
Wiser Documentation

Release 0.1

the Nile team

May 20, 2020

1	Table of Contents	3
1.1	Using Forum or Projects	3
1.1.1	Introduction	3
1.1.2	How to...	3
1.2	Using the technical documentation generator	8
1.2.1	Introduction	8
1.2.2	References	8
1.3	Using the version control system	44
1.3.1	Introduction	44
1.3.2	References	44
1.4	Commercial-Off-the-Shelf Software Selection Process	67
1.4.1	Overview	67
1.4.2	Software selection	67
1.4.3	Factors	74
1.4.4	Total Cost of Ownership	76
1.4.5	Additional References	76
1.4.6	Project Management Infrastructure	78
1.4.7	Version Control System	80
1.4.8	Documentation Generator	83
1.4.9	Issue Tracking System	85
1.4.10	Questionnaire Management System	90
1.4.11	Dynamic charts on web browser usage statistics	97
1.4.12	OSGEO list of webmapping tools	97
1.4.13	Web Server and Servlet Container	102
1.4.14	Rationale	102
1.4.15	Analysis of alternatives	103
1.4.16	Operating Systems	104
1.4.17	Database Management System	107
1.4.18	Open Standards in the Portuguese National Regulation on Digital Interoperability	111
1.4.19	Geospatial Data Server	117
1.4.20	Web Mapping	122
1.4.21	Web Browser	125
2	Notebook	131
2.1	Notebook	131
2.1.1	Introduction	131

2.1.2	References	131
2.2	Using the issue tracking system	140
2.2.1	Introduction	140
2.2.2	References	141
Bibliography		147
Index		149

The documents in this section (will) provide generic information about the technical documentation generator (**Sphinx**), the version control system (**Git**), the project management system and issue tracking system (TaskMan *aka* Redmine) and the helpdesk and support ticket system (OTRS).

It may also contain miscellaneous sections on **How-to...** do things in other tools that we may need to use (e.g. Forum).

It's a work-in-progress and it always will...

1.1 Using Forum or Projects

1.1.1 Introduction

This section contains tips on how to best use:

- Forum (<http://forum.eionet.europa.eu/>)
- Projects (<http://projects.eionet.europa.eu/>)

Warning: Don't panic!

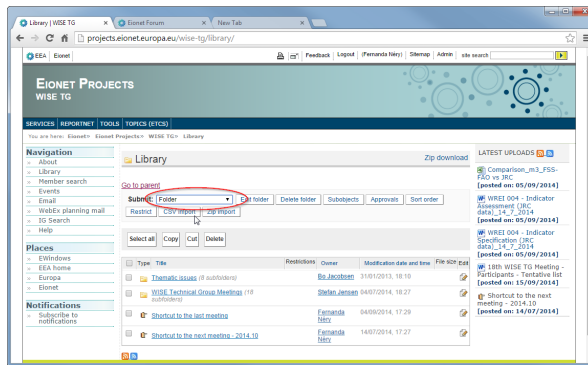
Help is available in <http://forum.eionet.europa.eu/help>

1.1.2 How to...

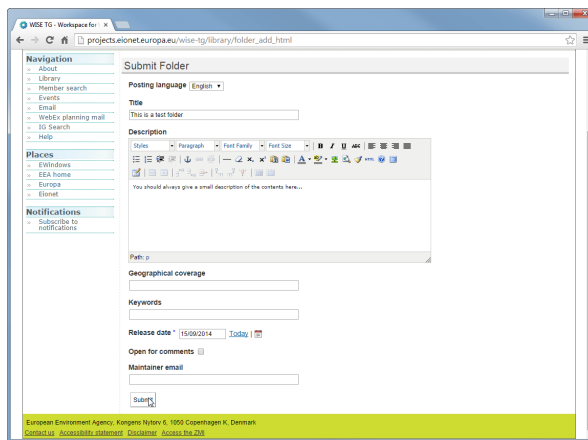
Create a folder

First, read http://forum.eionet.europa.eu/help/content_folder ... Then...

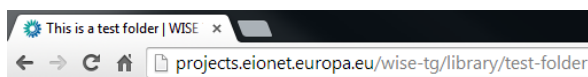
1. Go the library folder
2. Next to the **Submit:** label, select `Folder` in the droplist.



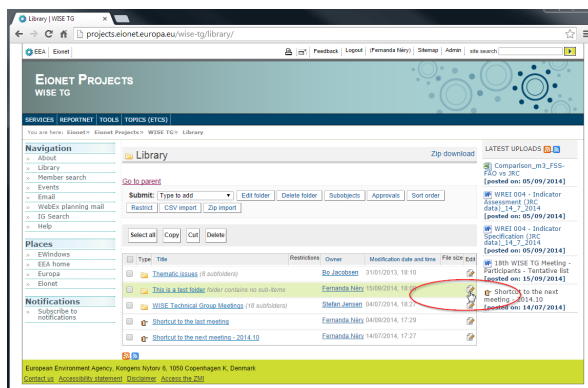
3. Enter the folder's Title and Description, then press Submit



4. Notice the browser's tab title and the URL:

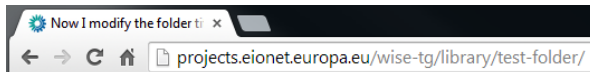


5. You can now edit the Title and Description, if you want:



6. ...and change the title and description.

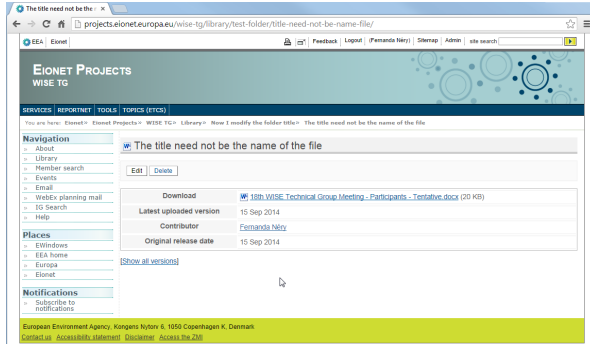
7. Notice that the browser's tab title changed, **but the URL is stable**:



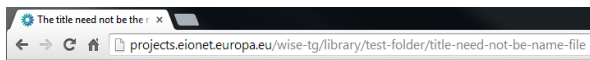
Upload a file (and keep older versions)

First, read http://forum.eionet.europa.eu/help/content_file ... Then...

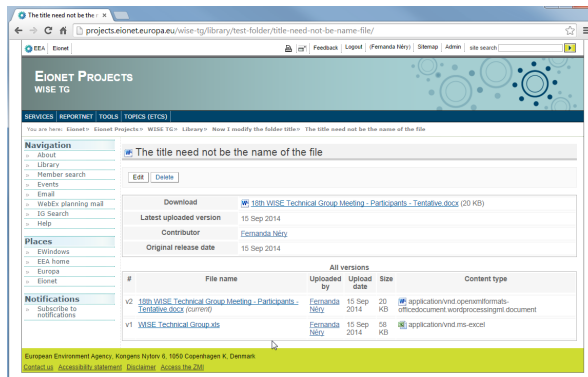
1. Go the library folder
2. Next to the **Submit:** label, select File in the droplist.
3. Enter the file's Title, the Description and press Choose file to select a file in your computer. Then press Submit.
4. Notice that the Title need not be the name of the file:



5. Also, notice the URL that s automatically created:



