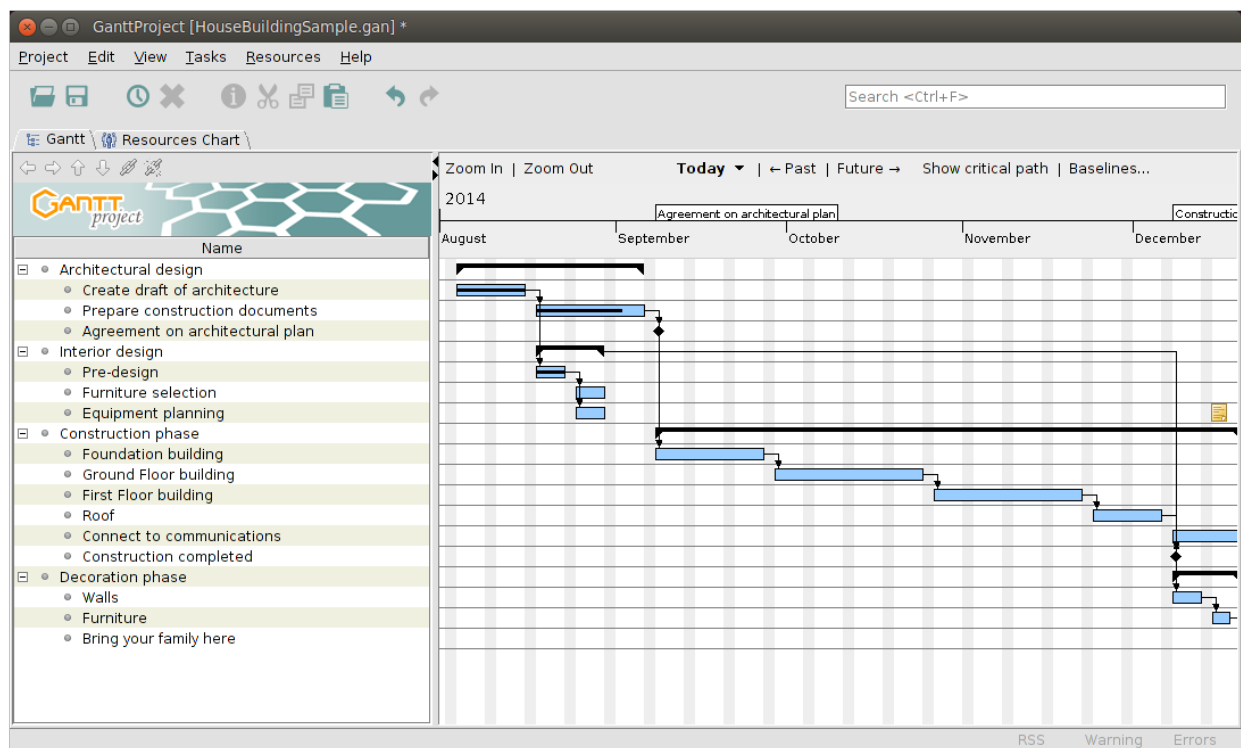


Fig. 1.1: gantt diagram editor - the *house building example*

GanttProject is mostly an editor for [gantt models](#) (see above). It also contains a viewer for [pert models](#) (see below). These read-only views are generated automatically.

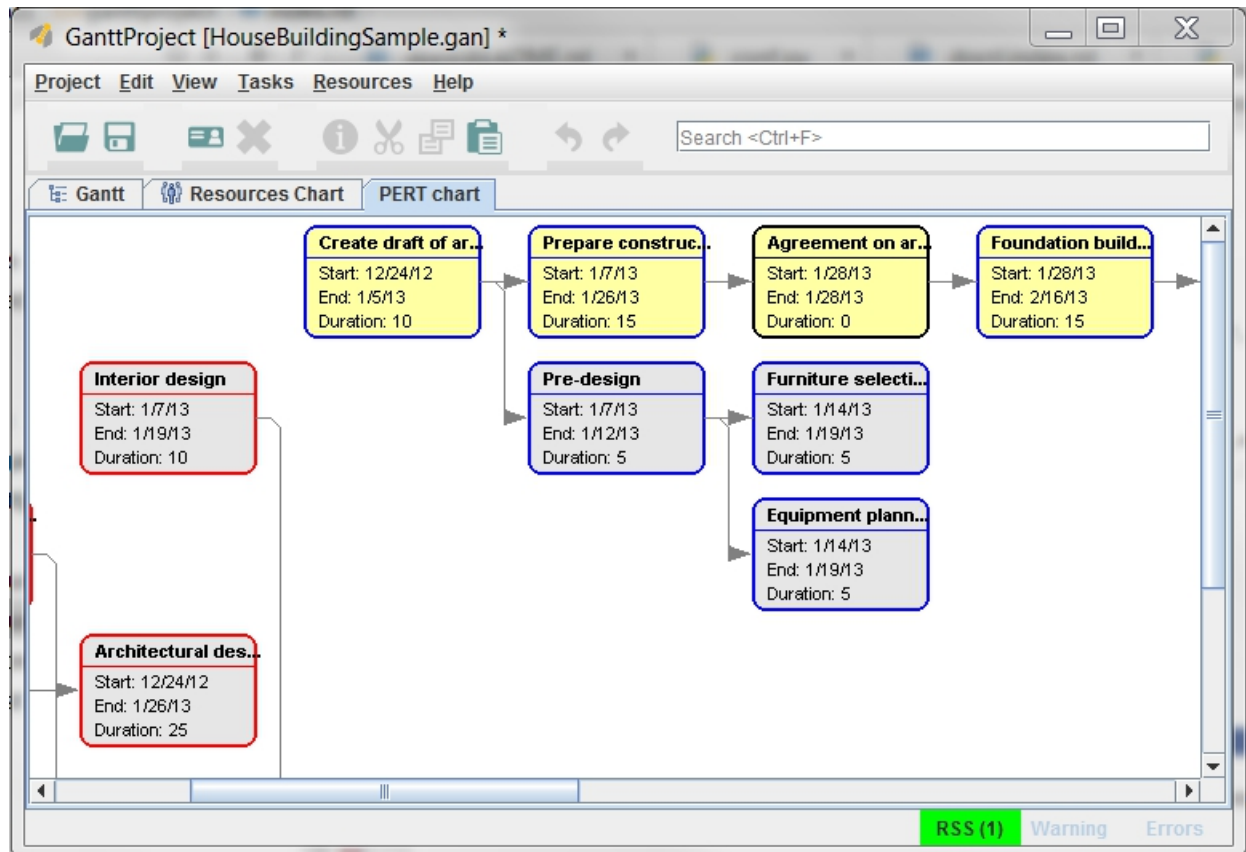


Fig. 1.2: *pert diagram viewer - the house building example*

GanttProject also supports [resource allocation models](#). *Resources* can be attached to *tasks*. The tool generates a resource allocation model showing the allocation of each resource along the project (see below).

## Interoperability

Microsoft Project and CSV files can be imported and exported from the Graphical User Interface (GUI). GanttProject can also generate PDF, PNG, JPEG.

The tool uses an .xml format reasonably simple so interoperability with other tools is also possible with some development.

The tool also has (a quite limited) Command Line Interface (CLI) allowing to export models in different formats.

## Installation

- Download the archive [web](#)
- Copy the archive into %SCRIBESTOOLS% and extract it here.
- Rename the directory to obtain %SCRIBESTOOLS%\GanttProject.
- Copy the ganttproject.cmd into the directory.



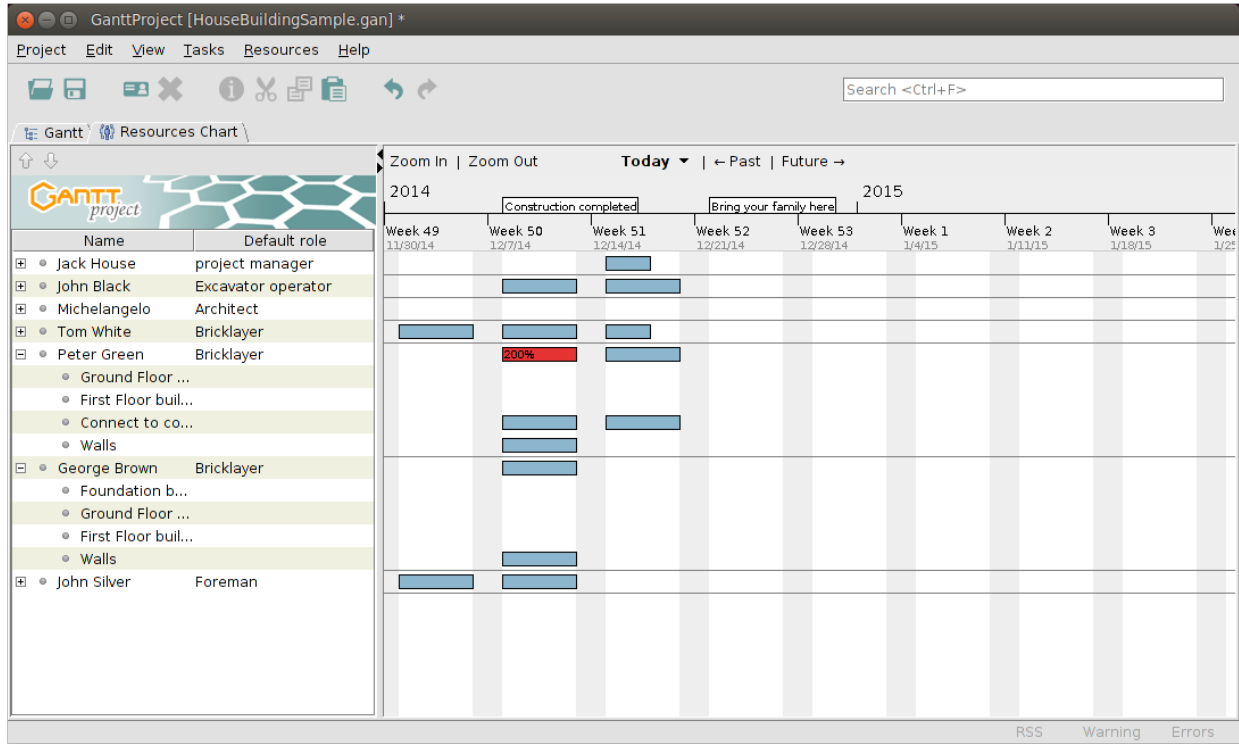


Fig. 1.3: resource allocation viewer - *house building example*

- Add %SCRIBESTOOLS%\GanttProject to the PATH variable

## Launching GanttProject

GanttProject is mostly used through its Graphical User Interface (GUI), but it also has a Command Line Interface (CLI) allowing some limited kind of automation. In a (new) shell you can type the following command to see the help about the CLI:

```
ganttproject.bat -h
```

The normal way to use the program is the GUI though. You can launch it just clicking on the executable or the shortcut that the installer might have created.

When launched, one way to see what GanttProject is all about is to load *house building example*. In order to do that use the menu “ Projects >> Open “ and select the HouseBuildingSample.gan file in the installation directory of the tools.

## Documentation

As far as we know there is no document describing GanttProject. The tool is nevertheless rather easy to use for someone acquainted with project management basics. There is an extended demonstration in the form of a 15' video ([youtube](#)). The *house building example* is also a valuable resource for learning how to use GanttProject.

You may also want to have a look at the following *unofficial* documents created by people outside of the project:

- “Apprendre Gantt project Ver 2.6” ([local](#), [web](#)) by Lycée du Dauphiné DD & JPG. This tutorial, in french, is rather good.

- “GanttProject Handbook 0.52” (local, web) by alexandre thomas. This handbook is rather obsolete and do not contains too much information.

## Example

GanttProject is delivered with an example of “house building” project. This project is store in the `HouseBuildingSample.gan` (local) file in the root of the installation directory.

## Graphical User Interface

This section describes the main panels of the GanttProject software.

### Project

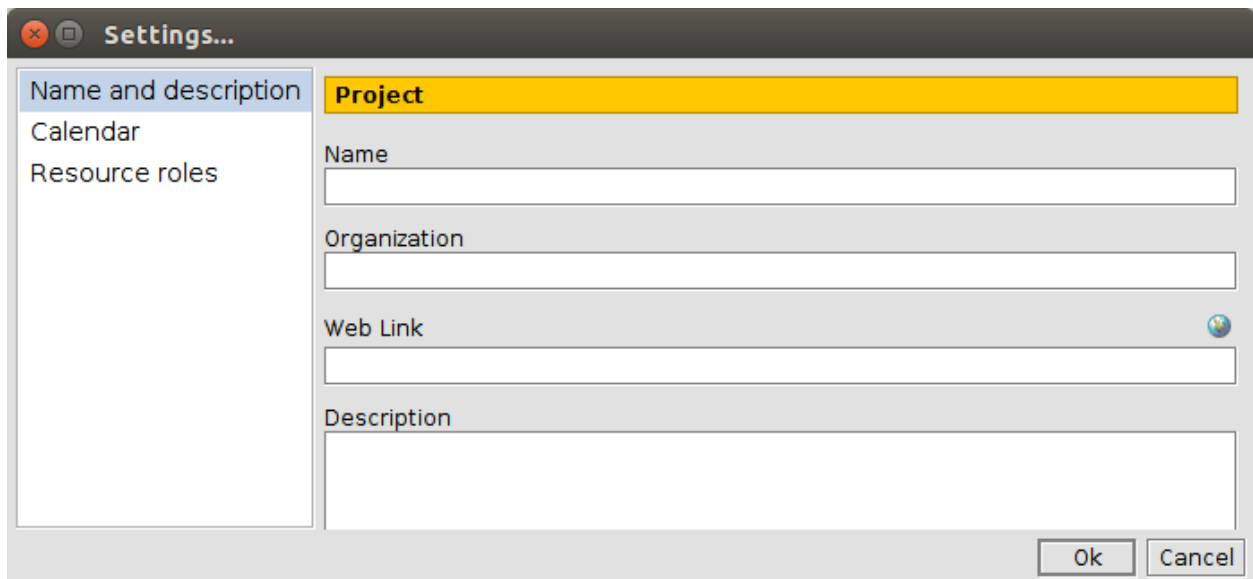


Fig. 1.4: Project\_NameAndDescription

### Tasks

### Resources

## Tools Interface

Gantt projects are saved in `.gan` files which are indeed `xml` files. This format can serve as an interoperability mean with other tools and constitutes therefore a potential interface for tools.

The `HouseBuildingSample.gan` file (local) corresponds to the House Building example. Developing a set of much simpler examples would be useful to serve as test cases and/or specification.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project name="" company="" webLink="" view-date="2012-12-16" view-index="0" gantt-divider-location=
3   <description/>
```

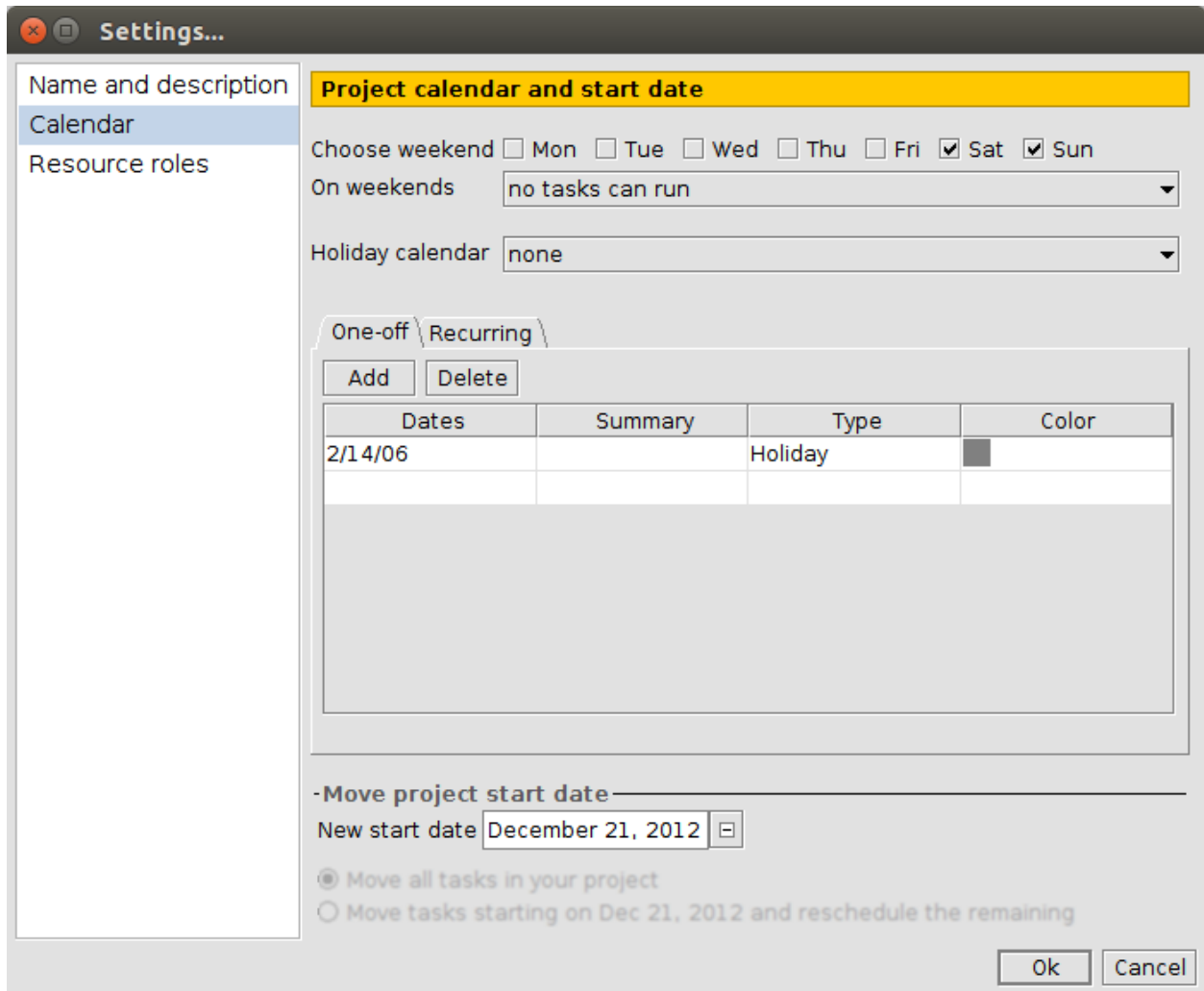


Fig. 1.5: Project\_Calendar

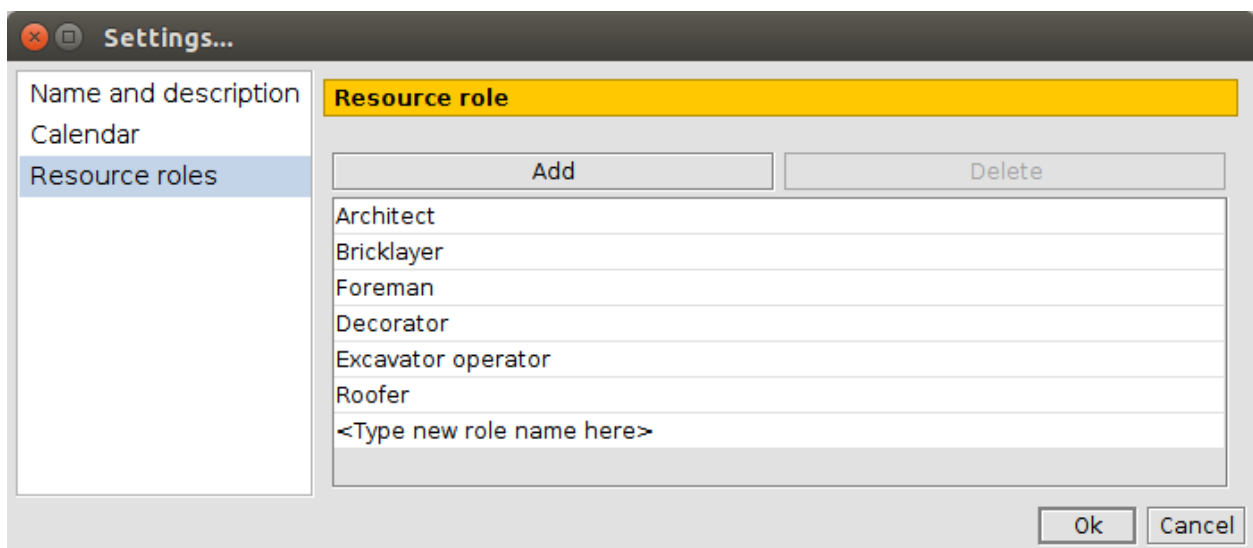


Fig. 1.6: Project\_ResourceRoles

Name	Begin date	End date
☐ • Architectural design	12/24/12	1/25/13
• Create draft of architecture	12/24/12	1/4/13
• Prepare construction documents	1/7/13	1/25/13
• Agreement on architectural plan	1/28/13	1/28/13
☐ • Interior design	1/7/13	1/18/13
• Pre-design	1/7/13	1/11/13
• Furniture selection	1/14/13	1/18/13
• Equipment planning	1/14/13	1/18/13
☐ • Construction phase	1/28/13	5/10/13
• Foundation building	1/28/13	2/15/13
• Ground Floor building	2/18/13	3/15/13
• First Floor building	3/18/13	4/12/13
• Roof	4/15/13	4/26/13
• Connect to communications	4/29/13	5/10/13
• Construction completed	4/29/13	4/29/13
☐ • Decoration phase	4/29/13	5/10/13
• Walls	4/29/13	5/3/13
• Furniture	5/6/13	5/8/13
• Bring your family here	5/13/13	5/13/13
• Survive the End of the World	12/21/12	12/21/12

Fig. 1.7: Tasks\_Hierarchy

**Properties for Furniture selection**

General | Predecessors | Resources | Custom Columns

Name: Furniture selection

Milestone: ☐

Scheduling options: [in this dialog](#) ▼

Begin date: January 14, 2013

End date: January 18, 2013

Duration: 5

Additional constraint:

Priority: Normal

Progress: 0

Show in timeline: ☐

Shape:

Colors: [Choose ...](#) Default

Web Link:

-Edit Notes...

Ok Cancel

Fig. 1.8: Tasks\_General

**Properties for Prepare construction documents**

General | Predecessors | Resources | Custom Columns

Add Delete

ID	Task name	Type	Delay	Link hardness
9	Create draft of architecture	Finish-Start	0	Strong

Ok Cancel

Fig. 1.9: Tasks\_Predecessors

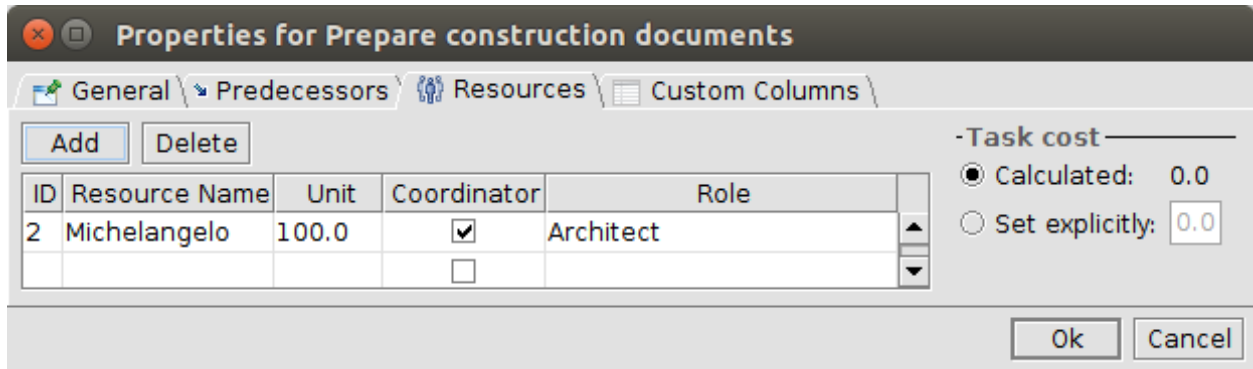


Fig. 1.10: Tasks\_Resources

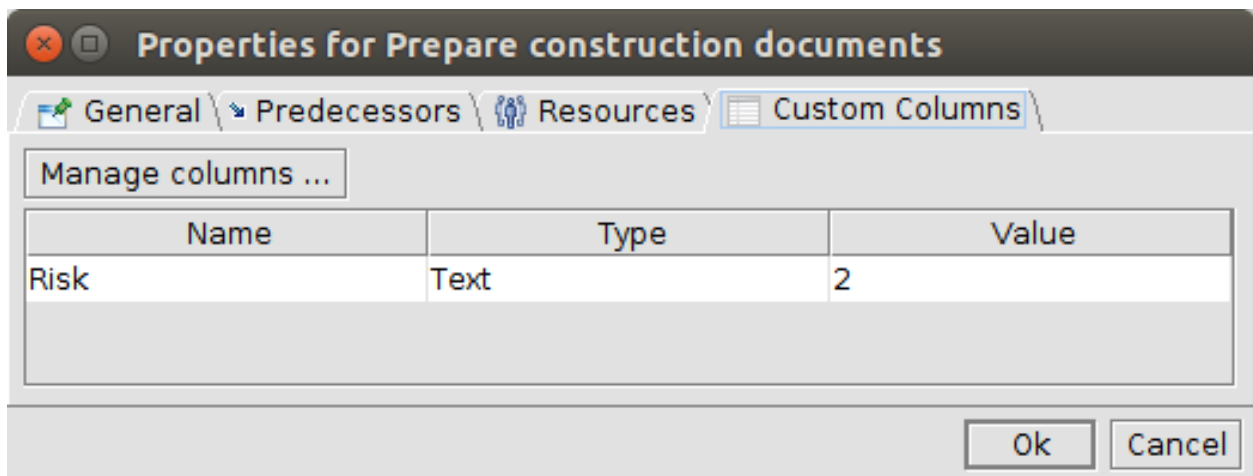


Fig. 1.11: Tasks\_CustomColumns

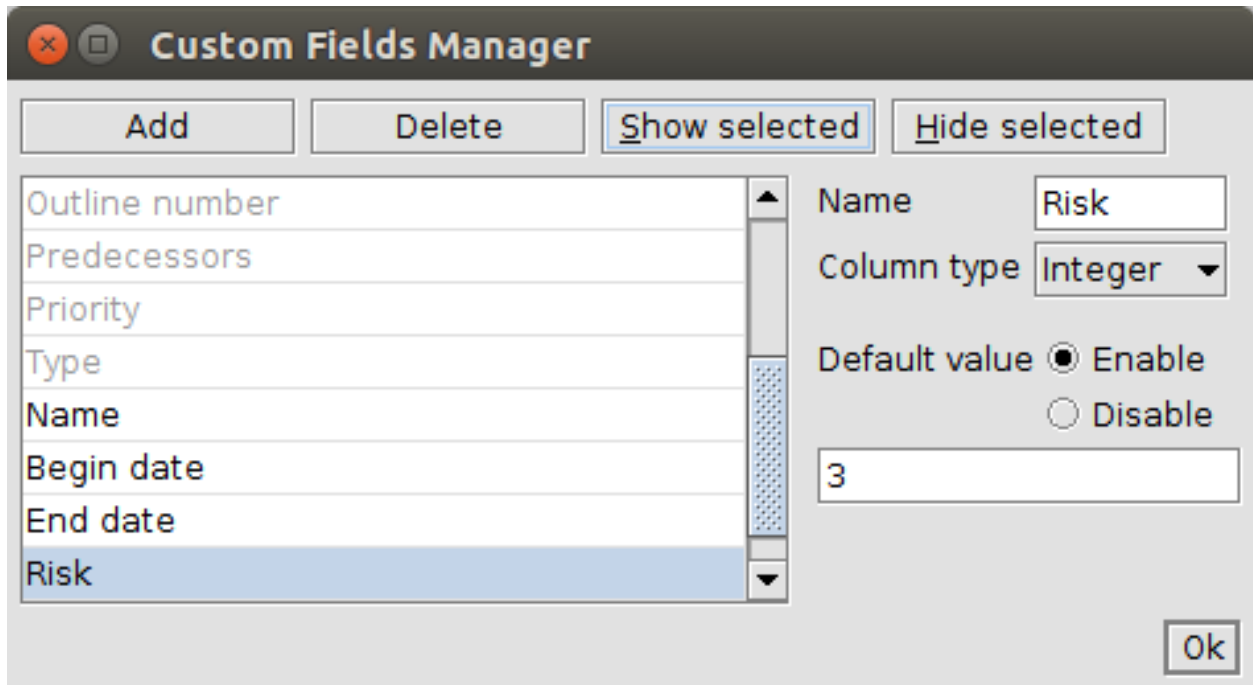


Fig. 1.12: Tasks\_CustomFieldsManager

```

4 <view zooming-state="default:6" id="ganttt-chart">
5   <field id="tpd3" name="Name" width="33" order="0"/>
6   <field id="tpd4" name="Begin date" width="33" order="1"/>
7   <field id="tpd5" name="End date" width="33" order="2"/>
8   <option id="taskLabelUp" value=""/>
9   <option id="taskLabelDown" value=""/>
10  <option id="taskLabelLeft" value="name"/>
11  <option id="taskLabelRight" value=""/>
12 </view>
13 <view id="resource-table">
14   <field id="0" name="Name" width="51" order="0"/>
15   <field id="1" name="Default role" width="48" order="1"/>
16 </view>
17 <!-- -->
18 <calendars>
19   <day-types>
20     <day-type id="0"/>
21     <day-type id="1"/>
22     <calendar id="1" name="default">
23       <default-week sun="1" mon="0" tue="0" wed="0" thu="0" fri="0" sat="1"/>
24       <only-show-weekends value="false"/>
25       <overridden-day-types/>
26       <days/>
27     </calendar>
28   </day-types>
29   <date year="2006" month="2" date="14"/>
30 </calendars>
31 <tasks empty-milestones="true">
32   <taskproperties>
33     <taskproperty id="tpd0" name="type" type="default" valuetype="icon"/>
34     <taskproperty id="tpd1" name="priority" type="default" valuetype="icon"/>

```

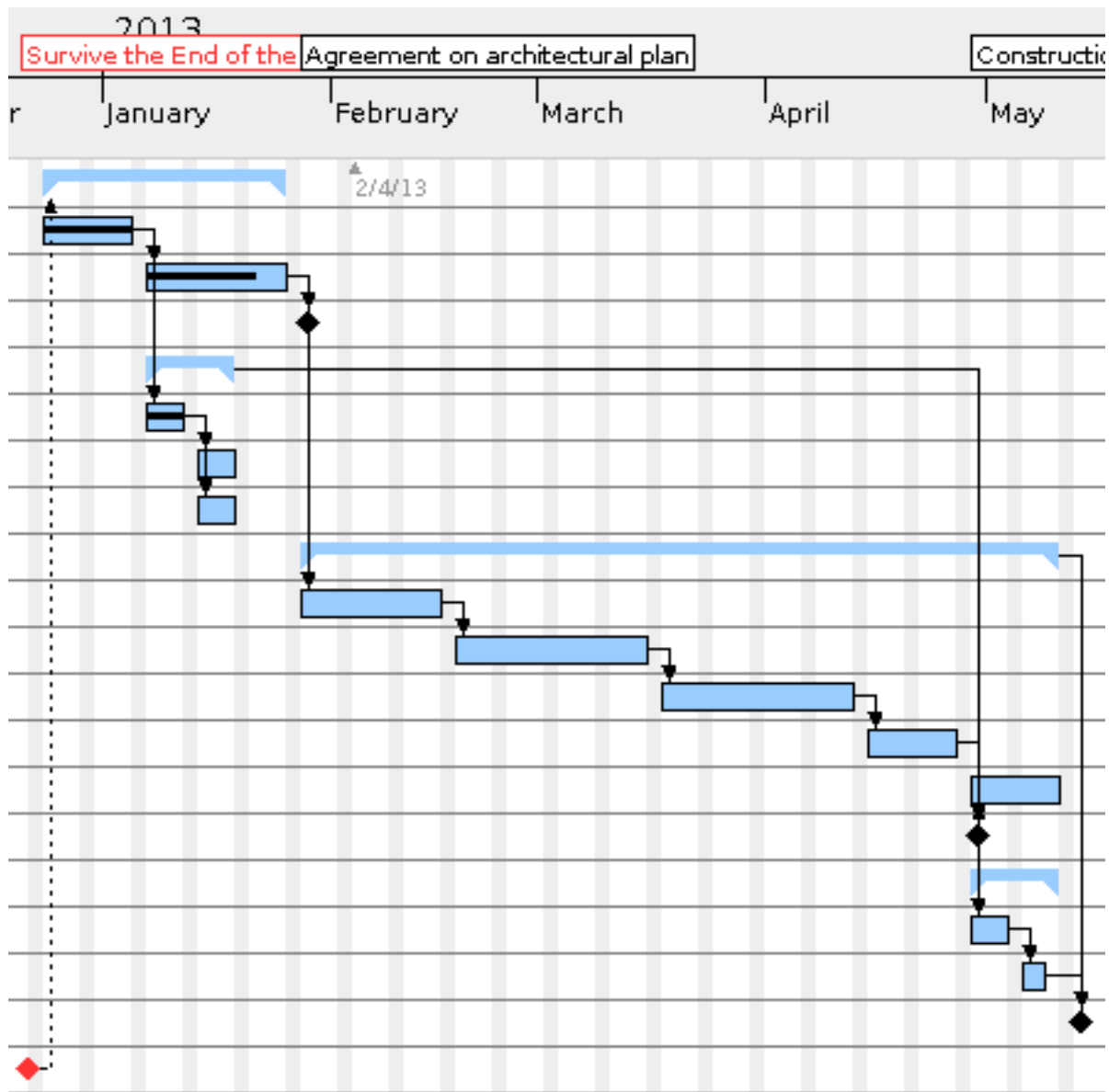


Fig. 1.13: Tasks\_GanttGraph



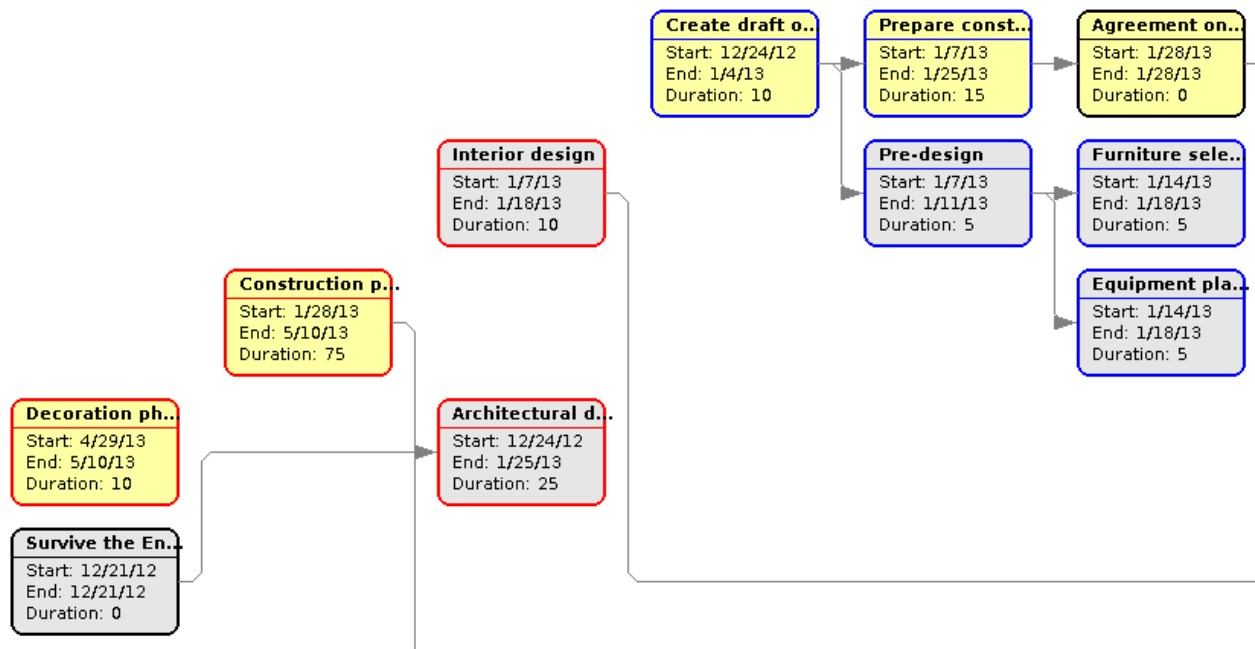


Fig. 1.14: Tasks\_PertGraph

The screenshot shows the 'Resources' dialog box with the 'General' tab selected. The fields are as follows:

- Name: Jack House
- Phone: 0044 077345456
- Mail: jack.house@myselfllc.net
- Default role: project manager
- Resource payment rate-
- Standard rate: 0

Buttons: Ok, Cancel

Fig. 1.15: Resources\_General

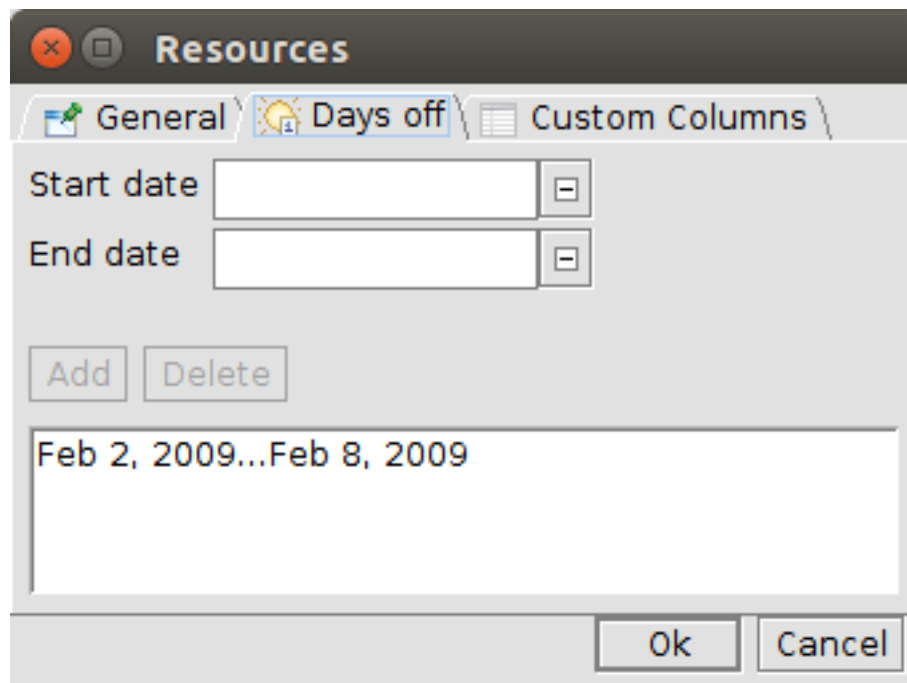


Fig. 1.16: Resources\_DaysOff

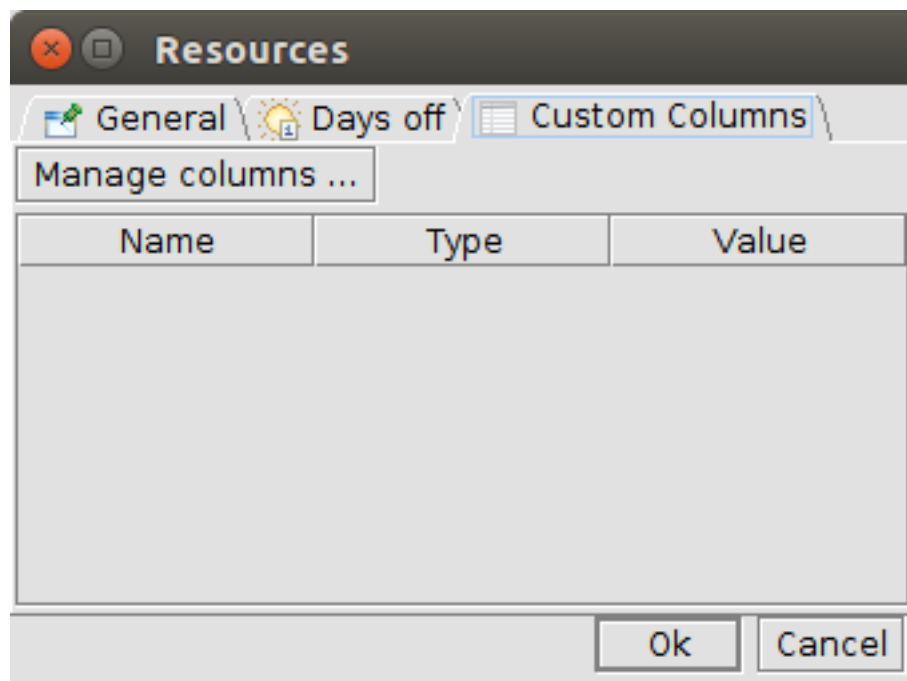


Fig. 1.17: Resources\_CustomColumns

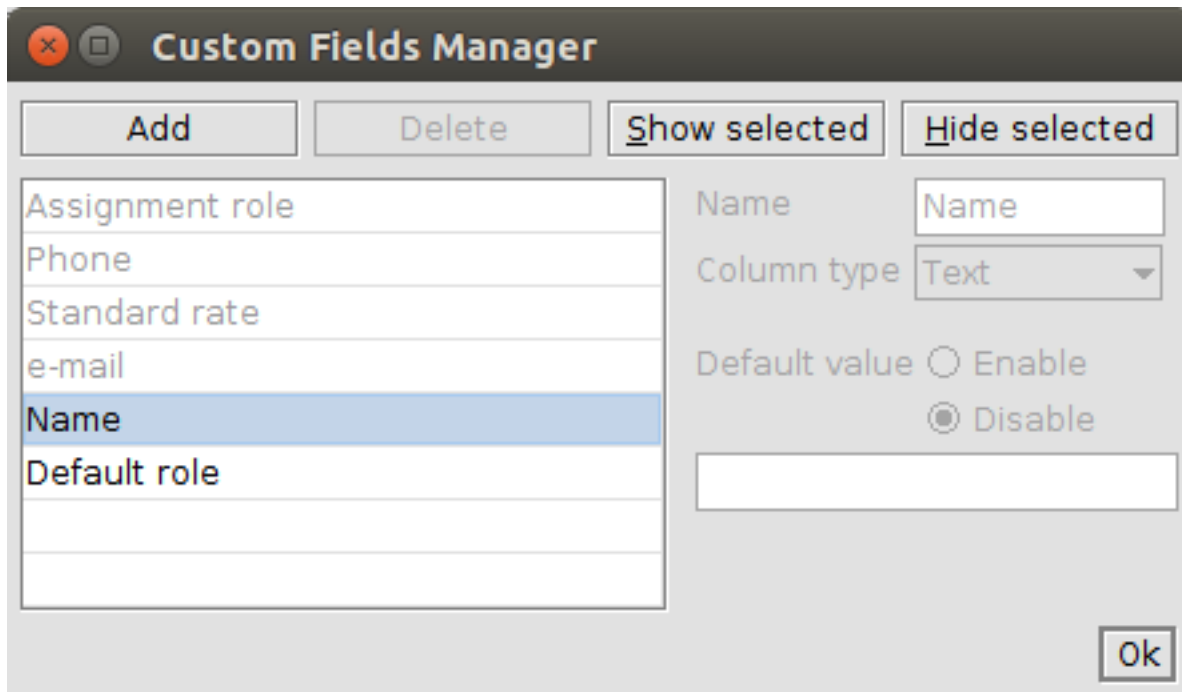


Fig. 1.18: Resources\_CustomFieldsManager

[-]	• Jack House	project manager
	• Create draft of architecture	
	• Pre-design	
	• Furniture selection	
	• Equipment planning	
	• Furniture	
[-]	• John Black	Excavator operator
	• Foundation building	
	• Connect to communications	
[-]	• Michelangelo	Architect
	• Create draft of architecture	
	• Prepare construction docum...	
	• Pre-design	
	• Equipment planning	
[+]	• Tom White	Bricklayer

Fig. 1.19: Resources\_TaskList

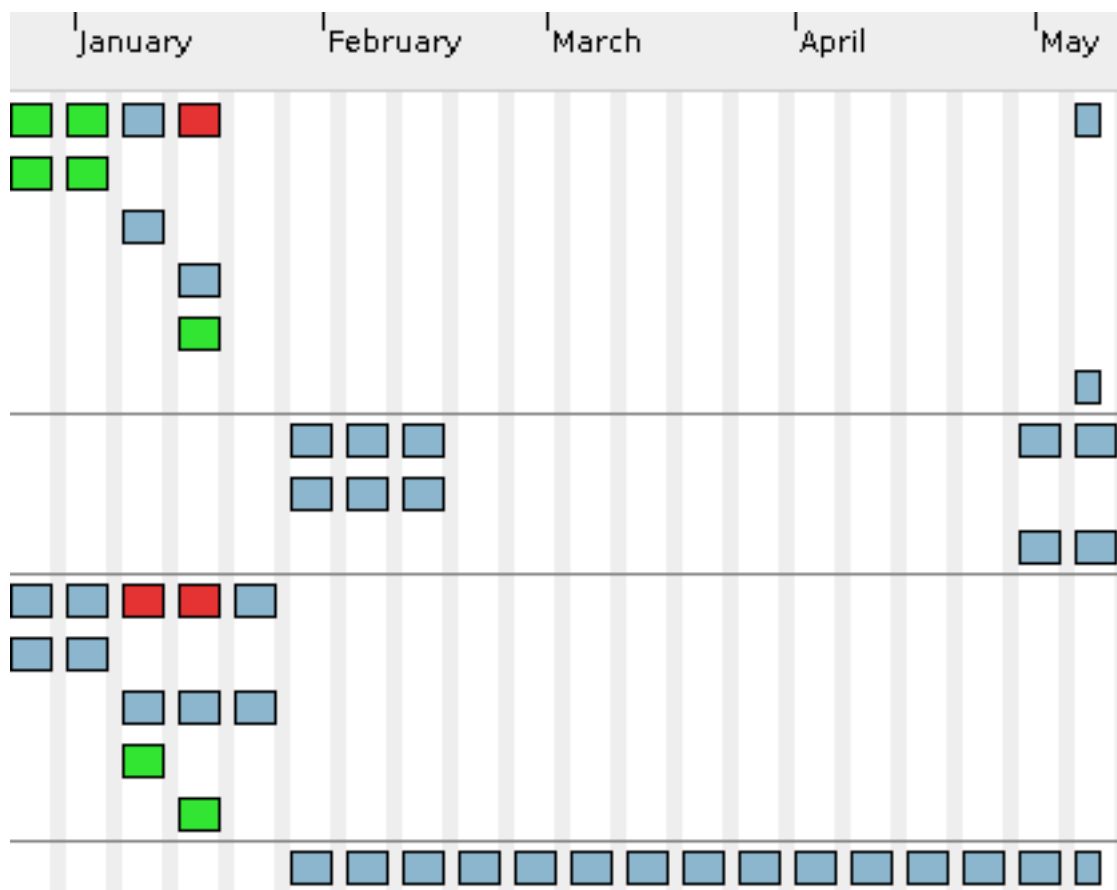


Fig. 1.20: Resources\_TaskAllocation

```

35     <taskproperty id="tpd2" name="info" type="default" valuetype="icon"/>
36     <taskproperty id="tpd3" name="name" type="default" valuetype="text"/>
37     <taskproperty id="tpd4" name="begindate" type="default" valuetype="date"/>
38     <taskproperty id="tpd5" name="enddate" type="default" valuetype="date"/>
39     <taskproperty id="tpd6" name="duration" type="default" valuetype="int"/>
40     <taskproperty id="tpd7" name="completion" type="default" valuetype="int"/>
41     <taskproperty id="tpd8" name="coordinator" type="default" valuetype="text"/>
42     <taskproperty id="tpd9" name="predecessorsr" type="default" valuetype="text"/>
43 </taskproperties>
44 <task id="0" name="Architectural design" color="#99ccff" meeting="false" start="2012-12-24" duration="10">
45     <task id="9" name="Create draft of architecture" color="#99ccff" meeting="false" start="2013-01-07" duration="5">
46         <depend id="10" type="2" difference="0" hardness="Strong"/>
47         <depend id="12" type="2" difference="0" hardness="Strong"/>
48     </task>
49     <task id="10" name="Prepare construction documents" color="#99ccff" meeting="false" start="2013-01-14" duration="5">
50         <depend id="17" type="2" difference="0" hardness="Strong"/>
51     </task>
52     <task id="17" name="Agreement on architectural plan " color="#000000" meeting="true" start="2013-01-21" duration="5">
53         <depend id="1" type="2" difference="0" hardness="Strong"/>
54     </task>
55 </task>
56 <task id="11" name="Interior design" color="#99ccff" meeting="false" start="2013-01-07" duration="10">
57     <depend id="6" type="2" difference="0" hardness="Strong"/>
58     <task id="12" name="Pre-design" color="#99ccff" meeting="false" start="2013-01-07" duration="5">
59         <depend id="13" type="2" difference="0" hardness="Strong"/>
60         <depend id="14" type="2" difference="0" hardness="Strong"/>
61     </task>
62     <task id="13" name="Furniture selection" color="#99ccff" meeting="false" start="2013-01-14" duration="5">
63     <task id="14" name="Equipment planning" color="#99ccff" meeting="false" start="2013-01-14" duration="5">
64         <notes><![CDATA[Embedded devices, kitchen, washing machine, dryer etc]]></notes>
65     </task>
66 </task>
67 <task id="7" name="Construction phase" color="#99ccff" meeting="false" start="2013-01-28" duration="10">
68     <depend id="20" type="2" difference="0" hardness="Strong"/>
69     <task id="1" name="Foundation building" color="#99ccff" meeting="false" start="2013-01-28" duration="5">
70         <depend id="2" type="2" difference="0" hardness="Strong"/>
71     </task>
72     <task id="2" name="Ground Floor building" color="#99ccff" meeting="false" start="2013-02-04" duration="5">
73         <depend id="4" type="2" difference="0" hardness="Strong"/>
74     </task>
75     <task id="4" name="First Floor building" color="#99ccff" meeting="false" start="2013-03-01" duration="5">
76         <depend id="5" type="2" difference="0" hardness="Strong"/>
77     </task>
78     <task id="5" name="Roof" color="#99ccff" meeting="false" start="2013-04-15" duration="10">
79         <depend id="18" type="2" difference="0" hardness="Strong"/>
80     </task>
81     <task id="16" name="Connect to communications" color="#99ccff" meeting="false" start="2013-05-06" duration="5">
82     <task id="18" name="Construction completed " color="#000000" meeting="true" start="2013-05-06" duration="5">
83         <depend id="6" type="2" difference="0" hardness="Strong"/>
84         <depend id="16" type="2" difference="0" hardness="Strong"/>
85     </task>
86 </task>
87 <task id="8" name="Decoration phase" color="#99ccff" meeting="false" start="2013-04-29" duration="10">
88     <task id="6" name="Walls" color="#99ccff" meeting="false" start="2013-04-29" duration="5">
89         <depend id="15" type="2" difference="0" hardness="Strong"/>
90     </task>
91     <task id="15" name="Furniture" color="#99ccff" meeting="false" start="2013-05-06" duration="5">
92         <depend id="20" type="2" difference="0" hardness="Strong"/>

```

```

93     </task>
94     <task id="20" name="Bring your family here" color="#000000" meeting="true" start="2013-01-01" />
95 </task>
96 <task id="49" name="Survive the End of the World" color="#ff3333" meeting="true" start="2012-12-31" />
97     <depend id="0" type="2" difference="0" hardness="Rubber"/>
98 </task>
99 </tasks>
100 <resources>
101     <resource id="1" name="Jack House" function="Default:1" contacts="jack.house@myselfllc.net" phone="+12025551234" />
102     <resource id="0" name="John Black" function="4" contacts="john.black@myselfllc.net" phone="+12025551234" />
103     <resource id="2" name="Michelangelo" function="0" contacts="mickelangelo@heaven.net" phone="+12025551234" />
104     <resource id="3" name="Tom White" function="1" contacts="tom.white@myselfllc.net" phone="+12025551234" />
105     <resource id="4" name="Peter Green" function="1" contacts="peter.green@myselfllc.net" phone="+12025551234" />
106     <resource id="5" name="George Brown" function="1" contacts="george.brown@myselfllc.net" phone="+12025551234" />
107     <resource id="6" name="John Silver" function="2" contacts="john.silver@myselfllc.net" phone="+12025551234" />
108 </resources>
109 <allocations>
110     <allocation task-id="9" resource-id="1" function="Default:1" responsible="false" load="50.0" />
111     <allocation task-id="12" resource-id="1" function="Default:1" responsible="true" load="100.0" />
112     <allocation task-id="13" resource-id="1" function="Default:1" responsible="true" load="100.0" />
113     <allocation task-id="14" resource-id="1" function="Default:1" responsible="true" load="50.0" />
114     <allocation task-id="15" resource-id="1" function="Default:1" responsible="true" load="100.0" />
115     <allocation task-id="1" resource-id="0" function="4" responsible="false" load="100.0" />
116     <allocation task-id="16" resource-id="0" function="4" responsible="true" load="100.0" />
117     <allocation task-id="9" resource-id="2" function="0" responsible="true" load="100.0" />
118     <allocation task-id="10" resource-id="2" function="0" responsible="true" load="100.0" />
119     <allocation task-id="12" resource-id="2" function="0" responsible="false" load="50.0" />
120     <allocation task-id="14" resource-id="2" function="0" responsible="false" load="50.0" />
121     <allocation task-id="1" resource-id="3" function="1" responsible="false" load="100.0" />
122     <allocation task-id="2" resource-id="3" function="1" responsible="false" load="100.0" />
123     <allocation task-id="4" resource-id="3" function="1" responsible="false" load="100.0" />
124     <allocation task-id="5" resource-id="3" function="5" responsible="false" load="100.0" />
125     <allocation task-id="6" resource-id="3" function="1" responsible="false" load="100.0" />
126     <allocation task-id="15" resource-id="3" function="1" responsible="false" load="100.0" />
127     <allocation task-id="2" resource-id="4" function="1" responsible="false" load="100.0" />
128     <allocation task-id="4" resource-id="4" function="1" responsible="false" load="100.0" />
129     <allocation task-id="16" resource-id="4" function="1" responsible="false" load="100.0" />
130     <allocation task-id="6" resource-id="4" function="1" responsible="false" load="100.0" />
131     <allocation task-id="1" resource-id="5" function="1" responsible="false" load="100.0" />
132     <allocation task-id="2" resource-id="5" function="1" responsible="false" load="100.0" />
133     <allocation task-id="4" resource-id="5" function="1" responsible="false" load="100.0" />
134     <allocation task-id="6" resource-id="5" function="1" responsible="false" load="100.0" />
135     <allocation task-id="1" resource-id="6" function="2" responsible="true" load="100.0" />
136     <allocation task-id="2" resource-id="6" function="2" responsible="true" load="100.0" />
137     <allocation task-id="4" resource-id="6" function="2" responsible="true" load="100.0" />
138     <allocation task-id="5" resource-id="6" function="2" responsible="true" load="100.0" />
139     <allocation task-id="6" resource-id="6" function="2" responsible="true" load="100.0" />
140 </allocations>
141 <vacations>
142     <vacation start="2009-02-02" end="2009-02-09" resourceid="1"/>
143 </vacations>
144 <taskdisplaycolumns>
145     <displaycolumn property-id="tpd2" order="-1" width="75" visible="false"/>
146     <displaycolumn property-id="tpd7" order="-1" width="75" visible="false"/>
147     <displaycolumn property-id="tpd8" order="-1" width="75" visible="false"/>
148     <displaycolumn property-id="tpd6" order="-1" width="75" visible="false"/>
149     <displaycolumn property-id="tpd10" order="-1" width="75" visible="false"/>
150     <displaycolumn property-id="tpd11" order="-1" width="75" visible="false"/>

```

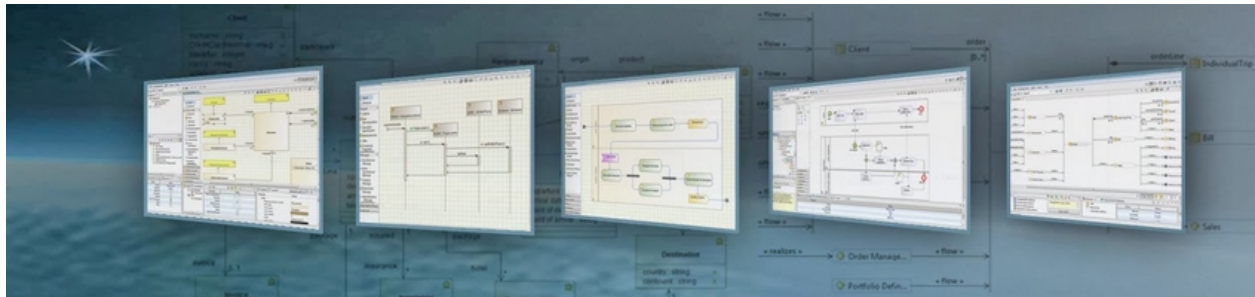
```

151     <displaycolumn property-id="tpd9" order="-1" width="75" visible="false"/>
152     <displaycolumn property-id="tpd1" order="-1" width="75" visible="false"/>
153     <displaycolumn property-id="tpd0" order="-1" width="75" visible="false"/>
154     <displaycolumn property-id="tpd3" order="0" width="122" visible="true"/>
155     <displaycolumn property-id="tpd4" order="1" width="122" visible="true"/>
156     <displaycolumn property-id="tpd5" order="2" width="121" visible="true"/>
157 </taskdisplaycolumns>
158 <previous/>
159 <roles roleset-name="Default"/>
160 <roles>
161     <role id="0" name="Architect"/>
162     <role id="1" name="Bricklayer"/>
163     <role id="2" name="Foreman"/>
164     <role id="3" name="Decorator"/>
165     <role id="4" name="Excavator operator"/>
166     <role id="5" name="Roofer"/>
167 </roles>
168 </project>

```

## Modelio

**Modelio** is an open source modeling environment supporting a wide range of modeling languages: **UML** and **BPMN** in a native form but also **SysML** or **TOGAF** for example.



Modelio is high extensible via java modules or python scripts . Modelio runs on Windows, Mac, and Linux.

## Features

Modelio if a full-fledged modeling environment supporting:

- model edition and validation for a wide range of modeling languages,
- model distribution over the web,
- model versioning with SVN (commercial edition only),
- documentation generation,
- code generation and or reverse engineering of C++, Java, C#, Hibernate, SQL, XSD...

Modelio can be extended to add more features via **java modules** or **python scripts**.

## Installation

**Modelio** exists both in an open source and commercial version. You need a licence in the later case.

---

**Tip:** If you have a licence code you can install both an open source version and a commercial version on the same machine. Just use different directories such as %SCRIBESTOOLS%\ModelioOpen and %SCRIBESTOOLS%\ModelioCommercial.

---

**Attention:** The architecture (32bits or 64bits) of your modelio and java installations much match. If this is not the case then launching modelio will likely generate a java error exit code 13. If you want to check which architecture is used by your java installation open a shell and type `java -version`. If this information is important it will be displayed on the 3rd line.

### Installing modelio open-source

- Go to the modelio [download center](#) and download the archive for your platform.
- Copy the archive into the directory that will contain the software (e.g. %SCRIBESTOOLS%\ModelioOpen).
- Extract the archive and then rename it into something like %SCRIBESTOOLS%\ModelioOpen (you may want later to install the commercial version as well or another version).

### Installing modelio commercial

**Attention:** You need either a *node-locked licence* or a *floating licence* to execute a commercial version:

- **node-locked licence (personal use).** You may have received a licence code that looks like 3BA5A-AVAT9-RUEJD-WK4LP. In this case after the installation you will have to enter this licence code following this procedure ([web](#)). Beware: a licence code can be used only in a single machine and when used on a machine it is *impossible* to move to another machine.
  - **floating licence (with organizations).** If you are running modelio for the first time in the context of an organization (e.g. at UGA), **Modelio** may ask you to enter the reference of a *licence server* following this procedure ([web](#)). You will have to enter the information of the licence server (e.g. at the UGA the host name is 152.77.82.15 and the port is 6200

- **Register to modelio community ([web](#)).** This is free. This will allow you to download commercial products but also to participate in modelio forums, etc.
- **Go to model the *ultimate solution* download space ([web](#)) and** select the installer for your platform.

## Launching Modelio

If a shortcut has been created to launch **Modelio** (it depends on your platform), you just have to click on it. Otherwise you can click on the executable (modelio or modelioexe) in the installation directory.

Modelio can be launched within a script or from a shell with a command line like that (here the installation directory is %SCRIBESTOOLS%\ModelioOpen\):

```
%SCRIBESTOOLS%\ModelioOpen\modelio.exe      # modelio on unix
```

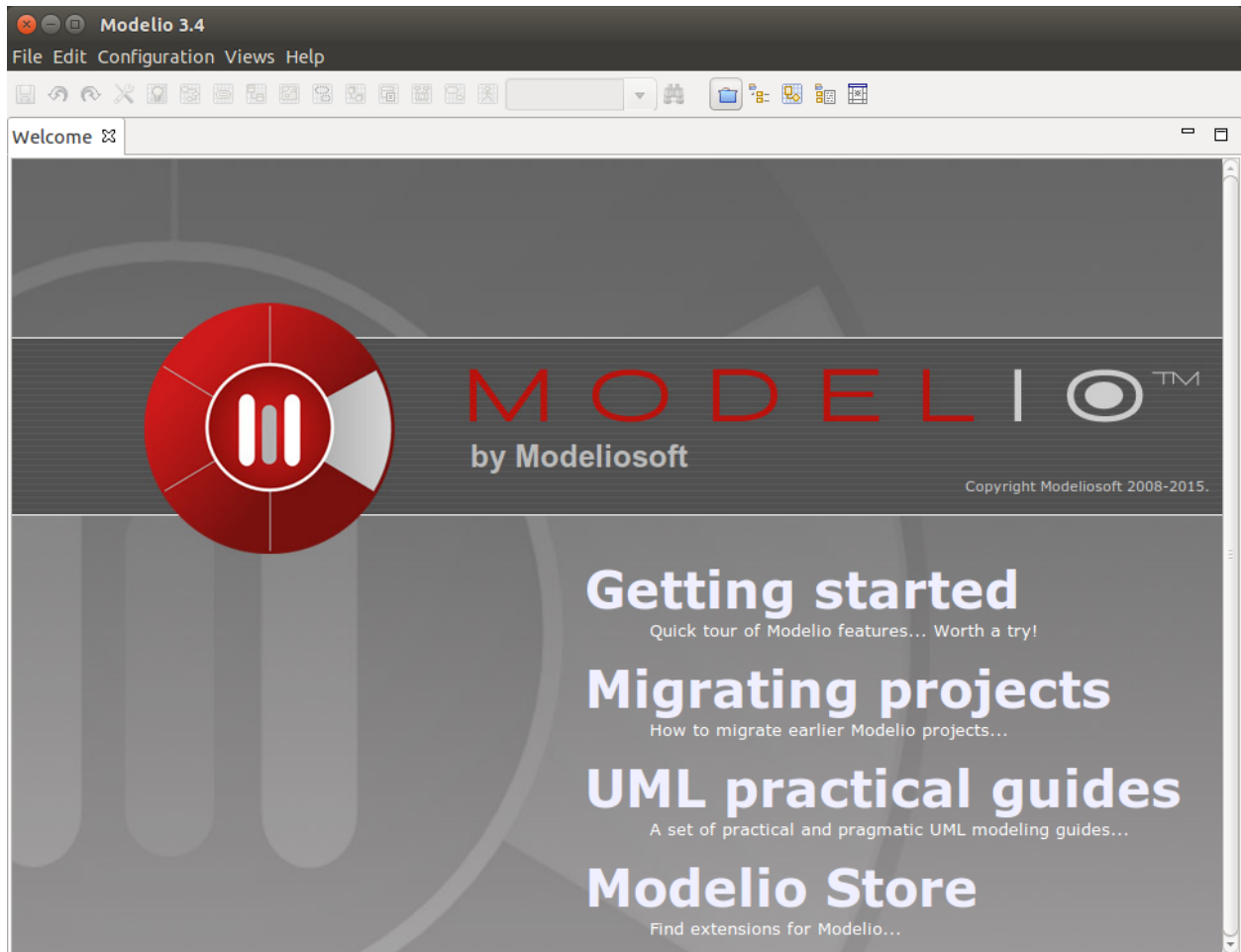
It could be wise to create a command to add parameters (or to change the shortcut on windows) in order to display the console window and run the **Modelio** in debug mode (this allows to have more messages in the console in case of modelio errors):



```
%SCRIBESTOOLS%\ModelioOpen\modelio.exe -mdebug -consoleLog
```

If you get an error ‘exit code 13’ it is likely that the java and modelio architecture (32 bits or 64 bits) do not match.

Note that on the first launch [Modelio](#) could take quite some time (up to a few minutes) to start. This could be the case on a slow computer or if your home directory is on a remote server. Modelio creates the ‘module catalog’ and this involves big files. After this, [Modelio](#) shows a “Welcome” page. You can simply close it.



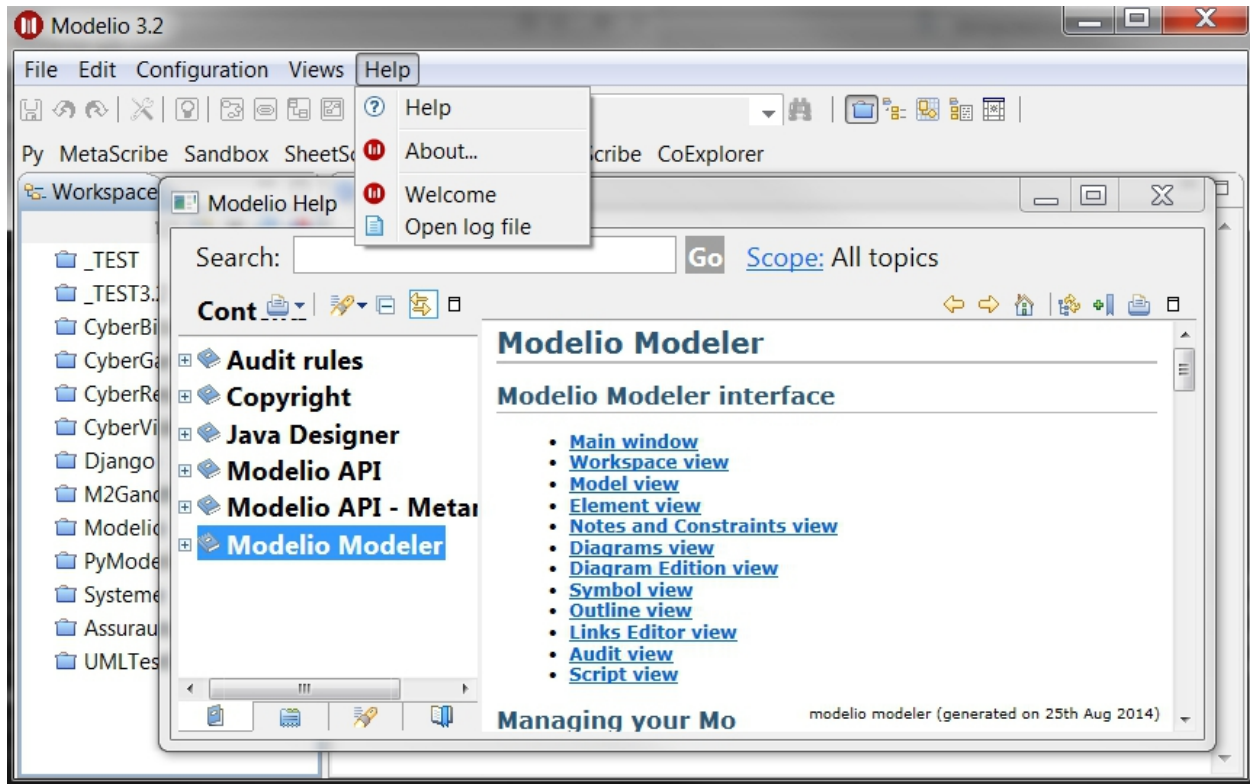
## Documentation

Documentation on [Modelio](#) is available from the **Help > Help** menu of [Modelio](#) as shown below. You will most probably want to use the section **Modelio Modeler** if you are just interested in using modelio.

Different kind of documentation is also available on the web:

- [user documentation](#)
- [developer documentation](#) (for developing extensions)
- [FAQ](#)
- a few [tutorials](#) (including some [videos](#))

**Tip:** The [forums](#) provides also a very valuable source of information! Use the “Search” tab if you want to search



some information about a given topic.

---

## Working with Projects

Just like in many software engineering environments Modelio is based on the notion of “Workspace” (a directory) which contains “Projects” (subdirectories inside the workspace directory).

### Do Not ...

The internal structure of “Projects” is entirely managed by Modelio and must be considered as a black box.

- **DOT NOT** makes any changes in a project directory.
- **DOT NOT** put in a DropBox or GoogleDrive in order to share the project with someone else. Various problems have been reported in the past, probably because of synchronisation processes.
- **DOT NOT** use git or versioning tools. Models are complex artefacts and merging will fail. You might also encounter problems if you use git for “backups”: git does not save empty directories and Modelio use a lot of them...

### Project Archives

The only safe way to work with modelio is:

- to use the modelio user interface to make changes during a modeling session,
- to use “project archives” to make project backups or exchange projects.



The screenshot shows the Modelio forum interface. At the top, there's a banner with the Modelio logo (a red circle with three vertical bars) and the text "modelio the open source modeling environment". To the right of the banner, there are social media links (RSS, Twitter, Facebook, YouTube) and a box for "Web Model Publisher Documentation generator (Free Modelio module)". Below the banner is a navigation bar with links: Home, About Modelio, Downloads, Community, Documentation, and a search bar. The main content area has tabs for Index, Recent Topics, Rules, and Search. A login section follows, with fields for Username and Password, a "Remember me" checkbox, and a "Login" button. Below the login section, it says "Modelio forum" and "(2 viewing) jmfavre, (1) Guest". A "Board Categories" dropdown menu is also present. The forum is divided into two main sections: "General" and "Usage". The "General" section contains three categories: "Announcements" (21 Topics, 4 Replies), "Modelio website" (14 Topics, 30 Replies), and "opensource-uml.org" (3 Topics, 5 Replies). The "Usage" section contains two categories: "General help" (298 Topics, 1136 Replies) and "Installation" (46 Topics, 201 Replies). Each category entry includes a brief description and the name of the moderators.

**modelio**  
the open source modeling environment

Follow us: [RSS](#) [Twitter](#) [Facebook](#) [YouTube](#)

Web Model Publisher  
Documentation generator  
(Free Modelio module)

Home About Modelio Downloads Community Documentation search...

Home Forum Index

Index Recent Topics Rules Search

Welcome, **Guest**  
Username:  Password:  Remember me ☐ [Login](#)  
Forgot your password? Forgot your username? Create an account

Modelio forum  
(2 viewing) jmfavre, (1) Guest

Board Categories  Go

**General**

<b>Announcements</b> Latest information, news from Modelio. Moderators: andy	21 Topics	4 Replies	Last Post: Modelio 3.2.1 released! by pan 2 months, 2 weeks ago
<b>Modelio website</b> Post here any suggestion, question or difficulty encountered on Modelio websites. Moderators: andy	14 Topics	30 Replies	Last Post: Re: Can't access at tutorial by gmistral 1 month, 1 week ago
<b>opensource-uml.org</b> Questions and information about opensource-uml.org website	3 Topics	5 Replies	Last Post: The Java software Spring Frame ... by pan 3 months, 1 week ago

**Usage**  
Learn how to use Modelio

<b>General help</b> Forum for general discussions that don't fit in the other forums. Moderators: tma, ebr	298 Topics	1136 Replies	Last Post: Re: Constraint expressions in ... by ebr 2 days, 5 hours ago
<b>Installation</b> Questions and information about Modelio installation and configuration. Moderators: cde, osl	46 Topics	201 Replies	Last Post: Use modelio in eclipse . Openi ... by matheus 2 months, 3 weeks ago

**Extensions**

A “project archive” is a zip file created and managed by Modelio. **Do not unzip these files.** Use instead “Import” and “Export” functions of Modelio available in from modelio [Workspace View](#). In practice these commands are available *when no project is open*:

- import: menu “File > Import a project”
- export: contextual menu on a project `Export a project` (the project should be closed).

At the end of a modeling session, it could be wise to “Export the project”, that is save the project in a .zip file like “MyProject-3.zip”. Saving in it the workspace if fine. Just increment the version number each time to keep an history of your work. If you want to restore a given version you will just have to “Import the project” (you may want to save first the current one).

If you want to work with someone else, just send the last version. Your partner will:

- import the project archive ,
- work on it with modelio,
- make some backups (via exports) if necessary,
- send the last version to you when finished.

If you do not want to guest lost you should increment the version number each time you save an archive.

## Scripting with Modelio

This section shows how to extend [Modelio](#) with its scripting feature. [Jython](#) is the scripting language. [Jython](#) is just Python running on a Java virtual machine. Otherwise this is the same programming language. Simply put [Jython](#) programs can call all python libraries but also all java libraries... Modelio is written in java but [Jython](#) makes it possible to use its full API and this without the burden to compile, package, deploy java plugins.

The first sub section shows how to use [Jython](#) interactively in the console. The next section shows how to store scripts in .py files to ease development and use regular text editors or python environments.

### Using the console

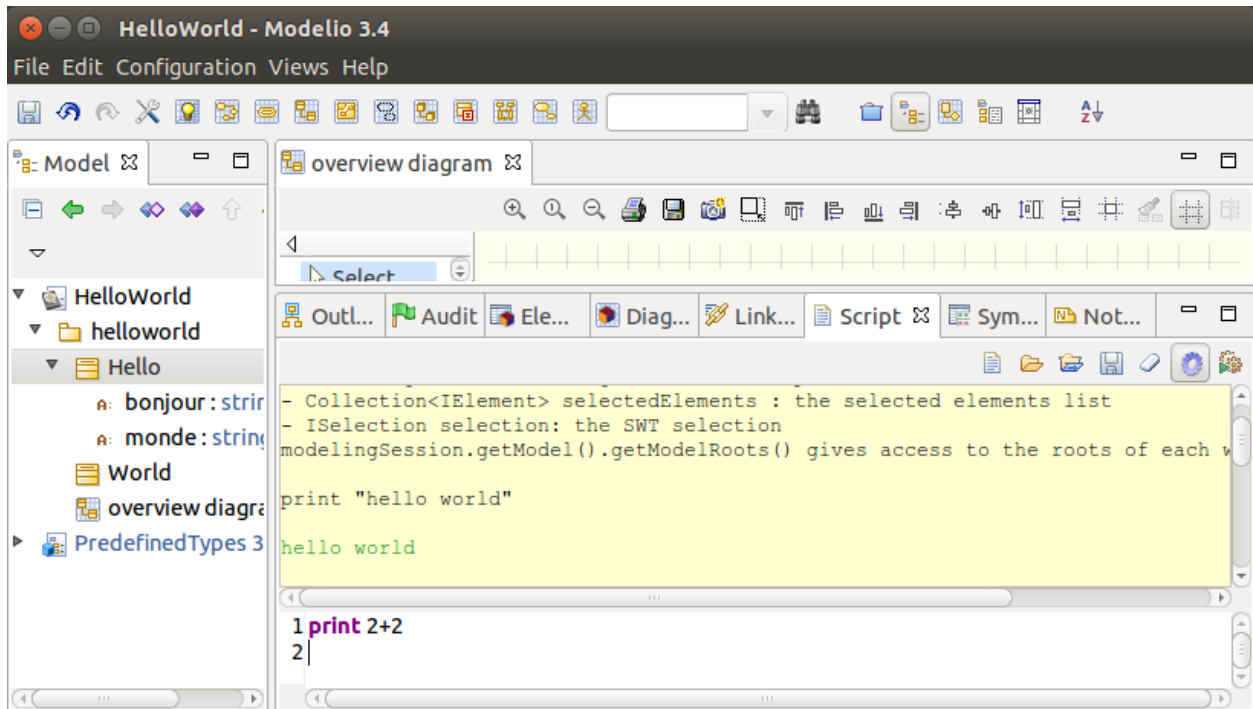
In this section shows how to use the script console of Modelio. This is a really excellent feature of Modelio as this allowed to play interactively with models, explore the metamodel, experiment with transformations, etc.

- Open an existing project or create a new one with a few classes and attributes. This will make it possible to run the macro below on some example.
- Using the browser on the left, select some classes that contains attributes.
- Choose the menu `View > Script`. This open the python engine (this takes a few seconds the first time. A [Jython](#) interpreter is loaded).
- Type the following line in the console and then press `Ctrl-Enter`:

```
print "Hello World"
```

- Observe the result. Try other expressions such as `print 'a'*3`.
- Then copy-paste the following program to the console and press `Ctrl-Enter`:

```
for c in selectedElements:
    if isinstance(c,Class):
        attributes = c.ownedAttribute
```



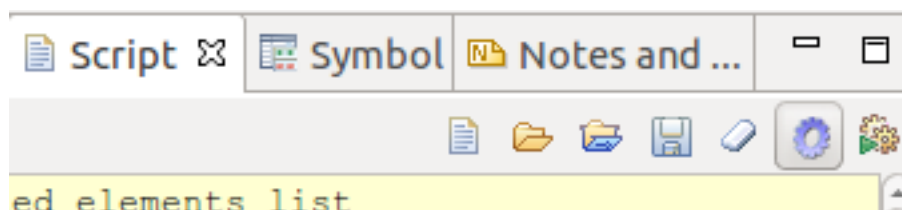
```
print '<h2> %s </h2>' % c.name
print 'The class %s has %i attributes: <ul>' % (c.name, len(attributes))
for a in attributes:
    print '<li> %s : %s </li>' % (a.name, a.type.name )
print '</ul>'
```

- You might get an error message like the following one (displayed in red):

```
'... line 1 ... SyntaxError: mismatched input ' ' expecting EOF'
```

If so this is due to some extra spaces in the copy paste. Python is based on the indentation to represent blocks. If there are some extra spaces before the first line (for c in ...) the interpreter will complain: a top level statement is expected (hence no spaces).

As you can see, the code you have just pasted has disappeared when you press Ctrl-Enter. This is n really convenient... Press the icon that looks like a 'blue gearing' (the penultimate logo in the console toolbar). The tooltip on this logo is 'Activate/Deactivate debug mode'.



In fact, this mode just allows to keep the text in the console instead of erasing it when Ctrl-Enter is pressed. Copy the program above again, check for spaces and press Ctrl-Enter again. Now the program stays in the console. If there are still some space problems you can correct the program there.

- At some point you will get the following error:

```
'... line 6 ... expecting COLON'
```

This is because all composed statements (for ... : , if ... :, etc) must have : at the end to indicate that a new block is going to start. Java programmers often tend to forget this : and will get this error. Otherwise the python syntax is rather straight forward. Correct the program by adding : after for a in attributes and press Ctrl-Enter.

A little “html generator” has been developed in a few lines of code. If you select some classes in modelio browser (on the left pane) and run the program you will see the list of classes with their attributes in html. Obviously if you want to see a nice result you should put this in a .html file and launch a browser, but this is another story (googling something like “python write lines in file” will probably bring you close to the solution).

Writing code in the console is very convenient for testing code snippets interactively. However if you want to develop more complex programs and deliver it to other users, Modelio ‘macros’ should be used.

## Developing macros

Macros are just Jython programs saved in a file. That’s all.

The only things to know is where to put these files and where to register it. Macros can be located in three difference places (but not elsewhere, this is a current limitation of Modelio):

- **Project macros.** This location is only useful for macro that are specific to a particular projects. Most of the time this is not the case. So the project location is seldom used.
- **Workspace macros.** These macros can be used in all projects within this location. *This option is the most convenient and this is the one that you are going to use.*
- **System macros.** These macros are located in your .modelio directory, but are normally not for users-defined macros.

Macros are just python files with the .py prefix. Workspace macros are stored in the macros directory of the workspace. In order to make macros accessible from the Modelio user interface, the macro should be registered in the ‘macro catalog’: the XML file named macros/.catalog. As an illustration the following .catalog file will register the helloworld.py macro.

```
<?xml version="1.0" ?>
<catalog>
  <script name="HelloWorld" path="helloworld.py" icon-path="" show-menu="true" show-toolbar="true">
    <description></description>
  </script>
</catalog>
```

There is one <script /> element per macro. Just create the file macros\helloworld.py as following:

```
print 'hello world!'
```

Modelio should be restarted in order for the macros to be registered. But don’t worry, this should be done only once! It is indeed not very common to add macros. Restart modelio and open a project (in the workspace containing the macros directory just modified). A button HelloWorld should appear in the toolbar just below Modelio menu bar. Pressing it will execute the content of the file. The good point is that now that the macro is registered everything becomes really handy. Use your favorite editor (notepad++, vi, gedit or a python environment like PyCharm (see ref:PyCharm) and change the file as you want. Just press again the HelloWorld button. The code is executed immediately. No compilation, no packaging, no deployment. Try to develop plugins written in java for eclipse and you will see the difference... The benefit of python is right there.

## Learning Jython

Learning Jython is just learning Python as this is actually the same language. There are plenty of resources available on the web. Just google “python” with a few terms and you have good chance to get the answer to your question.

To start look at the [Documentation](#) section of the [Python](#). You will find some useful cheat sheets. Have also a look at the slides “J/Python in a Nutshell”.

In fact for creating simple macros for Modelio, Jython/Python should not represent an issue. One just have to know how to write:

- conditional statements: `if cond: else:`
- loops `for e in expr:`
- functions `def f(x):`

The challenge is in fact to deal with the [Modelio API](#) and in particular with [Modelio metamodel](#).

## Modelio metamodel

The [Modelio API](#) is based on [Modelio metamodel](#). This metamodel integrates in a single metamodel 3 languages:

- [UML](#) the Unified Modeling language, an international standard.
- [BPMN](#) the Business Process Modeling Notation, an international standard.
- “Analyst” a modelio proprietary language for requirements engineering.

The metamodel is made of about [300 metaclasses](#) spread in about 30 packages.

[Modelio metamodel](#) is proprietary. Although is neither compatible with the standard [UML metamodel](#) nor the standard [BPMN metamodel](#) but it support most of the features of these languages (in an integrated way).

---

**Note:** The metamodel links depend on the version. Adapt the URLs according to the version of the UML/Modelio metamodel you use.

---

There are three ways to explore Modelio metamodel:

- browsing the “metamodel documentation”.
- browsing the “API javadoc”.
- using the “CoExplorer”.

## Metamodel documentation


The metamodel is described in the [modelio metamodel documentation](#). Each metaclass is described in a web page containing a class diagram centered around the metaclass. See for instance the page for the “[class](#)” metaclass.

Note that the images are clickable. The best way to find a metaclass in this documentation is probably to use the [metamodel index](#) (and using the “search” feature of the browser).

## API javadoc

Using the [modelio API javadoc](#) is another alternative. There are more than 500 classes and 67 packages in this API, but this is because the API provides many other services. The metamodel is just about “model management”. In other words, all metaclasses are in the javadoc but there are more classes.




ModelioSoft

Metamodel user guide

< Previous
Home
Next >

### INFORMATIONS

**Status**  
Release

**Category**  
Modelio user guide

**Version**  
9026 (Modelio 3.4)

**Author**  
Modeliosoft

**Title**  
Metamodel user guide

**Subject**

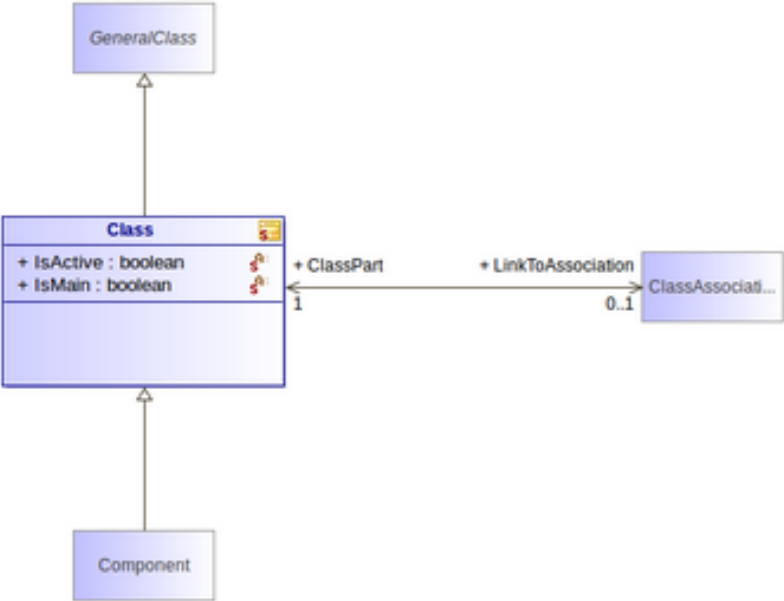
### SHORTCUTS

- ▶ Home
- ▶ Site Map
- ▶ Model Index
- ▶ Table of Figures

## Class

Description of a set of objects that share the same Attributes, Operations, methods, relationships and semantics.  
The Class is the main concept used in object-oriented modeling. It specifies which Instances can exist in an application.

In Modelio, a Class is owned by a Namespace (ModelTree) that can be a Package or a Class.



```

classDiagram
    class GeneralClass
    class Class {
        +IsActive : boolean
        +IsMain : boolean
    }
    class Component
    class ClassPart
    class ClassAssociati...
    GeneralClass <|-- Class
    GeneralClass <|-- Component
    Class "1" -- "0..1" ClassPart : + ClassPart
    Class "1" -- "0..1" ClassAssociati... : + LinkToAssociation
    
```

**Figure 148 : Class (architecture\_autodiagram)**

Attribute	Description
boolean IsActive [1..1]	Specifies whether an Object of the Class maintains its own thread of control. If true, then an Object has its own thread of control and runs concurrently with other active Objects. If false, then Operations run in the address space and under the control of the active Object that controls the caller.
boolean IsMain [1..1]	A main Class is a Class whose unique instance represents the application.

Fig. 1.21: Documentation of the “class” metaclass



Have a look to the “class” `metaclass javadoc` and compare it the corresponding “UML” documentation. As you can see the content is the same although the javadoc page just reflects that the API is implemented in java. Naming conventions allows to go from the API to the metamodel seamlessly.

## CoExplorer

Another alternative, most probably the most convenient one, is to use the CoExplorer plugin. This plugin (actually a jython macro) should be installed in order to be used. Once installed select the elements to be explored in modelio and then press the `CoExplorer` button in the top right toolbar.

The CoExplorer allows to explore at the same time a model and the metamodel.

---

## TODO

to be documented

---

## Collaborative Modeling with SVN

Thanks to the `TeamworkManager` module Modelio can store projet fragments in a remote SVN repository. `TeamworkManager` allows various users to work on the same model at the same time.

**Warning:** `TeamworkManager` feature is available only in some commercial versions of modelio. The `ultimate edition` provides this feature. It is not easy to configure behind a firewall. This section is reserved to advanced users only.

The documentation to create and use SVN fragments with `TeamworkManager` is available in the menu help of Modelio (but not on the web). It can be found in the section `Modelio by Modeliosoft extensions > Teamwork` as shown in the figure below.

### Creating a SVN repository

This step is necessary only to create your own repository. *This is not necessary if someone give you access to a shared repository.*

To create the SVN directory itself you need to have a SVN server. If you don't have one, you can use Assembla which is a free-svn provider on the cloud (see [Assembla](#)).

### Connecting to a SVN repository

Open the project in which you want to add the access to SVN. A project is a set of `fragments`; there is always one `local fragment`, the one where you work. In this section a `remote fragment` will be added, the one that correspond to the SVN repository.

`Work models` are the fragments in read/write mode. That is, the fragment where the developer work. To add the possibility to work on the SVN model select the menu `Configuration > Work Models`.

In order to register the SVN repository as a remote work model, click the `Add` button in the `SVN models` section of the following window:

In the following form, you can choose a name for the fragment itself. The URI of the svn repository must be provided as well as the credentials for accessing it (if required). Checking the URI with the corresponding button is a good idea.

A new remote fragment is then available in your project as shown in the window below.

The screenshot shows the Javadoc for the `Class` metaclass in ScribesTools. The interface is organized into several sections:

- Navigation Bar:** Includes tabs for OVERVIEW, PACKAGE, **CLASS**, USE, TREE, DEPRECATED, INDEX, and HELP. Below these are links for PREVIOUS CLASS, NEXT CLASS, FRAMES, and NO FRAMES. A summary bar lists SUMMARY: NESTED | FIELD | CONSTR | METHOD and a detail bar lists DETAIL: FIELD | CONSTR | METHOD.
- Package List:** A sidebar on the left lists various packages and classes. The `Class` class is highlighted in the list.
- Class Information:**
  - Package:** `org.modelio.metamodel.uml.statik`
  - Interface Class**
  - All Superinterfaces:** `Classifier`, `Comparable<EObject>`, `Element`, `org.eclipse.emf.ecore.EObject`, `GeneralClass`, `EObject`, `ModelElement`, `ModelTree`, `Namespace`, `org.eclipse.emf.common.notify.Notifier`
  - All Known Subinterfaces:** `Component`, `ModuleComponent`
- Class Definition:**

```
public interface Class
extends GeneralClass
```
- Description:**

Class v0.0.9054 The Class is the main concept used in object-oriented modeling. It specifies which Instances can exist in an application. In Modelio, a Class is owned by a NameSpace (ModelTree) that can be a Package or a Class.
- Field Summary:**
  - Fields:** A table with two columns: **Modifier and Type** and **Field and Description**. It lists a static field `String MNAME`.
- Method Summary:**
  - Buttons for **All Methods**, **Instance Methods**, and **Abstract Methods**.
  - A table header for **Modifier and Type** and **Method and Description**.

Fig. 1.22: Javadoc of the “class” metaclass

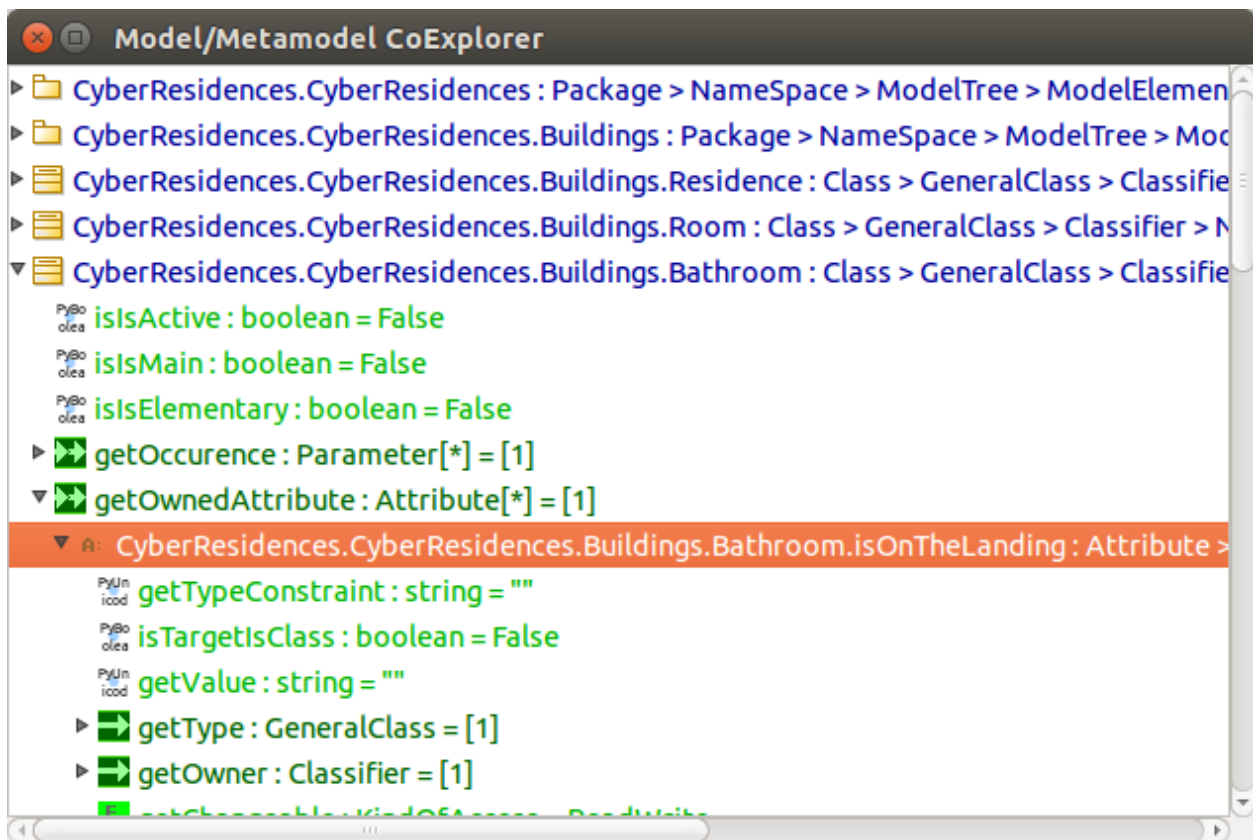
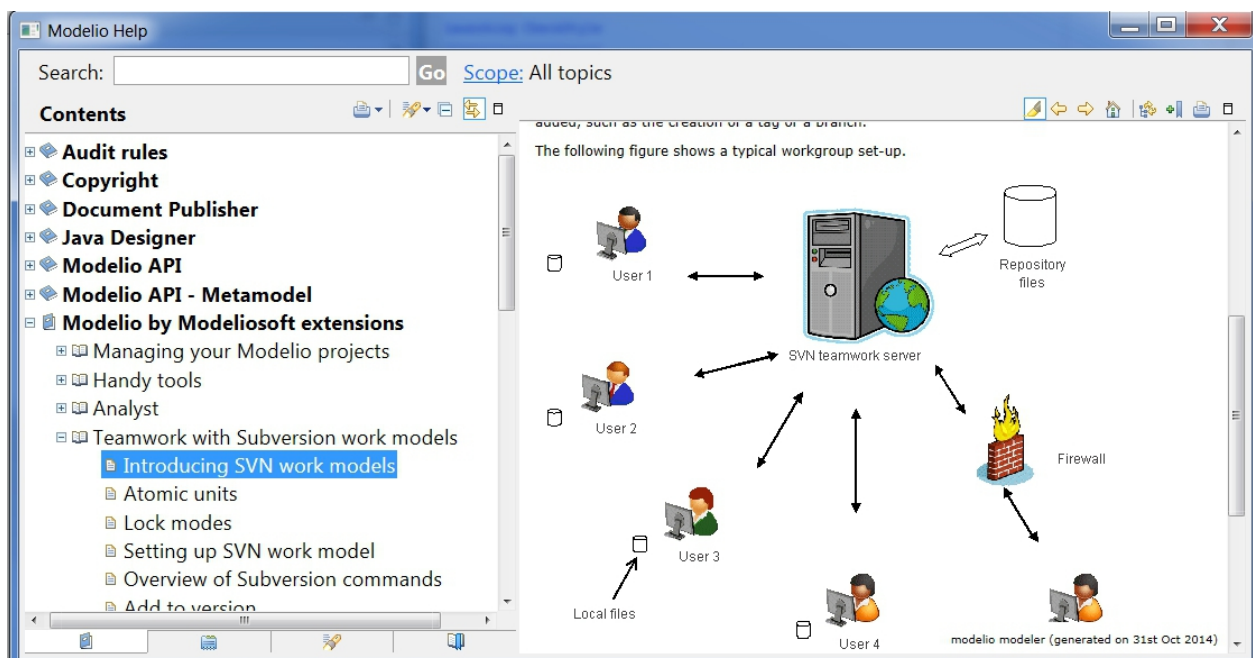
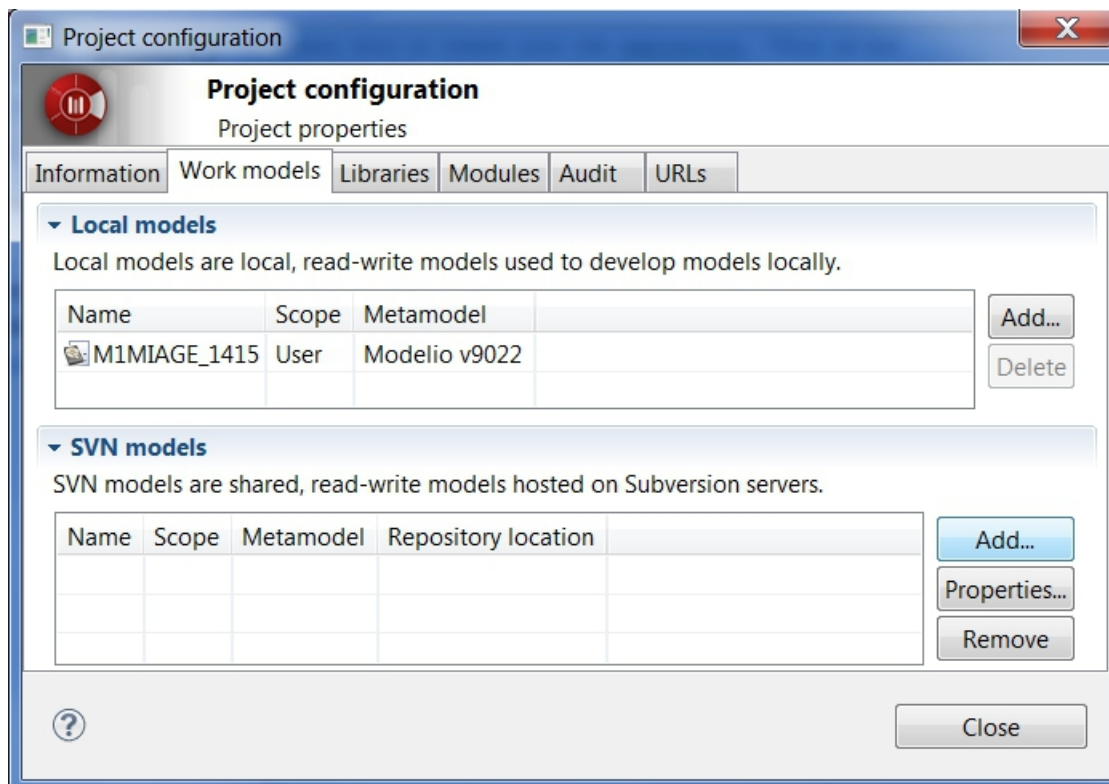
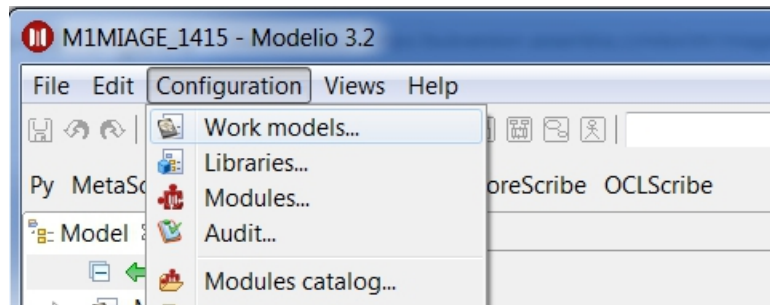


Fig. 1.23: A session with the CoExplorer plugin.





**Add SVN model**

**Add a SVN model to the project**  
Connect to a new SVN model.

Name:

Subversion connection parameters

Repository URI:

Authentication:

Type:

Login:

Password:

☒ Memorize login + password

Needs lock: ☐

Valid repository URI.

SVN models are shared, read-write models hosted on Subversion servers.

**Project configuration**

**Project properties**

Information | **Work models** | Libraries | Modules | Audit | URLs

**Local models**

Local models are local, read-write models used to develop models locally.

Name	Scope	Metamodel
M1MIAGE_1415	User	Modelio v9022

**SVN models**

SVN models are shared, read-write models hosted on Subversion servers.

Name	Scope	Metamodel	Repository location
G206	User	Modelio v9022	https://subversion.assembla.com/svn/m1miag

You should be able to use it and modifying it. The version control commands to use are mostly:

- update: to get the last updates from the central SVN repository,
- commit: to commit the local changes to the central SVN repository/

## UseOCL

“USE (OCL) is a system for the specification and validation of Information Systems based on a subset of the Unified Modeling Language (UML) and the Object Constraint Language (OCL).”

## Installation

Installing USE is rather easy:

- download the use zip file ([web](#)).
- extract this zip file where you want to install it (e.g. %SCRIBESTOOLS% on windows).
- rename the newly created directory to %SCRIBESTOOLS%\UseOCL.
- for convenience add the bin directory to your system PATH. For instance the PATH could look like:

```
%SCRIBESTOOLS%\UseOCL\bin; ...    # Windows. ; is the separator
/home/pablo/UseOCL/bin: ...      # Unix.    : is the separator
```

## Launching USE OCL

Once installed, you can just type the following command in a new shell:

```
use -nogui
```

If the *bin* directory is not in the PATH, then you have type the full path to use binary). This launch the USE OCL Command Line Interface (CLI).

**Warning:** On Windows, if you want to use cygwin, then type `use.bat -nogui` otherwise you may encounter problem with the use script.

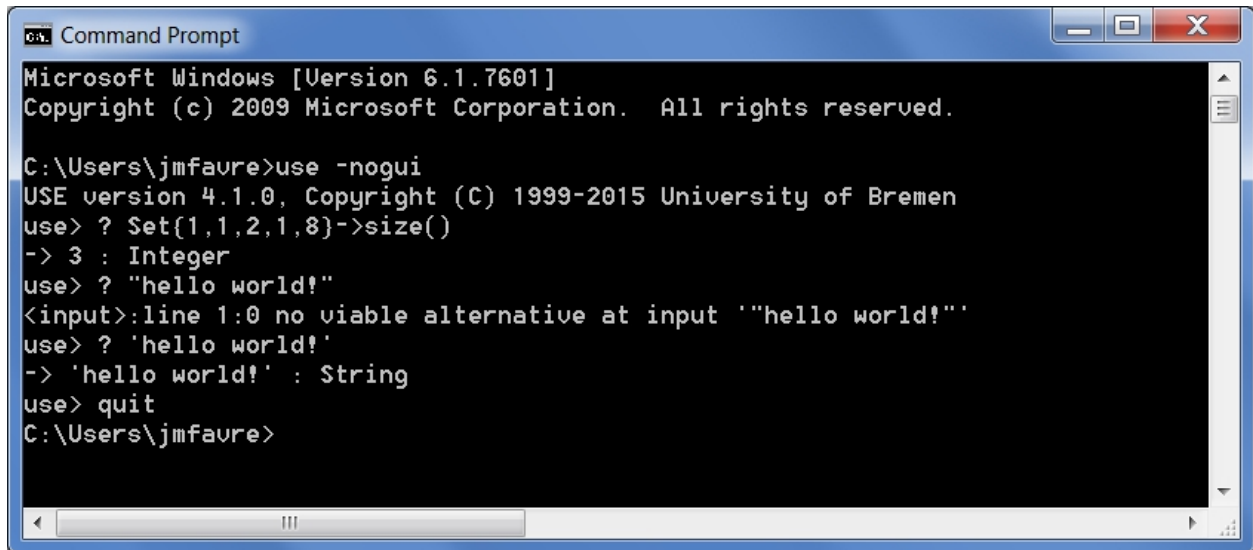
**Warning:** On Windows, you might see the following warning message that start with WARNING: Could not ... Software\JavaSoft\Prefs and ends with Windows RegCreateKeyEx(...) returned error code 5.. In this case in administrator mode one should create the key HKEY\_LOCAL\_MACHINE\Software\JavaSoft\Prefs with regedit. This is just a warning and you can use USE OCL safely interactively. This might although cause problems if you use USE OCL via some scripts as the command will always return an error code.

**Warning:** If you get the following warning when starting USE OCL then use the option `-nr`.  
`java.lang.UnsatisfiedLinkError: no natGNUReadline in java.library.path`

If you want you can also have a look at the graphical interface (in this case just type:

```
use
```





```
GA: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\jmfavre>use -nogui
USE version 4.1.0, Copyright (C) 1999-2015 University of Bremen
use> ? Set{1,1,2,1,8}->size()
-> 3 : Integer
use> ? "hello world!"
<input>:line 1:0 no viable alternative at input '"hello world!'"
use> ? 'hello world!'
-> 'hello world!' : String
use> quit
C:\Users\jmfavre>
```

Fig. 1.24: A session with the Command Line Interface. Use is started and then some OCL expression are evaluated with the ? command.

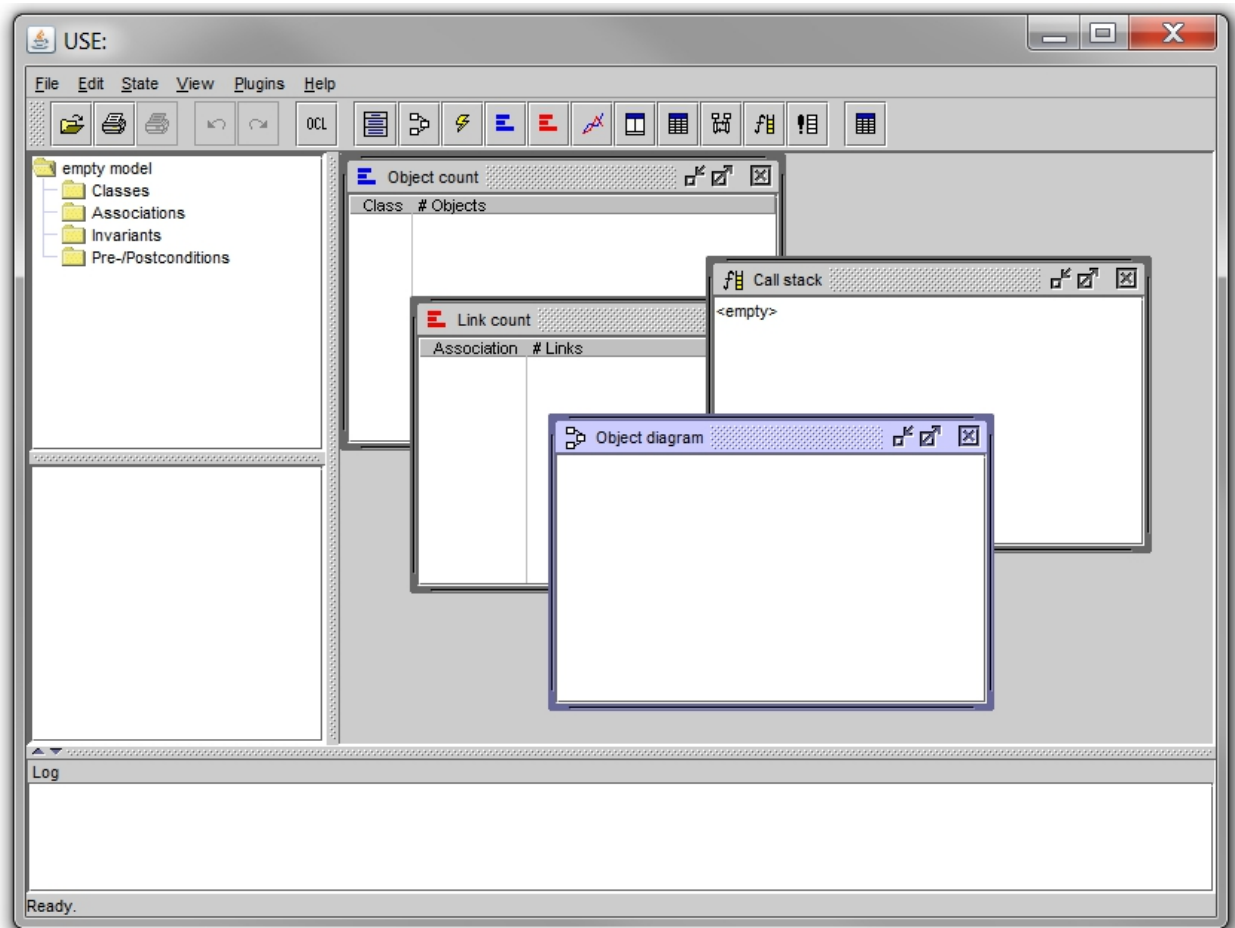


Fig. 1.25: The Graphical User Interface.

## Documentation

There is quite some documentation for USE OCL. We recommend the following order:

- 0. Two “cheat sheets” are provided in one file (local):
  - On one page, the OCL language is described according to the standard.

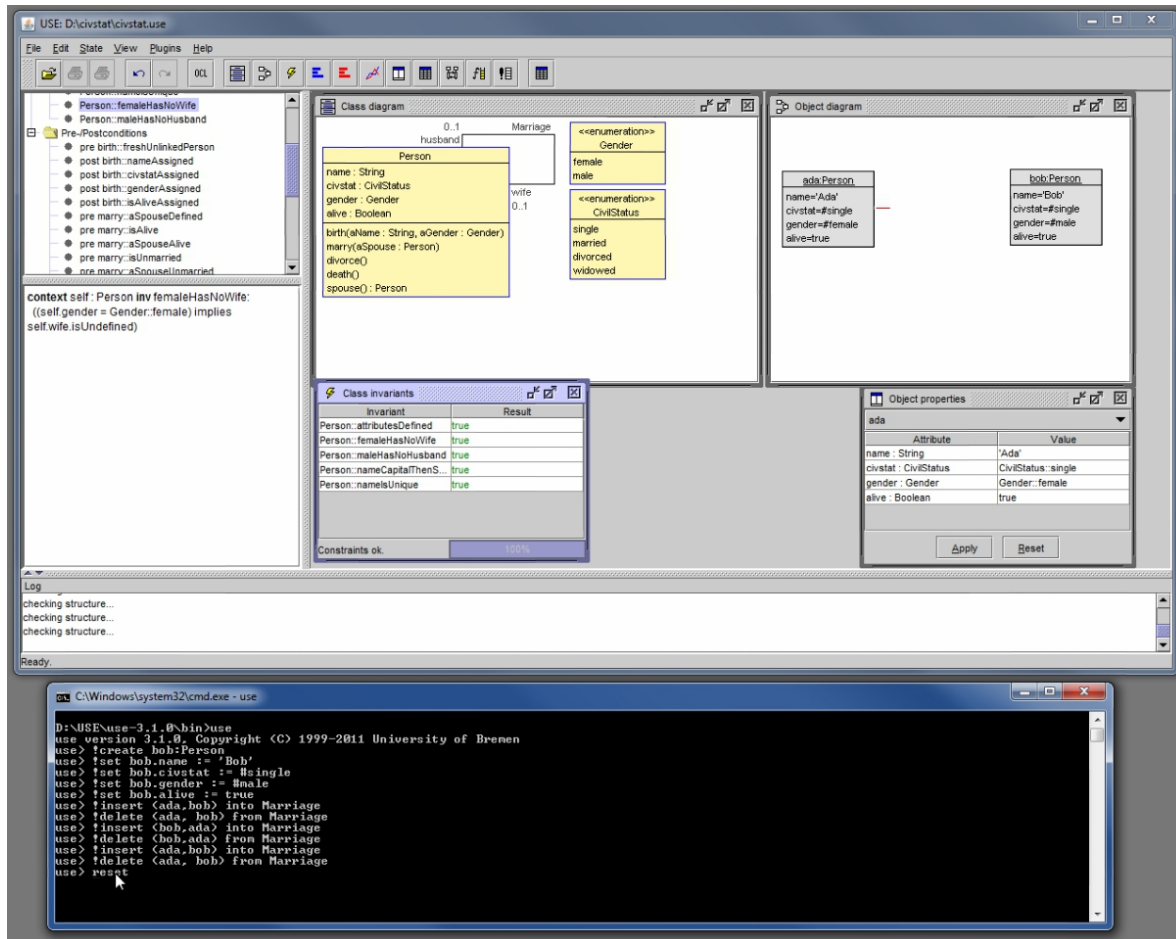
[illegible]

- On the other page, the languages of the USE environment are described: (1) the specification language (`.use`) which includes OCL, but also (2) the action language (`.soil`).





1. The “cheat sheets” just describe the language. To understand how USE can be used read first the quick tour ([local](#), [web](#)). A MUST READ. A simple class model with 3 classes, 3 associations and 4 OCL invariants.
2. Watch then the video ([local](#), [web](#)). The first part (until frame 128/208) deals with the static part and invariants. The second part deals with dynamics, operation simulation and pre/post conditions. You may be interested only by the first part or by both parts.



3. If *necessary* use the reference documentation (`local`, `web`). Use this document as a reference, for instance to check something about USE OCL language. Note that chapter 6 (page 82) contains the list of OCL operations supported by OCL USE.

**Note:** This documentation has not evolved for quite some time. More features have been added to USE OCL, in particular to follow the evolution of OCL/UML or to remove some limitations. For instance support for qualified associations is possible although undocumented. Use the cheat sheet for updated information. If needed the directory `examples` and/or `test` of the distribution contains also some examples demonstrating the use of these features. For instance various qualified association examples are visible in the `test/t086.use`.

**Attention:** The documentation and the video show both the Command Line Interface (CLI) and the Graphical User Interface (GUI). You might be however interested only by the CLI (in particular in the context of automation and integration with other tools). In this case, use the following option when launching USE OCL (don't forget `-nr` if you get a warning):

```
use -nogui
```

## Syntax

The syntax of OCL and USE/SOIL languages is summarized in (`local`). Only particular points are exemplified below.

## Enumerations

USE:

```
enum Season {winter, autumn, spring, summer}
```

SOIL/USE:

```
Season::winter
```

## Classes

Classes can be defined as following:

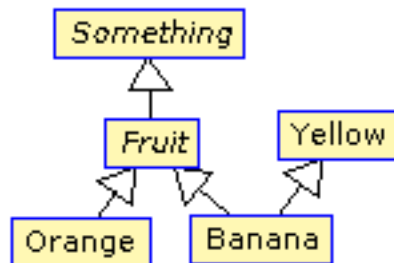
```
class Yellow
end

abstract class Something
end

abstract class Fruit < Something
end

class Banana < Fruit, Yellow
end
```

UML:



## Attributes

```
class Banana
attributes
  length : Integer /* Integer, Real, Boolean, String */
  growthTime : Season
  -- Tuple, Bag, Set, OrderedSet, Sequence
  goodies : OrderedSet (Bag (Sequence (Set (Tuple (x:Integer,y:Real,z:String))))))

  remainingDays : Integer
    init: 42 -- attribute initialization
    derived: self.length * self.size -- attribute derivation
end
```

Restriction form the standard \* No invariants directly declared on attributes \* No cardinality supported for attributes.  
For instance smoker : Boolean[0..1] is not supported.

## Operations

```

class Banana
operations

  sleep()                                -- operation signature

  wakeUp(n : Integer):String             -- operation specification
    pre notTooMuch: n > 10 and n < self.length    -- precondition
    post resultOK: result > 'anaconda'           -- postcondition

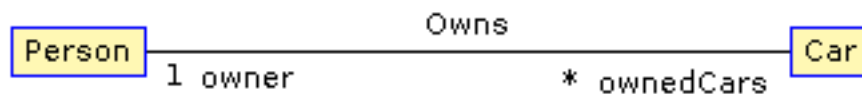
  helloJungle() : String                 -- operation implementation (SOIL)
    begin
      declare x : Banana ;
      WriteLine('hello') ;
      x := new Banana ;
      self.length := self.length + self.remainingDays*20+3 ;
      result := 'jungle' ;
      destroy x ;
    end
    pre freshEnough: self.remainingDays > 10

  smash() : Integer                      -- operation/query (OCL)
    = self.length + Set{4,2}->size*42
end

```

## Associations

UML:



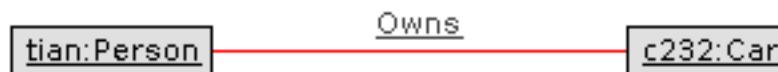
USE:

```

association Owns between
  Person [1] role owner
  Car[*] role ownedCars
                                -- more roles for n-ary association
end

```

An example of link:

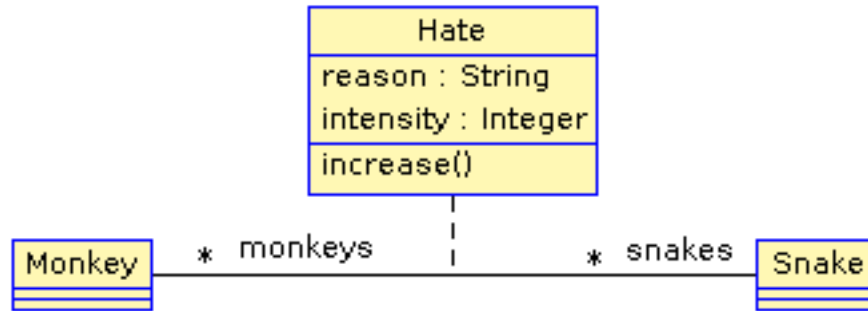


SOIL:

```
! insert(tian,c232) into Owns
```

## Association Classes

UML:



USE:

```

associationclass Hate between
    Monkey [*] role monkeys
    Snake [*] role snakes
attributes
    reason : String
    intensity : Integer
operations
    increase()
end
  
```

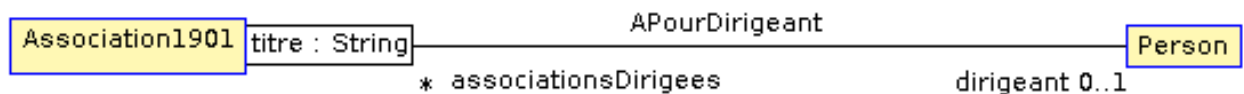
SOIL:

```

! c := new Hate between (chita,kaa)
! c.reason := "kaa is really mean"
! c.intensity = 1000
  
```

## Qualified Associations

Let us consider this following qualified association.

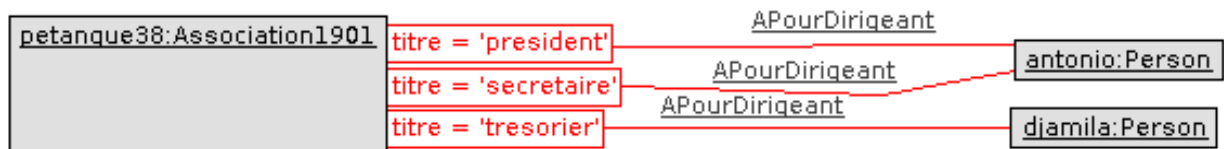


Note the qualifier construct in the USE association definition:

```

association APourDirigeant between
    Association1901[*] role associationsDirigees qualifier(titre:String)
    Person [0..1] role dirigeant
end
  
```

The example below represents a valid object model.



Note the insert (`<role>{<value>...}`) SOIL construct to create new link and `.<role>[<value>]` to traverse links:

```

! insert (petanque38{'president'},antonio) into APourDirigeant
! insert (petanque38{'secretaire'},antonio) into APourDirigeant
  
```

```
! insert(petanque38{'tresorier'},djamila) into APourDirigeant
? petanque38.dirigeant['president']
? petanque38.dirigeant
```

## Creating diagrams

USE Graphical User Interface (GUI) can be used to create class diagrams as well as object diagrams (among other kind of diagrams).

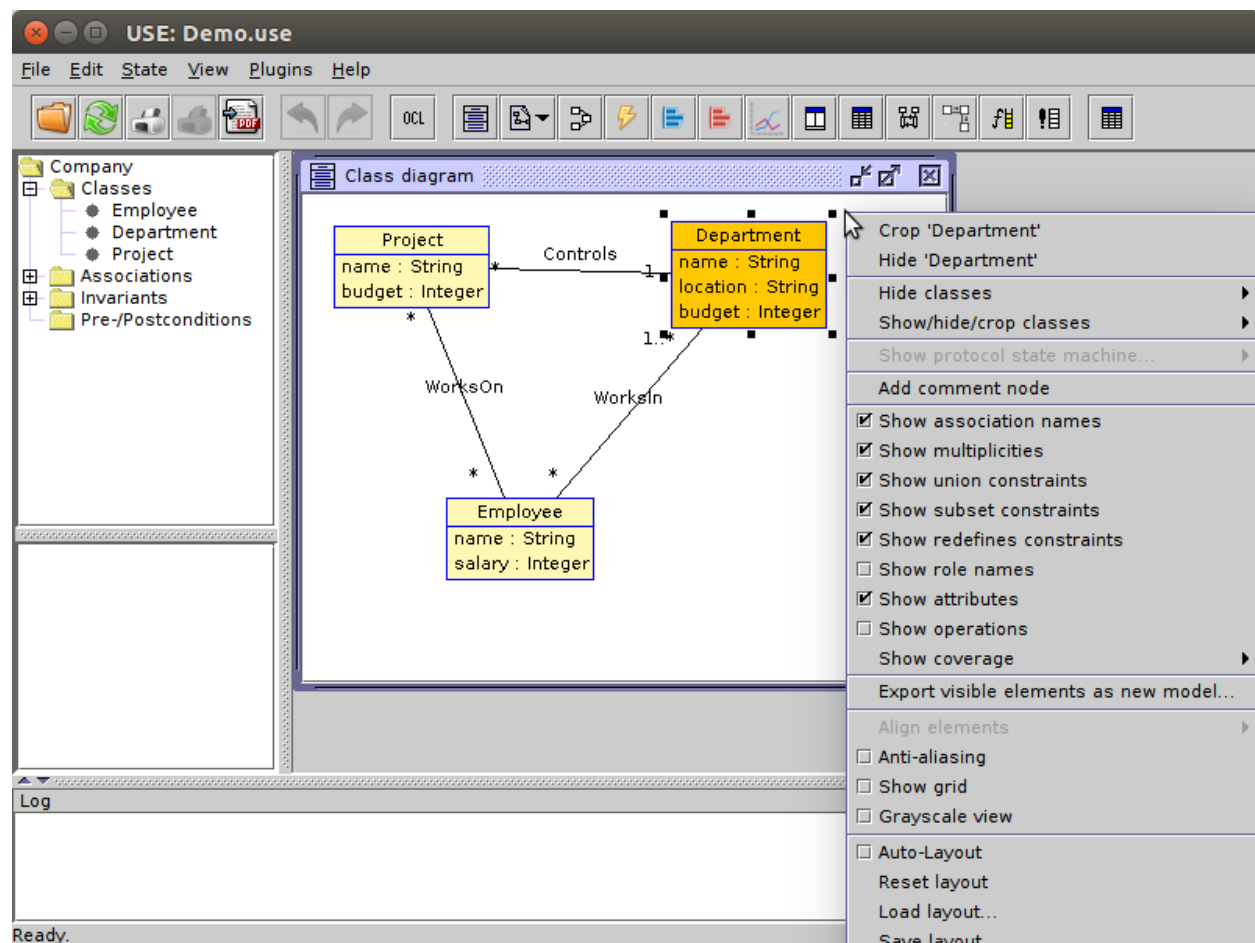
In order to do so launch use *without* the `-nogui` option. If you want to get both a class diagram and a object diagram specify both a `.use` and `.soil` file on the command line. Something like:

```
use -nr CyberGarage.use scenarios/CyberGarage/scenario.use
```

To create a diagram then use the menus:

- View > Create View > Class Diagram and
- View > Create View > Object Diagram

Many useful options are then available in the contextual menu of each diagram (right click).



Many options are available. You are likely to use:

- Auto-Layout to get a first layout automatically. Don't apply this option on an existing layout ...

- `Save-Layout` to save the layout after some manual arrangement. This save the diagram layout in the form of a `.clt` (Class Layout) file or `.olt` (Object Layout) file depending on the diagram. Quitting use without saving the current layout will create the file `'...-default.clt'`. It is a bad practice to use this file, especially when various diagrams are to be build. Save the layout in proper files with proper names.
- `Show multiplicities`, `Show role names`, etc.

If you intend to create various diagrams for the same model (to create different views) you are likely to use `Hide` options.

---

**Note:** Naming objects

If you have trouble in getting what you want as object identifier in the object diagram (you might get for instance `Vehicule1 : Vehicule`) this is mostly due because no name have been assigned to yours objects. In order to do so you have to use the following syntax:

```
v803 := new Vehicule('v803')
```

Note that the first occurrence of `v803` is a variable name (not display in the diagram), while the second occurrence is the object identifier displayed in the diagram. If none is given, use will define one automatically.

---

## Using PyCharm IDE

USE specifications are just plain text files. A regular file editor and a shell are just enough to work with USE.

You may however want to use syntax highlighting with PyCharm (see [PyCharm](#) for installation instructions). Read also [Launching PyCharm](#). When you launch PyCharm select the “project directory”, the top directory that contains all your files including the `.git` subdirectory if you use git.

PyCharm has to be configured and this imply to follow *various* steps. A PyCharm plugin has to be installed and a few configurations files have to be copied. At the end you will get the following result.

The figure shows the following elements:

- On the left a `.use` specification is “syntax-highlighted”. This is handy especially if the OCL language is used as they are many keywords and operations.
- On the right a `.soil` scenario is “syntax-highlighted”. This is handy since there is typically a lot of comments in such a scenario; the statments are much more visible in this way.
- On the top bar, a button allows to check the scenario against the specification. There is no magic here, this button should be configured.
- On the bottom window, the output of USE is displayed with colors for errors. This is quite handy when the output is large.

The instructions below will allow you to get an environment as shown in the following picture. If you are just going to se USE OCL only once don’t waste your time. Use a regular editor. Otherwise your might consider following the procedure below.

### USE and SOIL highlighting

PyCharm support syntax highlighting for many languages but not USE OCL. The file contains the definition of the language (keywords, comments, etc.).

1. PyCharm should be stopped.
2. Download (`UseOCL.xml`).

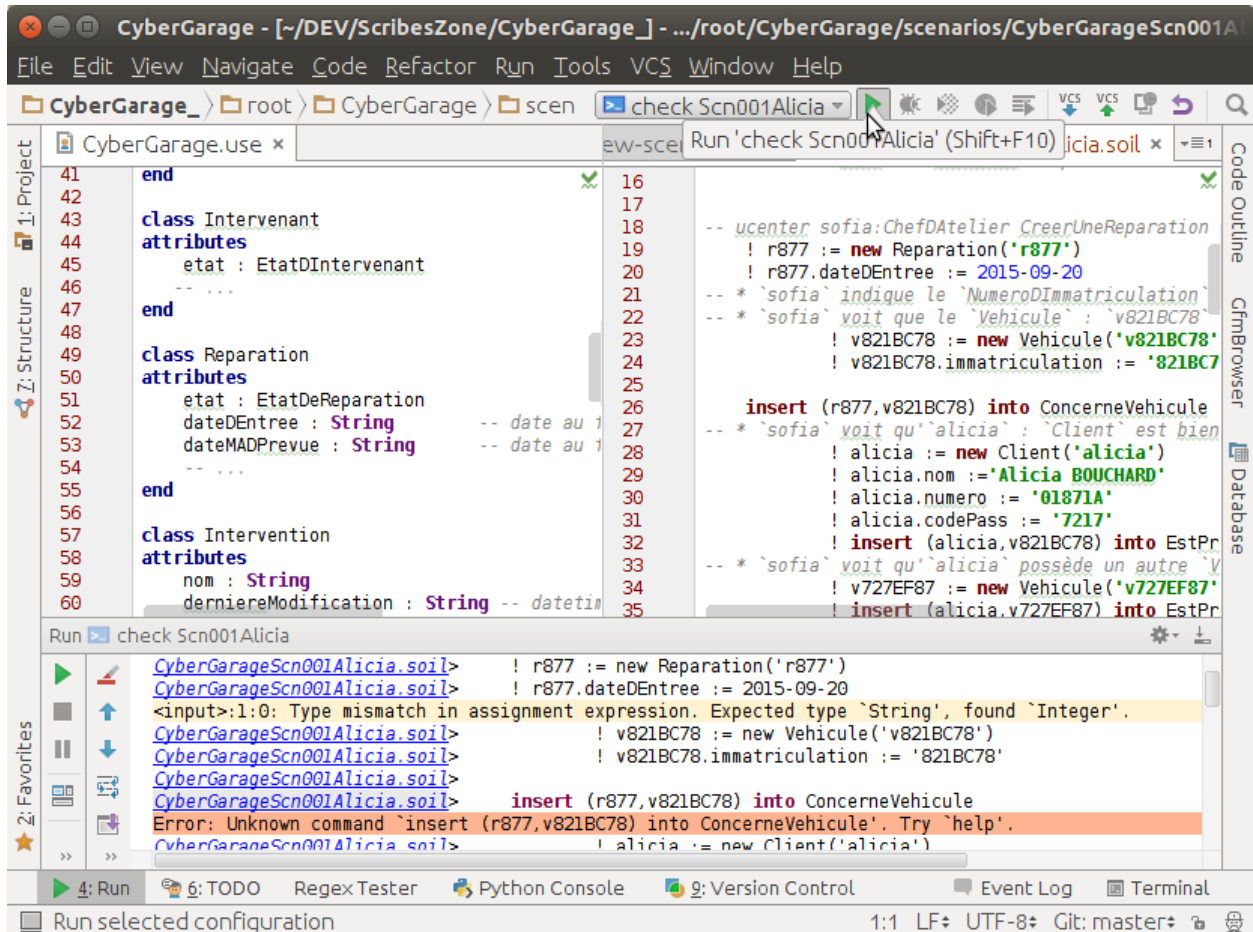


Fig. 1.26: Using PyCharm for USE syntax and output highlighting



3. Copy this file to `.PyCharm50/config/filetypes` (create the directory `filetypes` if it does not exist already).

---

**Note:** The `.PyCharm50` directory is used for global IDE settings. The number (e.g. 50) vary according to the version of the product. This directory it is usually located in your home directory (not on OS X). See [IDE Settings](#) for more information.

---

Start PyCharm. From now on, all `.use`, `.soil` and `.con` files should be colored. If you are curious, the (`UseOCL.xml`) file has been produced using PyCharm feature to define [new file types](#).

## Output highlighting

To get the **output** of USE OCL colored (to see the errors as shown in the figure above) three steps should be followed:

1. Installing the Grep Console plugin
2. Installing a configuration suitable for USE OCL
3. Creating a “Run Configuration”

## Installing Grep Console

In PyCharm go to File (menu) > Settings (menu) > Plugins (tab) > Browse Repositories (button).

---

**Note:** If you computer is behind a firewall you have to specify a proxy. In this case select HTTP Proxy Settings > Manual Proxy Configuration and fill the parameters. For instance at the UGA you will need to enter: HostName : `www-cache.ujf-grenoble.fr`, Port Number : 3128

---

A list of plugins should be displayed. Type "Grep" in the search field and install Grep Console.

## Configuring Grep Console

The Grep Console plugin allows to associate colors to regular expressions matching program outputs. Download the (`GrepConsole.xml`) file which defines a configuration suitable for USE outputs. PyCharm should be stopped. Copy this file into the `.PyCharm50/config/options` directory (see above). Override the existing file with the same name.

## Creating a Run Configuration

Environments like PyCharm use Run Configuration to launch repetitive tasks. PyCharm should be stopped. Copy the (`checkScn001Razamanaz.xml`) configuration in the directory `.idea/runConfigurations` of your project (create the directory `runConfigurations` if it does not exist already). After starting PyCharm and selecting the menu Run > Edit the Configuration you should see the following configuration:

Adjust this configuration where necessary.

At the time of writing this configuration refers to the ScribesInfra repository which must be clone at the same level.

---

**Note:** As an alternative to PyCharm, on Windows, you can use notepad++. A syntax file is available for OCL sources although it is not updated, do not support `.soil` files nor output highlighting.

To install this file:

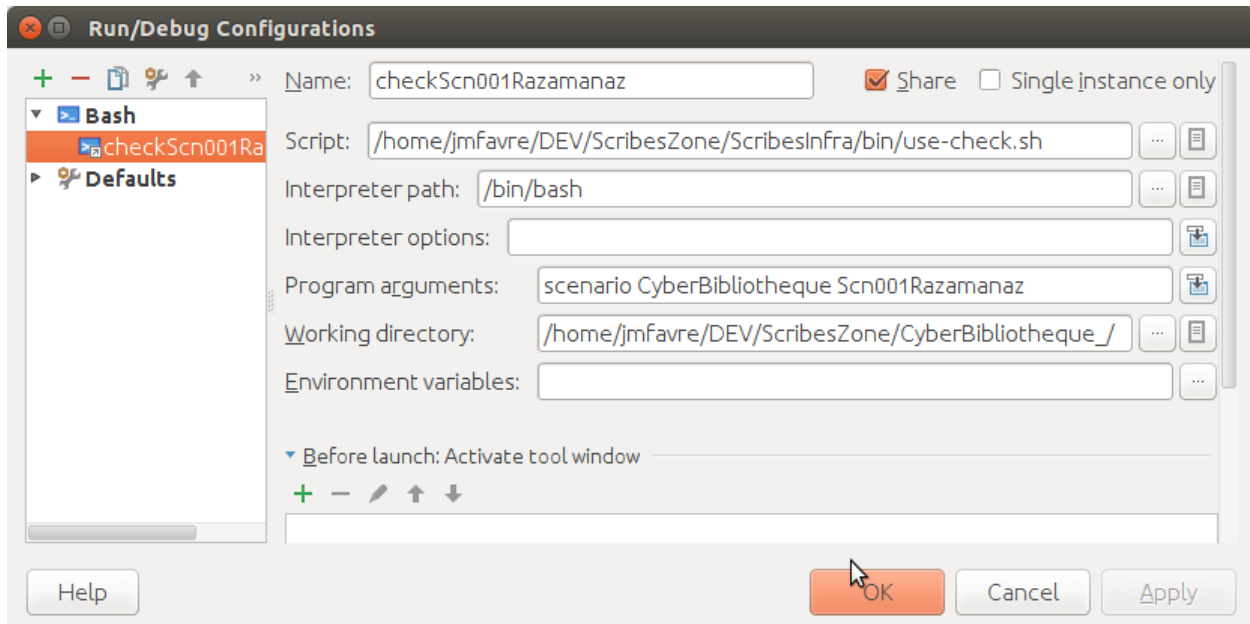


Fig. 1.27: The checkScn001Razamanaz configuration parameters

- in Notepad++ go to “*Main menu > Language > Define your language... > Import ...*”
- select the file `USE_Notepad_plusplus_User_Defined_Language.xml` [web](#).
- You may have to restart notepad++.

---

## Examples

Various examples of use specifications are available in the distribution [web](#) in particular in the directory `examples`. The file `README.examples(local)` provides an interesting index that show which OCL features are used in which files.

## KMADe

**KMADe** (*Kernel of Model for Activity Description environment*) is an editor of *task models* along with a simulator. As this tools is written in java it runs on all platforms.

## Features

KMADe is an editor of task models.

## Interoperability

KMADe represents task models as `kxml` files.

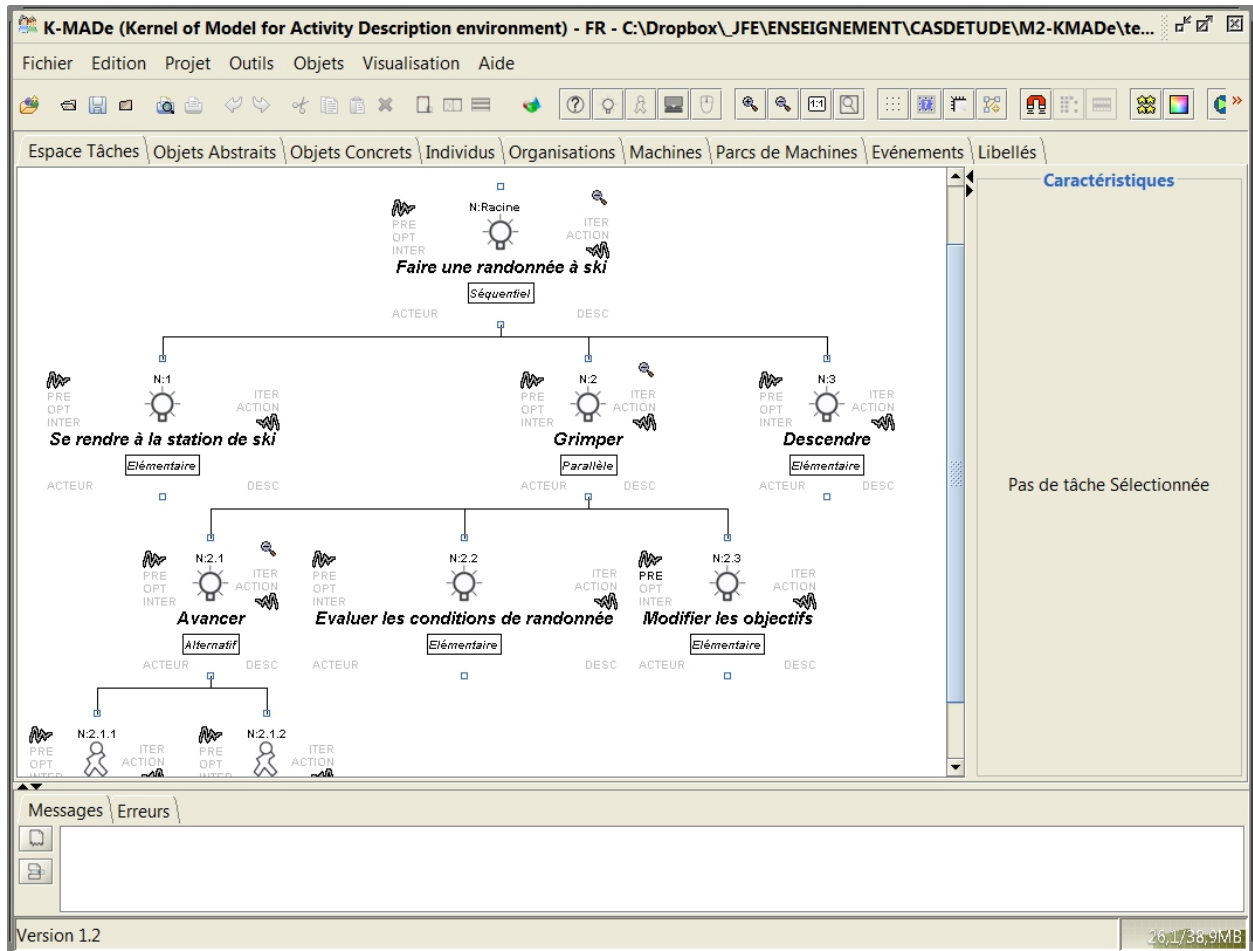


Fig. 1.28: task model by Sybille Caffiau

## Installation

KMADe is easy to install:

- Download the KMADe zip archive [web](#).
- Move this file to the directory %SCRIBESTOOLS%\.
- Unzip the file here.
- Rename the created directory to KMADe
- Remove the zip file if you want.

## Launching KMADe

Once installed, you can just have to double click on the `kmade.jar` file in the installation direction. KMADe should start as this is a java executable.

## Documentation

KMADe has an extensive documentation ([local](#), [web](#)) but it is mostly in french.

## Pandoc

[Pandoc](#) is a converter that allows to transform documents across a very wide range of documentation formats including html, markdown, reStructuredText, textile, DocBook, LaTeX, MediaWiki, Word docx, EPUB, etc.

## Installation

See the [installation page](#) for detailed information. Install the application in a directory like %SCRIBESTOOLS%\Pandoc.

On Windows:

- download `pandoc-1.15-windows.msi` [web](#).
- if you are on a personal PC just click on the executable. If the application should be installed on multiple users type instead:

```
msiexec /i C:\DOWNLOADS\pandoc-1.15-windows.msi ALLUSERS=1 APPLICATIONFOLDER=%SCRIBESTOOLS%\Pandoc
```

On Ubuntu:

```
sudo apt-get install pandoc
```

## Launching Pandoc

To test your installation type in a new shell:

```
pandoc --version
```

## Documentation

There is a lot of documentation online. The [user guide](#) describes command lines options. ..  
.....

## SchemaSpy

[SchemaSpy](#) is a simple reverse engineering tool that takes a SQL database and generate diagrams of the database schema. This is a java program; it runs on all platforms.

## Features

[SchemaSpy](#) reads a relational database and

- generates a XML file corresponding to the database schema
- detects a few kind of defaults in the database design

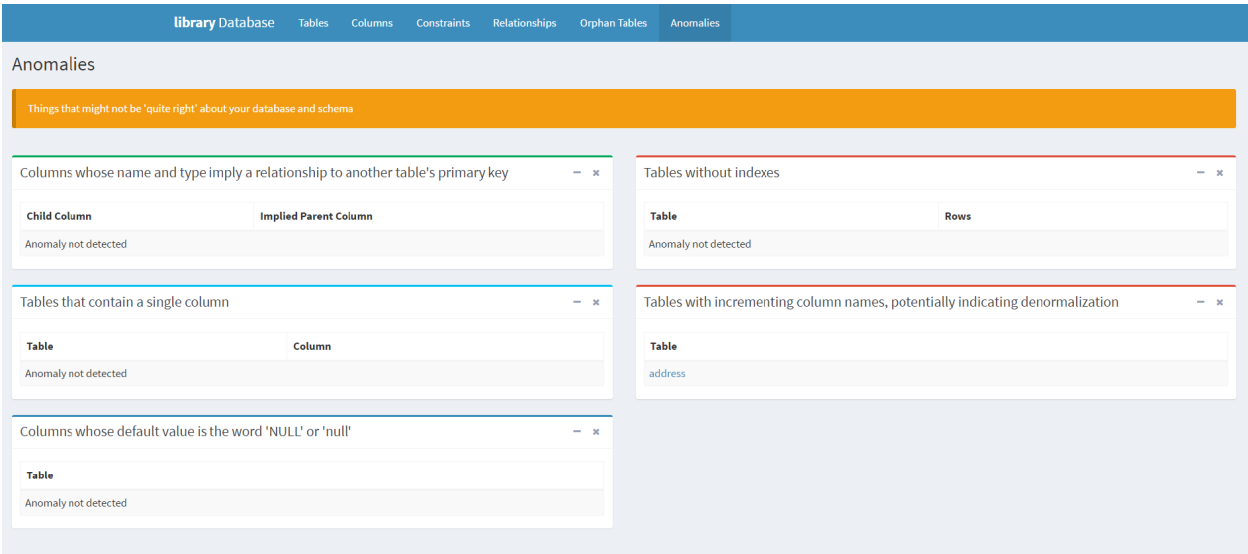


Fig. 1.29: Examples of defaults detected by [SchemaSpy](#)

- generates a html site a la javadoc with clickable entity-relationship diagrams.

## Interoperability

[SchemaSpy](#) is particularly interesting because the reverse engineering process generates a rather simple XML file so that other tools can be build by consuming this file.

## Installation

To install [SchemaSpy](#):

- Download the [SchemaSpy](#) jar file [web](#).

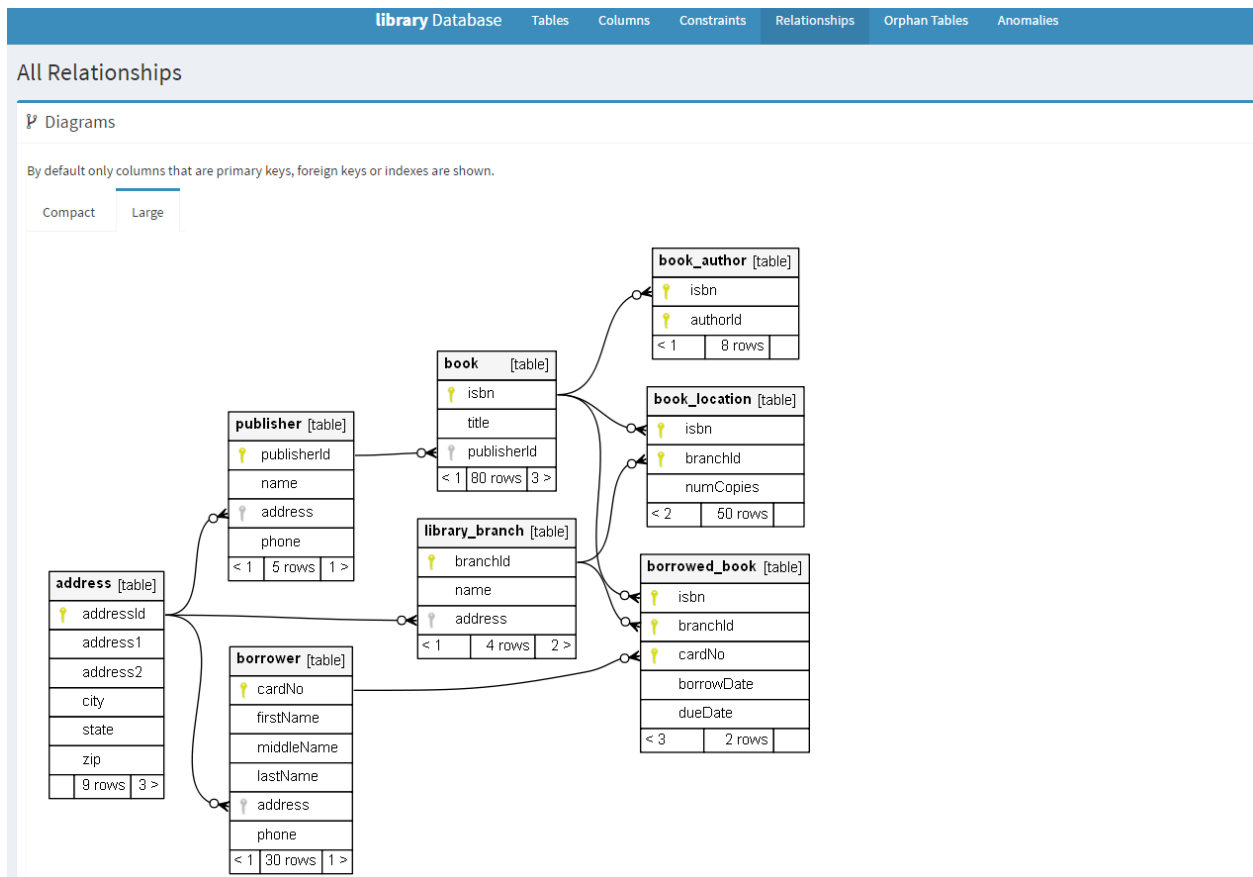


Fig. 1.30: Example of diagram generated by SchemaSpy

- Create the directory %SCRIBESTOOLS%\SchemaSpy.
- Move the `schemaspy-6.0.0-rc1.jar` file into %SCRIBESTOOLS%\SchemaSpy\schemaSpy.jar.
- Add %SCRIBESTOOLS%\SchemaSpy to the system PATH.
- Copy the files `schemaspy.bat` and `schemaspy.sh` into the directory

## Launching SchemaSpy

You can check if the installation (in a new shell) by trying the `-dbhelp` option:

```
schemaspy -dbhelp
```

This command should display a rather long list of supported database types, with the option to use to connect to the database in each case.

You need an access to a SQL database if you want to execute SchemaSpy. The command line options are described in the documentation.

## Documentation

The documentation is available on the web. It mostly describes the many possible command line options of [SchemaSpy](#). There is also an example of generated html, “[library example](#)”.

## SchemaSpy and SQLite

The SQLite driver delivered with SchemaSpy is not working. If you want to use SQLite, a driver and a property file must be installed:

- make sure that the driver have been installed (see the [Installation](#) section of [SQLite](#)). This results in the driver %SCRIBESTOOLS%\SQLite\sqlite-jdbc.jar.
- Copy the file `sqlite.properties` into the directory %SCRIBESTOOLS%\SchemaSpy.
- Adjust the location of the driver in the `sqlite.properties` file.

To use a SQLite database file you will have to specify the path to the property files with the `-t` option: For instance the following command line generate the documentation of the database `db.sqlite3` in the `docs\SchemaSpy` directory:

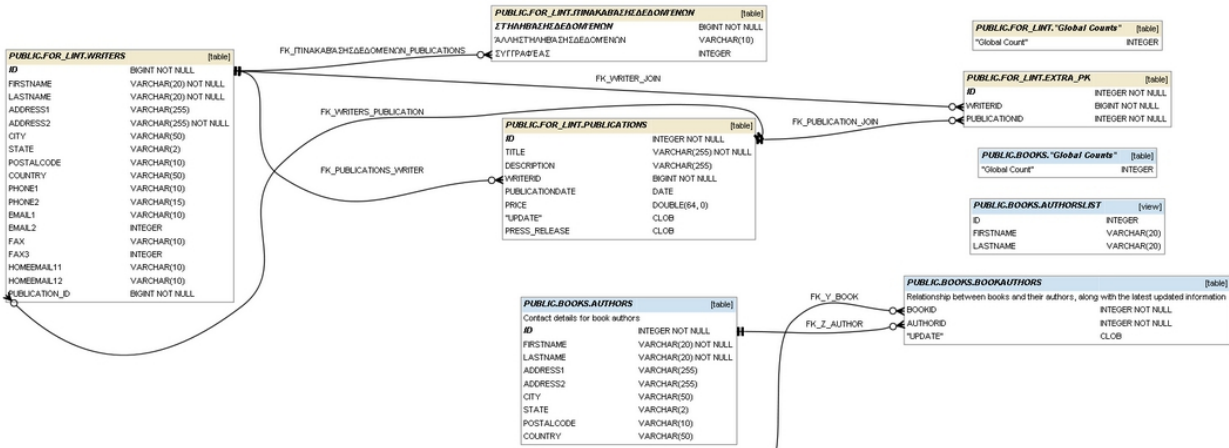
```
schemaspy -t %SCRIBESTOOLS%/SchemaSpy/sqlite.properties -db db.sqlite3 -sso -o docs\SchemaSpy
```

With oracle:

```
schemaspy -t orathin -host im2ag-oracle.e.ujf-grenoble.fr -port 1521 -db ufrima -s MYLOGIN -u mylogin  
-p * -o out
```

## SchemaCrawler

[SchemaCrawler](#) is a tool and an API dealing with SQL database schemas information.



## Features

The tool provides:

- extracted information about database schemas in various formats (e.g. text, json, csv, html5)
- documentation generation with ER diagrams generated by graphviz,
- lint like quality insurance,
- grep/diff like support.

The API allows to access to SQL databases in an homogeneous way.

## Installation

To install [SchemaCrawler](#):

- Create a directory that will contain the software (e.g. %SCRIBESTOOLS%\SchemaCrawler).
- Download [schemacrawler-12.06.03-main.zip](#) [web](#).
- Extract the content of the archive in the target directory.
- As a result you will 3 sub directories `_ivy`, `_schemacrawler`, `examples`.
- Add the directory to the system PATH

The `sc` scripts provided (“sc” stands for SchemaCrawler) assume that this command will run in the `_schemacrawler` directory. In order to remove this constraints replaces the content of `_schemacrawler\sc.cmd` by:

```
@SET DIR=%~dp0
@java -classpath %DIR%/lib/*;%SC% schemacrawler.Main %*
```

On unix replace the content of `_schemacrawler\sc.sh` by:

```
#!/bin/sh
SC=`dirname $0`
java -cp $(echo $SC/lib/*.jar | tr ' ' ':') schemacrawler.Main $*
```

[SchemaCrawler](#) is written in java but supports scripting with different languages (). In order to use Python for Scripting the following commands should be executed (assuming that the directory is %SCRIBESTOOLS%\SchemaCrawler):



```
cd %SCRIBESTOOLS%\SchemaCrawler\_ivy
download.cmd python      # use download.sh on unix
```

## Launching SchemaCrawler

The simplest test to installation is to type the following:

```
cd %SCRIBESTOOLS%\SchemaCrawler\_schemacrawler      # to be adapted if necessary
schemacrawler --version
schemacrawler --help
```

This should display the version of [SchemaCrawler](#) and then the help of the command line.

To go further the directory `examples` contains many examples of usage. A HSQLDB database server is provided for testing purposes. To launch this example database server open a shell window and type the following commands:

```
cd %SCRIBESTOOLS%\SchemaCrawler\examples      # directory to be adapted if necessary
StartDatabase.cmd                             # .sh on unix
```

At that level the test server should be running in this window (and the window color may have changed).

Open another shell and try:

```
cd %SCRIBESTOOLS%\SchemaCrawler\_schemacrawler      # to be adapted
sc.cmd -server=hsqldb -database=schemacrawler -password= -infolevel=detail -command=schema
schemacrawler -server=hsqldb -database=schemacrawler -password= -infolevel=detail -command=schema
```

If you have a sqlite database you can also try:

```
schemacrawler -server sqlite -database <path to the sqlite file>\db.sqlite3 -infolevel=maximum -password=
```

## Documentation

The documentation of the command line interface is available via the command:

```
schemacrawler -help
```

The detail of the API is documented through [SchemaCrawler javadocs](#).

## Examples

The directory `examples` of the installation contains examples of use for the command line interface, for scripting or for the API.

## CheckStyle

[Checkstyle](#) is a [quality control](#) tool allowing to check adherence of java programs to coding standards. [Checkstyle](#) is written in java and runs on all platforms.

## Features

[Checkstyle](#) comes by default with configurations files supporting Sun Code Conventions and Google Java Style but nearly everything can be configured. New checks can be written in java classes so nearly all kind of coding standards can virtually be implemented with [Checkstyle](#).

## Interoperability

[Checkstyle](#) is already integrated in many different IDE in the form of plugins for instance. There is a large list of [tools related to Checkstyle](#).

## Installation

On Ubuntu you can install [CheckStyle](#) has following:

```
sudo apt-get install checkstyle
```

Installing [CheckStyle](#) as a standalone tool is easy:

- Create a directory `%SCRIBESTOOLS%\CheckStyle`.
- Download `checkstyle-X.Y-all.jar` [web](#).
- Copy the jar file *into* `%SCRIBESTOOLS%\CheckStyle\checkstyle-all.jar` (remove the version number).
- Copy the files `checkstyle.cmd` and `checkstyle.sh` into the directory.
- Add the `%SCRIBESTOOLS%\CheckStyle` directory to the path.

## Launching CheckStyle

You check the installation as following. The option `-h` and `-v` display the [CheckStyle](#) help and version respectively:

```
checkstyle -h
checkstyle -v
```

If you have a java program at hand (let's say `MyProg.java`) and you want to use Sun Code Conventions on it type:

```
checkstyle -c /sun_checks.xml MyProg.java
```

## Documentation

A lot of documentation is available online [web](#).

## PyLint

[PyLint](#) is a quality control tool allowing to check adherence of python programs to coding standards. [PyLint](#) is written in Python and runs on all platforms.

## Installation

Installation instructions ([web](#)) depends on the platform used but basically [PyLint](#) is installed via python `pip` or platform specific package installers (e.g. `apt-get` on debian or ubuntu). With `pip` just type:

```
pip install pylint
```

**Attention:** On windows, make sure that the `Scripts` directory of your python installation is included in the system path (see [Changing the PATH](#)). Some other options are discussed [here](#)

On Ubuntu you can install `pylint` as following:

```
sudo apt-get install pylint
```

## Lauching PyLint

To try the installation type:

```
pylint --version
pylint --help
```

## Documentation

The documentation is available on <http://docs.pylint.org/>

## Diigo

[Diigo](#) is a collaborative bookmarking and annotation service on the cloud.

## Features

[Diigo](#) allows to:

- manage bookmarks (and store them in the cloud, not on your computers)
- create notes and create libraries of images from the web
- annotate any web pages and also pdf from the web
- create groups and share information (bookmarks, notes, annotations) among these groups
- generate rss feeds that are updated every time you publish a new link

## Installation

You have fist to [register to Diigo](#) and select the free plan. This will give you access to the web interface of [Diigo](#).

To take the full profit of [Diigo](#) (and in particular to add annotations) you should install a plugin for you favorite browser. You can either search the web for with google `diigo plugin chrome` for instance or use the [Diigo tools](#) page (use the Extension/Web Apps). [Diigo](#) is also available on android and iphone.

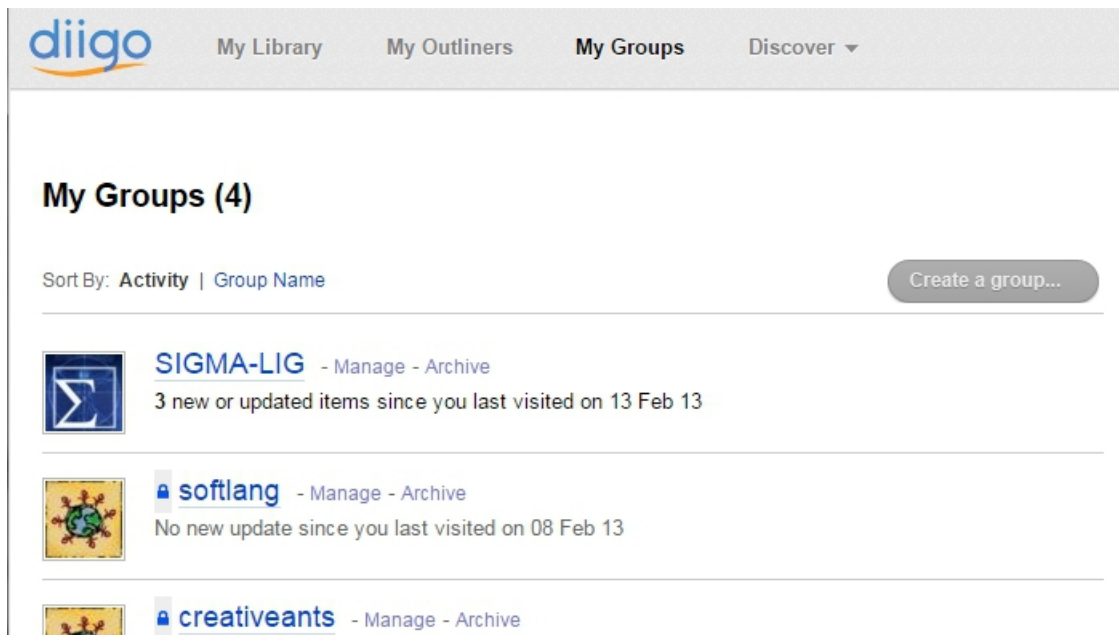


Fig. 1.31: Diigo web interface for managing groups

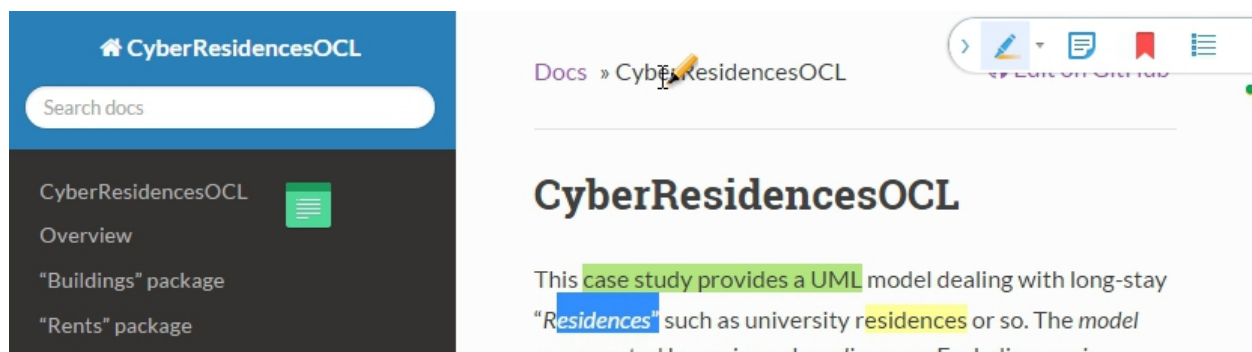
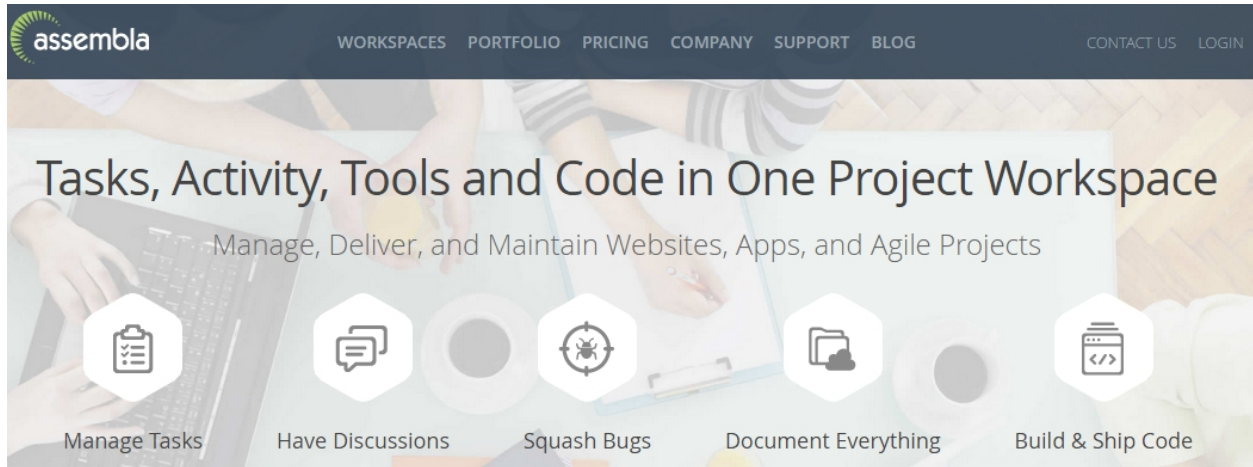


Fig. 1.32: "Annotate &amp; Capture the Web" in action, the plugin for Chrome

## Assembla

Assembla is a hosting service in the cloud that support in particular SVN and git repositories. In practice this is a good option to host SVN repositories freely. Much more feature are available with paid plans.



### Creating an account

To get a free account use 'Pricing' and then click on the link Sign Up for a Free Plan that is on the page as shown below:

**START MY 15 DAY FREE TRIAL**

## Free SVN, Git & P4\* Repositories

1 Space, Unlimited Users, Unlimited Repositories, 1 GB Storage - [Sign Up for a Free Plan](#)

\*Each Perforce repository includes 17 users, up to 20 depots and 1000 files.

You can sign up with a google account. Otherwise you will just need to a few information: username, password, email, first name, last name.


Select add a SVN repository if this is what you want and give it a name. That's all.


### Adding collaborators to a repository


To add collaborators to the repository click on the repository and then select the Team tab. You just have to enter the email of each collaborator and select their status for the selected repository:

- member for read/write access

## Free Unlimited SVN, Git and P4 Repositories

 Username

 Password


 Email Address

Already have an Account? [Log in](#)

**CREATE ACCOUNT**

By clicking **Create Account** you agree to the [Terms of Service](#)

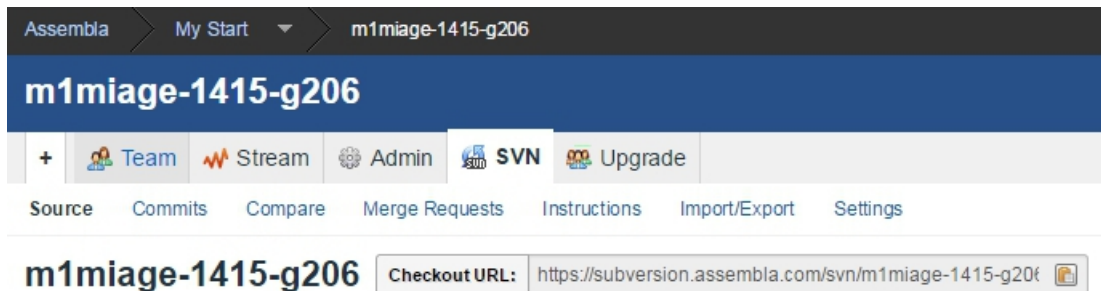
[Cancel](#)

 **Sign up with Google Account**

- `watcher` for read access

### Using a SVN repository

SVN repositories are used remotely via SVN client. In order to do this you must know the URI of the repository and give this URI to the client. The URI can be found on the repository page in the SVN tab in the Checkout URL field as shown below.



### Git(Hub)

Git is one of the most trendy Distributed Versioning Control System (DVCS) especially in the context of open source projects. [GitHub](#) and [BitBucket](#) are very popular hosting services on the cloud.

### Overview

It is important to clearly distinguish right from the beginning [Git](#) and [GitHub](#). [Git](#) is a general tool for versioning like [SVN](#). [GitHub](#) is a hosting service based on [Git](#) and also a particular tool (the [GitHub](#) client) that can be installed on your machine.

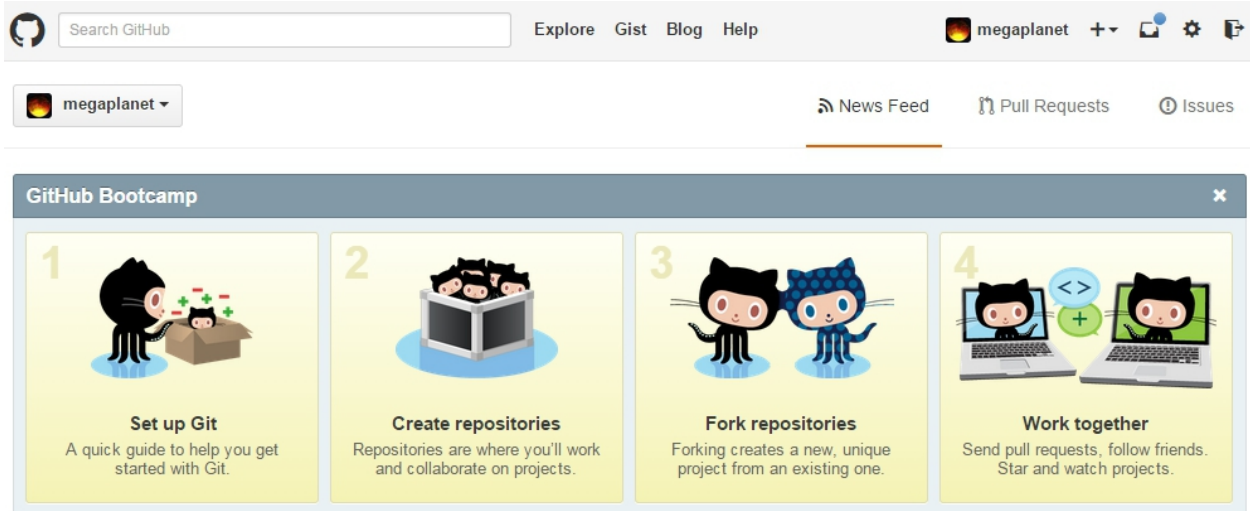


Fig. 1.33: GitHub splash screen

## Git

Git is a toolkit that can be used in a totally standalone mode, that is locally on any kind of machine (e.g. a laptop running windows/linux/mac). This is essentially a set of commands that you can run in a shell.

```

MINGW32/c/DEV/PyAlaOCL
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   pyalaocl/useocl/useengine.py

no changes added to commit (use "git add" and/or "git commit -a")

```

Fig. 1.34: A shell with a typical git command `git status`

Git can be used both locally by a single developer but also locally and on the web in a collaborative mode.

- **git for local usage.** One interesting aspect of Git is that it is file-based repository: installing it do not necessitate to install any kind of server, database; no daemon process is required, etc. Using git locally is therefore cheap and easy. It will just create a `.git` directory in your project directory and that's all. Full stop. Git can be useful really useful when working alone. This provides you some revision control for instance.
- **git for collaborative usage.** Obviously if the goal is to collaborate with others one need a *git server*. In this case the local copy of Git installed on your machine will serve as a *client*. You can setup a git server (if you are in a company) or reuse existing services from the cloud.

The next section is about git on the cloud.

## GitHub or/and BitBucket

If you need to collaborate with others to develop an open source project, [GitHub](#) or [BitBucket](#) are obvious solutions to consider, at least if you plan to use [Git](#). They both provide hosting public git repositories for free. GitHub and BitBucket have web-based interfaces to explore repositories.

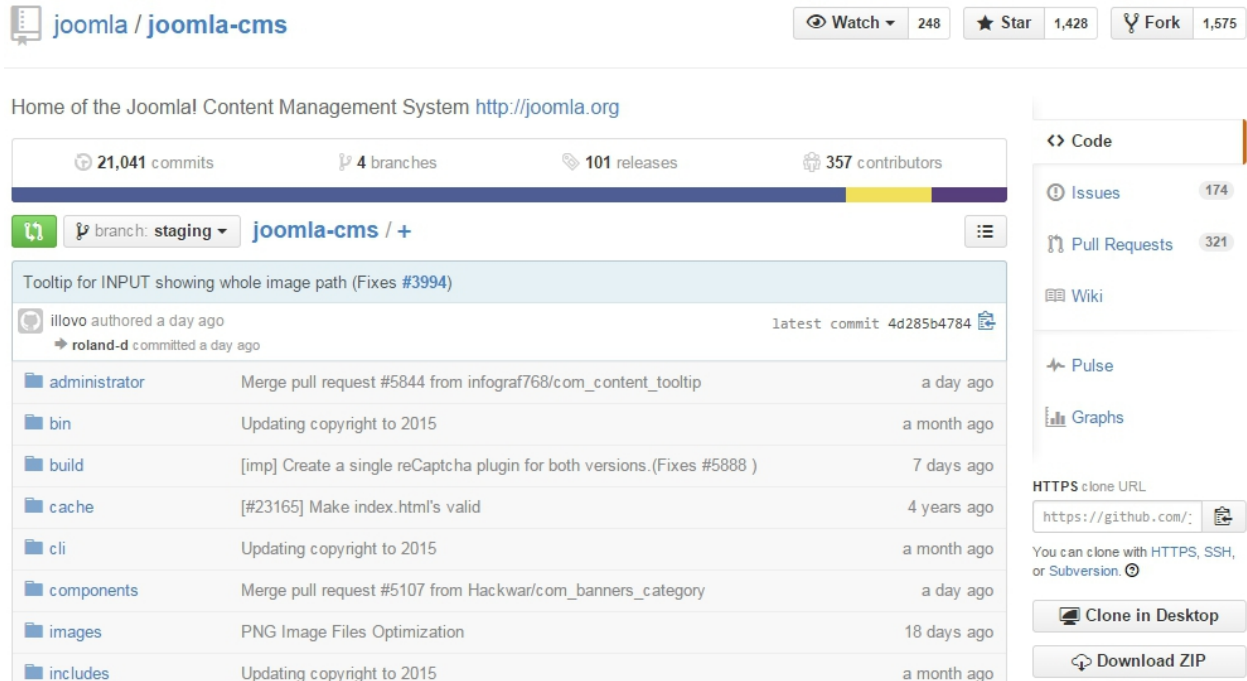


Fig. 1.35: A git repository as shown on github

We highly recommend to use GitHub, unless if you need private repository. If you want you can have a look at the article [bitbucket vs. github](#) or to have a look at a more general [comparison of hosting services](#).

Here we will use GitHub since:

- this is the most popular service,
- it provides an excellent *git client* which is amazingly easy to use.

If you need to have a private repository, and want to have that for free, then you have to use BitBucket for that repository. By contrast to Github (which offer only public repository), BitBucket provide unlimited private repositories (will a limit of 5 collaborators including you). The good news is that tools work with any provider so the solution is most probably: all public repositories on GitHub, all private repositories on BitBucket.

## Installation

In this section you will learn how to:

1. create a github account.
2. install the github client for your machine.
3. install a true 'git' toolkit (only if you need it).



The screenshot shows the Bitbucket interface for the repository 'jespern/django-piston'. The left sidebar has a 'Commits' link highlighted. The main content area shows a list of commits with the following details:

Author	Commit	Message	Date
Joshua Ginsberg	7c98898	Piston now supports Django 1.4 - thanks andialbrecht - Fixes #244	2012-03-31
Joshua Ginsberg	dbec00f	Closing imported branch name from andialbrecht	2012-03-30
Joshua Ginsberg	9ac3927	Merged 3a0d021dd042 from andialbrecht/django-piston-hmacfix	2012-03-30
Joshua Ginsberg	dafe5b9	Ignore docs/_build - Sphinxifying the docs underway...	2012-03-30
Andi Albrecht	3a0d021	Use _base_content_is_iter or _is_string on HttpResponse depend	2012-03-29
Joshua Ginsberg	023d4ae	Raise RuntimeError if emitters detect a recursive structure - fixes #164 (thanks	2012-03-30
Joshua Ginsberg	c0b6130	Added support for HTTP Accept header; explicit format in request string	2012-03-23
Joshua Ginsberg	8744863	Include proper admin.py and update test project accordingly. Fixes #117	2012-03-21
Joshua Ginsberg	d34db3f	Added rc response for HTTP status code 202 - Fixes #204	2012-03-21
Joshua Ginsberg	b9869d4	Added FIXME note to remember why list_fields is broken - Refs #157	2012-03-20
Joshua Ginsberg	43f41a4	Fixes #188 - bind request.FILES to form in @validate decorator	2012-03-20
Joshua Ginsberg	4fe8af1	Ported 0.2.2.1 tag to .hgtags	2011-11-01

Fig. 1.36: A git repository as shown on BitBucket

## Creating a GitHub account

Extremely simple. Just go to <https://github.com/join> and fill the various fields. In step 2 select a free plan. Basically this is it.

## Installing the GitHub client

At the time of writing this documentation, nice github clients exist only for Windows and for Mac. This step is anyway not obligatory. It install a nice graphical user interface for using git. This interface is extremely cool, but if you really want to use only shell commands and have already a git toolkit installed then you can skip this. If you are on linux you can go to the next section.

Installation is really simple:




- for Windows just visit <https://windows.github.com/> (windows installer web).

The screenshot shows the GitHub Desktop application. The left sidebar lists repositories, with 'octokit.net' selected. The main area shows the commit history for 'octokit.net', with a merge pull request #481 from 'octokit/sort-is-actually-case-sensitive' highlighted. The right pane shows the diff for the selected commit, with changes in the 'SearchClientTests.cs' file visible.

- for Mac just visit <https://mac.github.com/>.

# Join GitHub

The best way to design, build, and ship software.

 <b>Step 1:</b> Set up a personal account	 <b>Step 2:</b> Choose your plan	 <b>Step 3:</b> Go to your dashboard
---	--	--

## Create your personal account

**Username**

This will be your username — you can enter your organization's username next.

**Email Address**

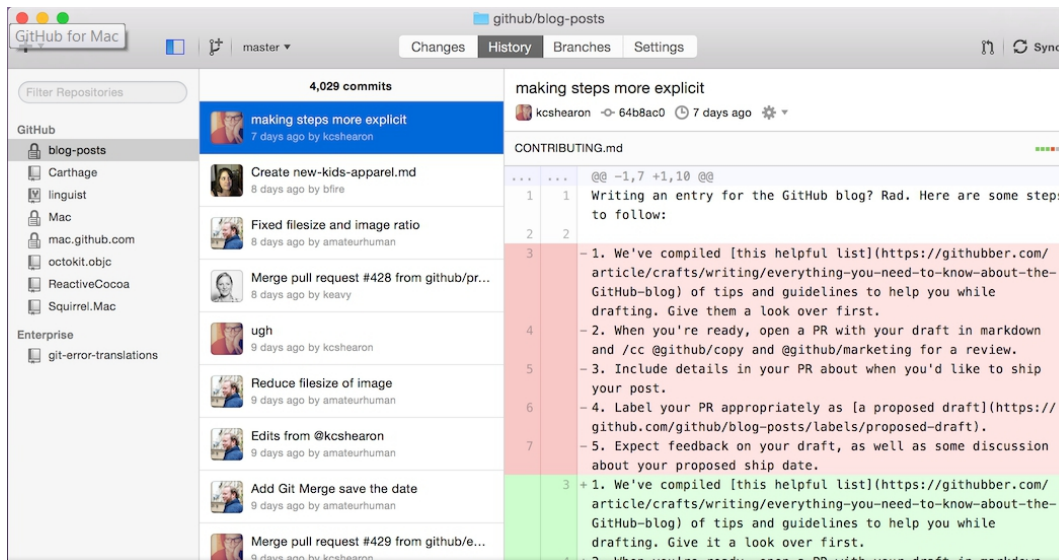
You will occasionally receive account related emails. We promise not to share your email with anyone.

**Password**

**You'll love GitHub**

- Unlimited** collaborators
- Unlimited** public repositories
- ✓ Great communication
- ✓ Friction-less development
- ✓ Open source community

Fig. 1.37: Joining GitHub - step 1



## Installing the BitBucket client

The github client is really easy to use. However if you want to learn more about git and want to do some advanced things without the shell, the github client is not the best choice. In fact, it hides some commands behind some cool features like 'Sync'. You might prefer another client, for instance the one proposed on BitBucket. This git client is

called SourceTree.

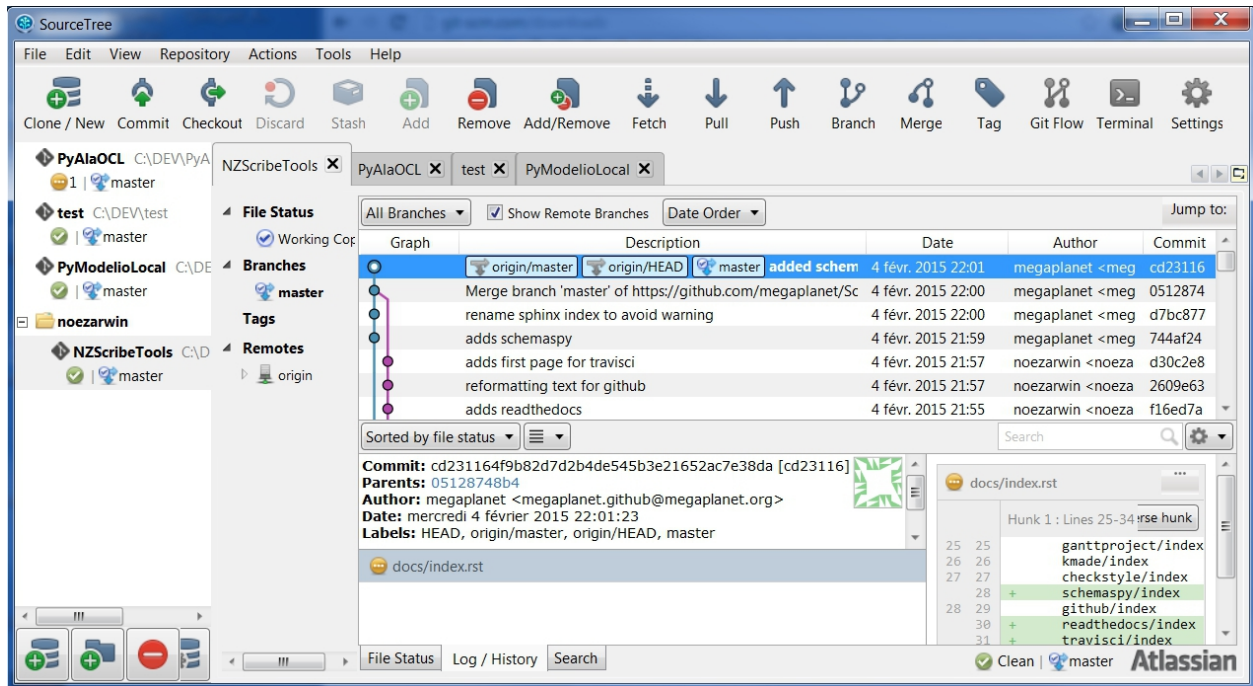


Fig. 1.38: SourceTree at work

You can download it from the [SourceTree home page](#). It works on windows and mac.

## Using the git shell

The github client installed in the previous sessions include not only a graphical interface, but also give you access to a git shell where you can type all commands.

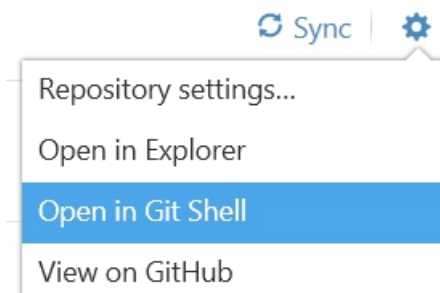


Fig. 1.39: Launching a git shell from the Windows GitHub client

While the GUI is very nice and enough for most situations, sometimes you will need to type git commands that are not available in the GUI. In this case using the git shell is the solution.

## Installing the git toolkit

In some situations you will however have to install a git toolkit.



Fig. 1.40: Git web site.

You have to install git:

- if you have not installed the GitHub client at all (either because you do not want or because it is not available on your platform).
- if you want to run some scripts with git commands but you cannot always run them from the GitHub shell (e.g. launching them from a cron tab).
- if you use some programs relying on the availability of git commands (those program will complain that git commands and not available).

In these case you will have to install git (possibly in addition to github client). In order to do so, just [download git](#) and install it following the instructions for your platform (e.g. for windows [web](#)).

On Ubuntu you can install git as following (if it is not already installed):

```
sudo add-apt-repository ppa:git-core/ppa -y      # if you want the last version
sudo apt-get update                            # if you want the last version
sudo apt-get install git
git --version
```

## Configuration

If you use the command line, you should configure git. The option `--global` save the configuration in your `.gitconfig` home file. Otherwise you could configure this for individual project.

Configure who you are (changing `escribis` by your github account):

```
git config --global user.name "escribis"
git config --global user.email "escribis@users.noreply.github.com"
```

To avoid typing always your password (see [Caching you Github password](#)):

```
git config --global credential.helper cache
git config --global credential.helper 'cache --timeout=3600'
```

You may also configure ([Dealing with line endings](#)):

```
git config --global core.autocrlf input      # if you use unix
```

## ScribesGit

The purpose of this page is to explain how to use Git and GitHub for course assignments.

General information about GitHub/Git can be found on in the [Git\(Hub\)](#). This page is really about using this general purpose tool for in the particular context of courses.

### Overview

Let us consider as an example the `aeis` course of the `m2r` school. with a typical scenario implying the a group `G12` in which two students working in pair:

- Noe and
- Babako.

Obviously **these values has to be replaced by actual values in the commands presented later.**

All information about the course `aeis` is available on [github.com](https://github.com). Each group can see three repositories:

Repository	Description	Access
<code>m2r-aeis-info</code>	Information and slides about the course	Read Only
<code>m2r-aeis-root</code>	Assignment skeletons (code, etc.)	Read Only
<code>m2r-aeis-G12</code>	Group repository	Read/Write

All repositories lives on GitHub. For instance `babako` and `noe` have at their disposal a “group repository”, at <https://github.com/m2r/m2r-aeis-G12>. This repository is private so only Noe, Babako and teachers can see it when logged in on GitHub.

This repository will be shared by Noe and Babako to work together. It will contain the final result evaluated at the end of assignments. In practice Noe and Babako will clone this repository on their local machines, will work separately on their local repositories and “push” and “pull” modification to the “group repository”. When the deadline will arrive the content of the “group repository” `m2r-aeis-G12` will be evaluated.

### Scenario

The steps below provides an overview of a possible process. Technical details are provided in the next sections.

1. Noe installs the git toolkit on her machine or uses git it at the university.
2. Noe first configures git on the local account(s) she uses (on her machine, at the university, or both).
3. Noe is at the university. She “clones” locally the “group repository” from GitHub in a home directory.
4. Noe “pulls” the assignment skeletons from the `m2r-aeis-root` repository. She gets the different files on her account at the university.
5. Noe puts Babako and her name in the `CONTRIBUTORS.md` file. She “commits” and “pushes” this changes to the “group repository” on GitHub.
6. Noe browses the assignments, looks at existing `WorkItems`, makes some first changes in order to start implementing the first assignment.
7. From time to time Noe “commits” and “pushes” the modifications to the “group repository” on GitHub. This will make it possible for her (or Babako) to continue to work at home or at the university later). She just has to remember to “push” the modifications to GitHub at the end of each work session. At some point Noe leaves the campus.

8. Noe arrives at home and want to continue on her laptop. She “pulls” the last modifications from the “group repository” and continue working on her laptop.
9. At anytime Babako can also make a “clone” of the “group repository”. He will get the last version “pushed” by Noe. He can then work in parallel on other issues for instance.
10. To get the last updates from the “group repository”, Noe and Babako “pull” the changes regularly. This allows them to incorporate modifications from each other. Since they are not fluent with git they avoid to modify the same parts of the same file at the same time. This helps avoiding [merge conflicts](#).
11. Deadline is about to arrived. Noe and Babako make sure that all their changes have been pushed to the “group repository” on GitHub. They also double check that the WorkItems has been updated and that they reflects what has been done.
12. The deadline arrives. Nothing has to be delivered: everything is already in the “group repository”. The work of the group is evaluated though the inspection of the content of the “group repository” at the deadline.

The following sections explain how to implement such a typical scenario.

## Installing Git

To install git on your machine (if not already installed) have a look at the [Installing the git toolkit](#).

## Configuring Git

You have to configure git once on *each* machine you use. For instance you may want to configure your account at the university and/or your account on your personal machine.

**Attention:** The values provided in this example **MUST** be replaced by actual values.

```
#---- configure git -----
# Attributes with --global goes the .gitconfig file.
# Configure the user associated with contributions (e.g. git push)
git config --global user.name "Noe ZARWIN"
git config --global user.email "noezarwin@users.noreply.github.com"

# Keep the password in memory for 2h
git config --global credential.helper "cache --timeout=7200"
# On some machine this method does not work and you will get an error message
# or a warning after each pull/push. In this case just remove the
# configuration option above using the following command:
#   git config --global --unset credential.helper

# OPTIONAL: Add a proxy ONLY if your machine is behind a firewall
git config --global http.proxy http://www-cache.ujf-grenoble.fr:3128

# OPTIONAL: Configure the editor used to edit message. Depends on the OS
git config --global core.editor "gedit -w -s" # For ubuntu

# To see current configuration you can use the "git config -l" command
# If you want to change something you can always edit the .gitconfig file
# using the following command (or any editor):
#   git config --global --edit
```



## Cloning the group repository

To create a local repository on your machine you have to “clone” your “group repository” (e.g. m2r-aeis-G12) from GitHub. This will create a local repository on your machine where you can work locally.

**Attention:** The values provided in this example MUST be replaced by actual values.

```
#---- Clone the "group repository" and into a "local repository" -----
# Go to your home directory
cd # On unix

# The "group repository" is at URL like (check this when connected to GitHub)
# https://<github_account>@github.com/<grade>/<grade>-<class>-<group>.git
# The GitHub account is specified explicitly (noezarwin below).
# The following command will ask for the corresponding password.
# Clone it in the current directory.
git clone https://noezarwin@github.com/m2r/m2r-aeis-G12.git
# If you get a message 'Failed to connect to github.com port 443: Time out'
# it is most probably that your machine is behind a firewall and that
# you need to define http.proxy (see the Configuration section above).
# If you get a message indicating that the repository does not exist
# this can either be due to:
# * an error in the url. Check it again and don't miss .git at the end.
# * a proper read access on the repository might be missing
#   the given login.
#   Check this by connecting to GitHub with this login.

# Enter the newly created directory.
cd m2r-aeis-G12
```

Two situations are possible here:

- (1) The **repository is empty**. If you are the first of your group performing this series of steps, your group repository could be empty. There will be at least the ‘.git’ hidden directory. That’s ok. Just continue.
- (2) The **repository is initialized**. If (an)other(s) member(s) of the group already followed these instructions, your group repository will already contains their work. This is fine. You will get a non-empty directory. There is in particular a .git hidden directory. That’s ok. Just continue.

Simply put, this directory contains the “local repository”. This directory is managed through git commands.

## Getting assignment skeletons

You now have to configure your repository to get assignment skeletons from the “root repository”. The “root repository” is maintained by teachers. This directory contains work definitions, directory structures, file skeletons, and so on.

**Attention:** The values provided in this example MUST be replaced by actual values.

```
#---- Declare the "root directory" and "pull" files from it -----
# Declare the m2r-aeis-root as a remote repository.
# You can check that you have access to this repository by logging in
# on GitHub and visiting https://github.com/m2r/m2r-aeis-root .
# Your declaration below should be done only once for each local repository.
git remote add root https://noezarwin@github.com/m2r/m2r-aeis-root.git
```

```
# If you want to see the list of remote directories use the
# command "git remote -v". If you made a mistake in the URL and need to change
# it use the command "git remote set-url <newurl>".

# "Pull" the assignment skeletons from the "root directory".
# If an editor opens just enter a message like "get assignment skeletons"
git pull root master
# You should now have the assignment skeletons in the local repository.
# Note that if you get an error at this level this could be either because:
# * you've made an error in the url above. Use git remote -v to check it.
#   If there is an error use the following command
#       git remote set-url root <the-url>
# * you do not have read access to this repository. Please check this
#   going on GitHub and check if you see it with your login.

# You can browse the content of the directory with "ls -la" on unix.
# There is one directory per assignment.
```

## Changing CONTRIBUTORS.rst

You have to fill the CONTRIBUTORS.rst file in the repository and to put the information about your group using the format such as below.

n	group	trigram	firstname	lastname	githubAccount	
1	G12	BST	Babako	SCHMIDT	babako12	babako.schmidt
2	G12	NZN	Noe	ZARWIN	noezarwin	noezarwin@gmail

There should be one line for each member. The list must be **sorted by lastnames**. Spaces and blank lines are important in the RST format.

**Attention:** The values provided in this example MUST be replaced by actual values.

```
#---- Edit CONTRIBUTORS.md, commit and push the change -----
# Use your favorite editor to edit CONTRIBUTORS.md.
# Enter the data about all group members in the following format.
```

The fields are the following:

**N** The indice of the member in the list. Members must be list in alphabetical order on the lastname, then firstname.

**group** The group number (e.g. G12).

**trigram** Three uppercase letters:

- the first letter of the firstname
- the first letter of the lastname
- the **last** letter of the lastname

See the quality rule [Trigramme](#) for details about composite names.

**firstname** The firstname of the member (e.g. Babako).

**lastname** The lastname of member using UPPERCASES\$ (e.g. SCHMIDT).



**githubAccount** The login used by the member to connect to GitHub.

**email** A valid email address.

**Attention:** The lines must be **sorted by lastnames** (ascending order). This is fundamental for defining the n indice.

The lastname must be all in uppercases.

Change the width of columns if you need more space for your name, email, etc. A strict alignement is necessary for the .rst processor to parse this table correctly.

```
# Save the file.
#
# Add the modified file to the files to be saved in the next commit
git add .

# Commit (e.g. save) the changes to the local repository
git commit -a -m "Set the authors for this repository"

# Push (e.g. publish) the state of the local repository to github
git push origin master
# If you get an error here indicating that there is no such repository
# this could be because you don't have write access to this repository.
# Go to GitHub using the login used in the url and check if you can edit
# files. If not post an issue in the root repository and writing rights
# will be associated to your account.
```

The changes should now appear on GitHub “group repository”. Log in to GitHub and go to your group repository (e.g. <https://github.com/m2gi/m2gi-idm-G12>) to check.

## Making and pushing changes

Time to work and deal with assignments. The process is all about making changes, committing these changes to the “local repository” and pushing these changes on GitHub to the “group repository”.

```
#---- Making changes, committing and pushing them -----
# Make some changes.

# Check which files have changed.
# Use the "-s" option if you prefer a shorter format.
git status

# Add files to be committed. Replace <files> below by actual file names.
# Use "git add ." to commit the whole directory
git add <files>

# Commit the files (save them in the local repository)
# Provide a useful message instead of <message>.
git commit -a -m '<message>'

# OPTIONAL: push changes to the "group repository" on GitHub
# You must do this at the end of a working session if you
# plan to continue on another machine (at home for instance)
# or if you want other group members to "see" the changes.
git push origin master
```

## Pulling changes from the group repo

If you work on various machines or if other group members work in parallel your local repository may not contain the last changes available on GitHub in the group repository. In this case you have to “pull” these changes as following.

```
#---- Pulling changes from the group repository on GitHub -----  
# Before making a "pull" make sure that you have committed all changes.  
# "origin" refers to the "group repository" on GitHub.  
# The "pull" command download the latest changes from the "group repository"  
# then it try to merge these changes with those made locally.  
git pull origin master
```

Pulling changes may cause some merge conflicts. See [resolving merge conflicts](#) in this case.

## Pulling changes from the root repo

During the course new assignments may be created and/or new material may be added into an existing assignment, for instance to bring precision to some tasks or to add additional skeletons. These changes will be made available through the “root repository” which contains assignment skeletons. In order to get last updates you just have to pull these changes in the same way you pull changes from your “group repository”.

```
# Before making a "pull" make sure that you have committed all changes.  
# "root" refers to the "root repository" on GitHub.  
# This "remote" repository has been declared in the "Getting assignment skeletons"  
# section.  
git pull root master
```

Pulling changes may cause some merge conflicts. See [resolving merge conflicts](#) in this case.

## Staying informed

In GitHub terms, “Watching” a repository means receiving notification when changes occur to it.

Since you are member of your “group repository” you should automatically receive notifications for new commits for instance. This is handy to keep in synch with other group members. By default you “Watch” this repository but you can change this by pressing on the “Unwatch” button on GitHub.

If you want to stay informed you may also want to “Watch” the following repositories.

- the “info” repository for general information about the course. This can be useful to get notified when new slides are added for instance.
- the “root” repository. Register to this repository if you want to receive information about assignments, get notification when questions are posted, etc.

---

**Note:** If you receive too much notifications you can change the settings at any moment.

---

## Questions/Bugs/...?

If you found a bug in an assignment, if you have some comments or or have a question about the course please post an [GitHub issue](<https://guides.github.com/features/issues/>).

Please select the repository that is most suited to your issue:

- If the “issue” is general or related to a particular assignment and the question/issue is relevant to other groups, then post the issue in “root” repository.

- If the “issue” is only related to your group (you and other group member partner) please post the issue in your “group” repository (m2r-aeis-G12 for instance). Use the (!) button in the web interface (see [create an issue](<https://guides.github.com/features/issues/>) for details).

If you have some answer to some posted issues, please provide it directly online.

**Attention:** Use issues to communicate, not emails.

## ReadTheDocs

ReadTheDocs is a cloud service for publishing and hosting documentation using the [Sphinx](#) documentation generator (see [Sphinx](#)). The documentation your are reading is hosted on [readthedocs](#).

### Features

ReadTheDocs provide the following services: \* hosting of documentation of open source projects \* continuous [Sphinx](#) generation (connected to GitHub) \* management of documentation versioning within the browser \* support for downloading documentation in html, pdf, and e-pub formats \* searchable documentation

### Installation

No installation is required as this service is on the cloud. However you must create an [ReadTheDocs account](#).

Then you need to connect this account with your GitHub account. Once this is done you can import the projects that have sphinx documentation.

### Usage

Once the ReadTheDocs and GitHub account are connected, the documentation pushed on the github repository is automatically compiled and published on the ReadTheDpcs web site.

It is worth to use the [sphinx rtd theme](#) as it provides an excellent rendering both on computer screens but also on tablet or smart phones.

### Documentation

The web site is rather easy to use, but if you need documentation for specific topics you can have a look at [ReadTheDocs documentation](#).

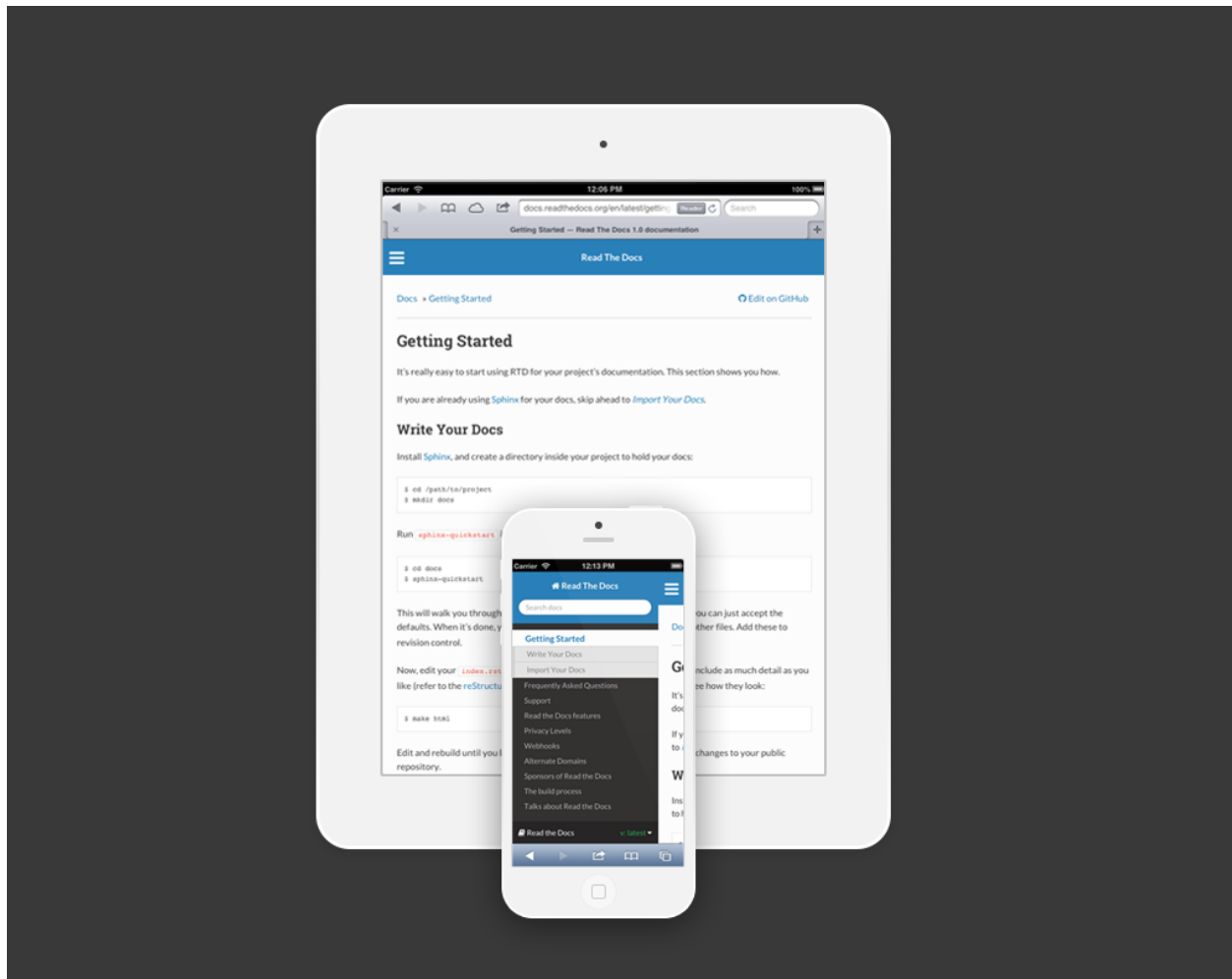


Fig. 1.41: sphinx rtd theme

## Selenium

### Installation

### Documentation

## TravisCI

TravisCI is a Continuous Integration (CI) system. It helps automating tests on git repository changes.

### Documentation

TravisCI could be configured for repositories based on a given list of programming languages. This includes in particular:

- documentation for [Python](#)
- documentation for [Java](#)

## Java

### Installation

One should distinguish two java components:

- **JRE**: The Java Runtime Environment enables **end-users** to **execute** java programs.
- **JDK**: The Java Development Toolkit allows **developers** to **develop** java programs.

The JRE can be installed and used independently from the JDK. By contrast, the JDK comes usually with a JRE. Here are short instructions to install the JDK (including the JRE). More information is available on the [java installation guide](#):

On Windows:

- Download the jdk from the [java download page](#).
- Create the following directory structure:

```
%SCRIBESTOOLS%
  Java
    jre
    jdk
```

- Install the **JDK** in %SCRIBESTOOLS%\Java\JDK and the **JRE** in %SCRIBESTOOLS%\Java\jdk
- Add the following directory to the system PATH:

```
%SCRIBESTOOLS%\Java\jdk\bin
```

- Set the JAVA\_HOME environment variable to:

```
%SCRIBESTOOLS%\Java\jdk
```

On Ubuntu:

```
# Remove existing OpenJDK
sudo apt-get purge openjdk*
# Install Oracle Java 8
# if add-apt-repository is missing in the next command
#     sudo apt-get install software-properties-common
#     sudo apt-get install python-software-properties
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
# Set Java Environment Variable
sudo apt-get install oracle-java8-set-default
```

If this does not work try instead to follow the instructions [how to install oracle java 9 in debian](#).

## Launching Java

To test the JRE type the following command:

```
java -version
```

To test the JDK type the:

```
javac -version
```

## Python

According to wikipedia “[Python](#) is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.”

## Installation

This section shows how to install Python 2.7, how to use create virtual environments, how to install necessary libraries including windows specific libraries. If you want to use Jython/Python in the context of modelio (see [Modelio](#)) you do not have anything to install as Jython 2.7 is embedded into this tool.

### Installing Python 2.7

Detailed information is available in the [BeginnersGuide](#) of python wiki. On most unix systems python is usually installed so try `python -V` on the command line. If the version is `>= 2.7.6` you can skip this step.

**Note:** There are two python ecosystems: python 2.X and 3.X. There some [language differences](#) between Python 2 and Python 3. Since not all libraries have been ported to 3.X Python 2 is still the most common choice when a lot of libraries are required.

- [Download Python 2.7.10](#). **WARNING: On windows you must select the 32 bits version (web)**.
- Install python in a directory like `%SCRIBESTOOLS%\Python27`. Keep the version number (at least 2.7) since it is common to have another directory like `%SCRIBESTOOLS%\Python34` for Python 3.4.

```
msiexec /i c:DOWNLOADSWinpython-2.7.10.msi TARGETDIR=%SCRIBESTOOLS%\Python27
ALLUSER=1 ADDLOCAL=ALL
```

- Change the two following directories to the PATH environment variable (if not already done by the installation program).

```
# replace ';' separator by ':' on unix
# add %SCRIBESTOOLS%\Python2.7;%SCRIBESTOOLS%\Python2.7\Scripts to PATH
```

- To test your installation, open a shell windows and type python. The python interpreter should open and you should be able to type print 'hello world' for instance. quit() will close the interpreter.

## Virtual environments

Install [virtualenvwrapper](#) (or [virtualenvwrapper-win](#) on windows). This can be done with one command (according to your platform):

```
pip install virtualenvwrapper-win # on windows
pip install virtualenvwrapper    # otherwise
```

Create a directory that will contains all virtual environments. For instance %SCRIBESTOOLS%\PyVEnvs27:

```
mkdir %SCRIBESTOOLS%\PyVEnvs27
```

Set the WORKON\_HOME environment variable to the directory just created and don't forget to open a new shell to see the effects of this change.:

```
# define WORKON_HOME variable as %SCRIBESTOOLS%\PyVEnvs27
```

Use a new shell to create a new virtual environment named ScribeEnv:

```
mkvirtualenv ScribeEnv
# this creates a directory ``%SCRIBESTOOLS%\PyVEnvs27\ScribeEnv``.
# You should also see that the prompt of the shell is now prefixed with
# (ScribeEnv) like in the following line:
# (ScribeEnv) C:\Users\jmfavre>
```

This (ScribeEnv) indicates that the python virtual environment used in this shell is ScribeEnv. This is valid only for *this* shell. If you want to open another shell you will need to “**activate**” this virtual environment:

```
workon ScribeEnv
```

From now on, you will have to activate the ScribeEnv virtual environment each time you open a new shell and want to use/install python software.

## Libraries

Some python libraries depends on the OS (those who are using C libraries for instance). Select the section corresponding on your OS:

- installing *Python libraries on windows*.
- installing *Python libraries on unix*.

### Libraries on windows

This section is devoted to installing python libraries on windows.

### Native libraries

**Note:** Some python libraries need to be compiled with a C compiler. This is tricky on standard windows boxes. When a python library is not available for windows, a good idea is to look on the following web site: [Unofficial Windows Binaries for Python Extension Packages](#).

---

**pywin32** `pywin32` is a set of python extensions for Windows. This library is required in particular by `Scrapy`.

- download the 32 bits version `pywin32-219.win32-py2.7.exe` ([web](#)).
- **do not click on the executable** but type instead the following command (make sure that `(ScribeEnv)` is in the prompt):

```
easy_install C:\DOWNLOADS\pywin32-219.win32-py2.7.exe
```

**pygraphviz** `pygraphviz` is a python API to use the `GraphViz` graph package.

- download `pygraphviz1.3rc2cp27nonewin32.whl` ([web](#)). This version works with `graphviz 2.38.msi`.
- **do not click on the executable** but type instead the following command (make sure that `(ScribeEnv)` is in the prompt):

```
pip install c:\DOWNLOADS\pygraphviz-1.3rc2-cp27-none-win32.whl
```

**pycrypto** `PyCrypto` is a Python cryptography package used by other packages.

- download `pygraphviz1.3rc2cp27nonewin32.whl` ([web](#)).
- Type instead the following command (make sure that `(ScribeEnv)` is in the prompt):

```
easy_install.exe c:\DOWNLOADS\pycrypto-2.6.win32-py2.7.exe
```

**pillow** `Pillow`, a replacement for `PIL`, the Python Image Library.

- download `Pillow-3.1.1-cp27-none-win32.whl` ([web](#)).
- Type instead the following command (make sure that `(ScribeEnv)` is in the prompt):

```
easy_install.exe c:\DOWNLOADS\Pillow-3.1.1-cp27-none-win32.whl
```

**Python libraries** To install regular python libraries (those that are based on python only) type the following command (make sure that `(ScribeEnv)` is in the prompt):

```
pip install XXXX\requirements-windows.txt
```

### Libraries on Unix

This section is devoted to installing python libraries on unix.



**Native libraries** On **Ubuntu 14.04** the following native libraries have to be installed for python libraries to work properly.

**Tip:** For other unix systems try first to install the python libraries (see next section) and check on Google what to do if you got error messages.

```
sudo apt-get install autoconf g++ python2.7-dev python-dev
sudo apt-get build-dep python-imaging
sudo apt-get install libjpeg8 libjpeg62-dev libfreetype6 libfreetype6-dev
sudo apt-get install libffi-dev libssl-dev libxml2-dev libxslt1-dev
```

# for Pillow  
# for Pillow

**Python libraries** To install python libraries first download `requirements-ubuntu.txt`. Then change the path in the following command by the path to your virtualenv directory (e.g. `/usr/share/PyVEnvs27/ScribesEnv`) and type:

```
sudo /<path-to-yout-virtual-env>/bin/pip install requirements-ubuntu.txt
```

## Launching Python

To test your python installation try the following command:

```
python -V
```

## Documentation

The best way to find information about python is just to ask question such as “python read file at once” on google. You may also want to have a look at `J/Python in a nutshell` and print a cheat sheet :

- `CheatSheetA`
- `CheatSheetB`
- `CheatSheetC`
- `FrenchCheatSheetD`
- `RegExCheatSheet`

## PyCharm

[PyCharm](#) is an excellent IDE for the Python language.

### Overview

Python being a dynamic language, using an IDE is a really good idea. This will help you find errors at development time rather than at runtime. There are various IDE for python around (see [python IDE comparison](#)).

If you are going to write non trivial python programs for some time, then we recommend using [PyCharm](#). It will save you time. [PyCharm](#) is usually faster, simpler and more powerful than [PyDev](#), the python environment on eclipse. If you are really a huge fan of eclipse, you might prefer [PyDev](#) indeed.



**PyCharm**

Overview What's New Features & Screenshots Docs & Demos Quickstart Guide Download Buy & Renew

## The Most Intelligent Python IDE

Enjoy productive Python, Django, and Web development with PyCharm, an intelligent Python IDE offering unique coding experience.

[Get PyCharm Now](#)

Full-fledged Professional or Free Community

Intelligent Coding Assistance

Smart Code Navigation

Fast and Safe Refactoring

Debugging and Testing

VCS and Deployment

## Installation

There are two versions:

- An open source version which is quite complete and very useful.
- A commercial version which has support for django, databases, etc.

**Attention:** If you are student you can very easily get [PyCharm for students](#). You just need to register with the email of your university. It's really easy and quite fast.

- [download PyCharm](#) page and then download the appropriate version (e.g. Windows community edition ([web](#)), Windows commercial edition ([web](#))).
- install the version downloaded in a directory like `%SCRIBESTOOLS%\PyCharmOpen` or `%SCRIBESTOOLS%\PyCharmCommercial`. This naming schema allows to have both editions installed at the same time.
- by default your personal configuration of PyCharm goes in a directory like `.PyCharm` in your home directory, but if you want you can change this by editing the file `bin/idea.properties` in the installation directory. You have to adjust for instance the following two lines:

```
idea.config.path=Z:/.PyCharm50/config
idea.system.path=Z:/.PyCharm50/system
```

## Launching PyCharm

The installer most likely created a shortcut. Click on it to launch PyCharm. Alternatively you can also find the executable in the bin directory of the installation or in a standard bin directory (e.g. `/usr/bin/pycharm`).

### Launching PyCharm the first time

When launching pycharm for the first time, you will have no `.PyCharm50` in your home directory. This directory contains the global configuration describing your global preference, your environment, the licencing information, and so on. PyCharm will therefore ask a few questions.

If you use the commercial version of PyCharm, you have to provide information for:

- your pycharm/jetbrain account (created before) if you work at home for instance,
- or a pycharm server licence if you work at university for instance. If you are in the context of the UGA enter `http://im2ag-licence.e-im2ag.ujf-grenoble.fr:1111` for the licence server.

### Opening/Creating PyCharm projects

Just like eclipse or netbeans, PyCharm is NOT a text **file** editor. So one do not open a file when launching PyCharm, but on the contrary create a **project** at a given place; and this project (or other projects) will always be opened and used in next sessions. A PyCharm project is simply a directory with a `.idea` subdirectory. This `.idea` subdirectory is created on the first time and then managed by PyCharm.

*When launching PyCharm you have to select your “project” directory, that is the directory that contains all files you want to work on. Usually this is the directory that also contains versioning information, so it is most likely that this project directory will contain both `.idea` and `.git` subdirectories (if you are using Git).*

**Note:** The normal way to proceed with PyCharm is to open only projects and you should therefore have only one `.idea` directory at the top level of your “project directory”.

## Documentation

Documentation is available from the `help` menu of the tool.

## Configuration

To configure [PyCharm](#) use the menu `File > Settings`:

```
Editor
  File Encodings
    UTF8
  Appearance
    Show line numbers
    Show method separators
Code Style
  General
    right margin (columns)
```

## Django

[Django](#) is a open source web application framework written in Python. Django's primary goal is to ease the creation of complex, database-driven websites.

## Features

[Django at Glance](#) gives an idea about django development.

## Installation

Django is based on python. Follow the instructions in the [Installation](#) section if not already done. These instructions install python but also [Django](#) as well as many useful [Django](#) components.

**Note:** The installation is done in the `ScribeEnv` virtual environment. Make sure to execute the following command:

```
workon ScribeEnv
```

The shell prompt should then start with `(ScribeEnv)` meaning that you are working in the proper virtual environment.

## Launching Django

If you want to check that [Django](#) is installed you can try the following:

```
workon ScribeEnv
python -c "import django; print(django.get_version())"
django-admin --version
```

## Documentation

In the open source community [Django](#) is reknown for the quality of its [documentation](#). After reading [Django at Glance](#) the best way to start is to read the [Django Tutorial](#).

## SQLite

[SQLite](#) is a file-based SQL database system.

### Installation

- download the “command line shell” for “sqlite3” from [SQLite download page](#). For Windows download [sqlite-shell-win32-x86-3081002.zip web](#).
- create a directory where you want to install sqlite3 (e.g. %SCRIBESTOOLS%\SQLite)
- extract the archive (only one file on windows: `sqlite3.exe`)
- move the content of the archive to the directory created
- add %SCRIBESTOOLS%\SQLite to the path.

On Ubuntu you can install [SQLite](#) as following:

```
sudo apt-get install sqlite3
```

### Launching SQLite

You can test the installation and check the version of sqlite3 with the following command:

```
sqlite3 -version
```

Then try:

```
sqlite3
```

This should launch the [SQLite](#) shell. Try the following commands:

```
.help  
.schema  
.quit
```

Since no database has been specified when launching the shell command the `.schema` command will not display anything.

### Sqlite3 for django

If you use [SQLite](#) with django, then you can launch the shell with the following commands:

```
python manage.py dbshell
```

## Documentation

The documentation is available online ([SQLite documentation](#)).

- [SQLite dialect](#).
- [SQL features omitted from SQLite](#)

To get some help about the command line type:

```
sqlite3 -help
```

If you want some brief help about the shell commands (including SQL) you can type in the shell:

```
.help
```

More generally, the documentation of the shell is available at <https://www.sqlite.org/cli.html>

## MySQL

[MySQL](#) community is an open source RDBMS.

## Installation

On Ubuntu you can install [MySQL](#) as following:

```
sudo apt-get install mysql-server
```

For more details you can have a look at this [short tutorial](#).

## Documentation

The documentation is available online ([MySQL documentation](#)).

## Graphviz

Graphviz is a set of tool to deal visualize graphs. See the [gallery](#).



## Installation

- download the installer for your operating system the [download page](#).

On Windows `graphviz-2.38.msi` must be used, at least if you plan to use the [pygraphviz](#) windows library for python [web](#).

On Ubuntu [Graphviz](#) can be installed via:

```
apt-get install graphviz libgraphviz-dev pkg-config
```

- install the program in a directory like `%SCRIBESTOOLS%\Graphviz\`
- add `%SCRIBESTOOLS%\Graphviz\` into the `PATH` environment variable.

## Launching Graphviz

To test your installation type the following command in a new shell. It should display the version of graphviz.

```
dot -V
```

You can try the `gvedit`, a very simple graphical interface. It is not really useful though as [Graphviz](#) is mostly used by other programs.

```
gvedit
```

## Documentation

There is a lot of [documentation](#).

## Sphinx

[Sphinx](#) is a documentation generator based on the reStructuredText (rst for short) for short. It can be used to publish documentation on [ReadTheDocs](#) (see [ReadTheDocs](#)).

## Installation

[Sphinx](#) is written in Python. Follow the instructions in the [Installation](#) section. This will install Python as well as [Sphinx](#) and related components.

---

**Note:** The installation is done in the `ScribeEnv` virtual environment. Make sure to execute the following command:

```
workon ScribeEnv
```

The shell prompt should then start with `(ScribeEnv)` meaning that you are working in the proper virtual environment.

---

## Launching Sphinx

To try the installation of [Sphinx](#) type the following command and press Ctrl-C to exit from the program:

sphinx-quickstart

## Documentation

An extensive documentation is available on the [Sphinx](#) and [rst](#) web sites. To learn [rst](#) different kinds of resources are probably better:

- **Examples.** The best way to learn [rst](#) is probably to look at the source of existing rst texts. For instance on [ReadTheDocs](#) the documentation usually include a link `View Code` on the top-left corner of each page.
- **Cheat Sheets** Here is a list of cheat sheets ordered approximatively by size.
  - [CheatSheet #1](#)
  - [CheatSheet #2](#)
  - [CheatSheet #3](#)
  - [CheatSheet #4](#)
- **Nice pages.** Some interesting pages from [Sphinx](#) documentation:
  - [reStructuredText Primer](#): a extended introduction
  - [Sphinx Markup Constructs](#): a rather easy-to-use reference

If you have already some knowledge about markdown you may be interested by [Markdown vs. rst](#) comparison.

## Extensions

- [sphinx-contrib](#)
- [awesome-sphinx](#)
- [sphinxext-survey](#)

### sphinx-googlemaps

**pip** `sphinxcontrib-googlemaps`

`sphinx-googlemaps` allows to embed maps using Google Maps.

### sphinx-googlechart

**pip** `sphinxcontrib-googlechart`

`sphinx-googlechart` allows to embed googlecharts.

### sphinx-libreoffice

**pip** `sphinxcontrib-libreoffice`

`sphinx-libreoffice` allows to embed libreoffice images.



## images

**pip** sphinxcontrib-images

images enables thumbnails, fancybox, group of images, captions, tooltip.

## css3image

**pip** sphinxcontrib-css3image

css3image provides an enhanced image directive with additional CSS properties.

## svg

**pip** download

svg provides svg templates. The name of the extensions is docutils\_ext...

## swf

**pip** sphinxcontrib-swf

swf allows embedding of flash swf files.

## youtube

**pip** sphinxcontrib.youtube

youtube allows embedding of youtube videos.

## sphinx-embedly

**pip** sphinxcontrib-embedly

sphinx-embedly allows to embed whatever can be embedded with embedly.

## sphinx-twitter

**pip** sphinxcontrib.twitter

sphinx-twitter allows to embed tweets.

## sphinx-sadisplay

**pip** sphinxcontrib-sadisplay

sphinx-sadisplay renders SQLAlchemy models with PlantUML diagrams or GraphViz directed graphs.

## schema2rst

**pip** schema2rst

schema2rst generates reST doc from database schema. Works with mysql, mysql+pymysql, postgresql.

## sqltable

**pip** sphinxcontrib-sqltable

**sqltable** allows to embed SQL statements in source documents and produce tabular output.

## exceltable

**pip** sphinxcontrib-exceltable

**Exceltable** adds support for including spreadsheets, or part of them, into Sphinx document

## rusty

**pip** rusty

**Rusty** is a collection of extensions:

**rusty.exceltable** Creates table from selected part of the Excel

**rusty.includesh** Extends the standard include directive by converting the given shell script

**rusty.regxlist** Creates bullet list based on regular expression rule. Similar to rolist directive.

**rusty.rolelist** Creates the bullet list from all the entries of the selected role, with some additional ways to custom the output.

## doctest

**pip** standard

**doctest** allows to add test snippets in the documentation in a natural way. It works by collecting specially-marked up code blocks and running them as doctest tests.

Within one document, test code is partitioned in groups, where each group consists of: \* zero or more setup code blocks (e.g. importing the module to test) \* one or more test blocks

## domaintools

**pip** sphinxcontrib-domaintools

**domaintools** provides a tool for easy sphinx domain creation.

## jinjdomain

**pip** sphinxcontrib-jinjdomain

**jinjdomain** provides a Sphinx domain for describing jinja templates.

## makedomain

**pip** sphinxcontrib-makedomain

**makedomain** provides a GNU Make domain.

## phpautodoc

**pip** tk.phpautodoc

phpautodoc enables to embed PHPDocs to sphinx document.

See also sphinx-php and \_phpdomain.

## httpdomain

**pip** sphinxcontrib-httpdomain

httpdomain provides a Sphinx domain for describing RESTful HTTP APIs.

## autojs

**pip** sphinxcontrib-autojs

autojs generates a reference documentation from a JavaScript source file.

## autoanysrc

**pip** sphinxcontrib-autoanysrc

autoanysrc insert reST docs from files to target documentation project without auto generation definitions and signatures.

## autoprogram

**pip** sphinxcontrib-autoprogram

autoprogram provides an automated way to document CLI programs. It scans argparse.ArgumentParser object, and then expands it into a set of .. program:: and .. option:: directives.

## autorun

**pip** sphinxcontrib-autorun

autorun execute the code from a runblock directive and attach the output of the execution to the document.

## programoutput

**pip** sphinxcontrib-programoutput

programoutput inserts the output of arbitrary commands into documents.

## jsoncall

**pip** sphinxcontrib-jsoncall

jsoncall adds a simple button to perform test calls to JSON based apis making also possible to change parameters values through a set of input fields.

## jsdemo

**pip** sphinxcontrib-jsdemo

`jsdemo` is an extension for embedding HTML/Javascript demo snippets.

## releases

**pip** releases

`Releases` is a Sphinx extension designed to help you keep a source control friendly, merge friendly changelog file & turn it into useful, human readable HTML output. The source format (kept in your Sphinx tree as `changelog.rst`) is a stream-like timeline that plays well with source control & only requires one entry per change (even for changes that exist in multiple release lines)

## sphinx-github

**pip** sphinxcontrib-github

`sphinx-github` shows github repos and pull requests.

## graphvizinclude

**pip** qubic.sphinx.graphvizinclude

`graphvizinclude` enables graphviz generation of external dot files.

## sphinx-yuml

**pip** sphinxcontrib-yuml

`sphinx-yuml` allows rendering of plots using `yUML` service.

## sphinx-seqdiag

**pip** sphinxcontrib-seqdiag

`seqdiag` allows to embed sequence diagrams generated with `seqdiag`.

Error message when installing pillow on ubuntu.

## sphinx-sdedit

**pip** sphinxcontrib-sdedit

`sphinx-sdedit` allows to embed sequence diagrams generated with `_sdedit`.

## sphinx-plantuml

**pip** sphinxcontrib-plantuml

`sphinx-plantuml` enables to embed `plantuml` diagrams.

## sphinx-pyreverse

**pip** sphinx-pyreverse

sphinx-pyreverse generates UML diagrams with pyreverse.

## slide

**pip** sphinxcontrib-slide

slide enable you to embed your slides on slideshare and other sites.

## hieroglyph

hieroglyph slides

## tut

**pip** tut

Tut is a tool that helps you write tutorial style documentation where sections build on one another, and include code examples along the way.

## spelling

**pip** sphinxcontrib-spelling

sphinxcontrib-spelling is a spelling checker for Sphinx. It uses PyEnchant to produce a report showing misspelled words.

## remoteinclude

**pip** download

remoteinclude is just like literalinclude but for remote files.

## hiddencode

**pip** download

hiddencode adds a directive for a highlighted code-block that may be toggled hidden and shown in HTML

## classy-code

**pip** sphinx-classy-code

classy-code provides drop-in replacements for Sphinx' code-block and literalinclude directives. In addition to specifying emphasize-lines, you can specify arbitrary classes to add on a per-line basis.

## getthecode

**pip** sphinxcontrib-getthecode

`getthecode` adds a new directive `getthecode` which is equivalent to the `literalinclude` directive, but adds in front of the code block an header with the file name and an icon to download the file.

## viewcode

**pip** standard

`viewcode` looks at your Python object descriptions (`.. class::`, `.. function::` etc.) and tries find the source files where the objects are contained. When found, a separate HTML page will be output for each module with a highlighted version of the source code, and a link will be added to all object descriptions that leads to the source code of the described object. A link back from the source to the description will also be inserted.

## linkcode

**pip** standard

`linkcode` is like `viewcode` but assumes the source code can be found somewhere on the Internet.

## paramlinks

**pip** sphinx-paramlinks

`paramlinks` allows param links in Sphinx function/method descriptions to be linkable.

## extlinks

**pip**

## traceables

**pip** sphinxcontrib-traceables

`traceables` defines traceable items and relationships between them in documentation generated by Sphinx. It also offers visualization of the traceables in various forms, such as relationship graphs.

## traceability

**pip** sphinxcontrib-traceability

`traceability` adds directives and roles that serve to identify and relate portions of Sphinx documents and create lists and traceability matrices based on them.

## requirements

**pip** sphinxcontrib-requirements

`requirements` Allows declaring requirement specs wherever in the documentation (for instance, in docstrings of `UnitTest.test_*` methods) and displaying them as a single list.

## gen\_node

**pip** sphinxcontrib-gen\_node

gen\_node a generic “todo like” nodes.





---

## Platforms

---

The pages below provide some hints about some platform customization:

### Windows

This page provides some hints about customizing windows.

### Changing the PATH

There are different ways to change the PATH variable on windows and these techniques depends on the version of windows. Please have a look at this note ([web](#)) for instance.

### Ubuntu

This page gives some information about [Ubuntu](#)

### Installation

If you use some editors like LibreOffice you might be interested in installing True Type Font available on Windows. This will bring “Arial” and “Times New Roman” for instance. Otherwise you might have problems when vizualizing document on Windows and Ubuntu.

```
sudo apt-get install ttf-mscorefonts-installer
```



## Le jeu du promoteur

Joue le jeu du promoteur sur <http://lejeudupromoteur.lol>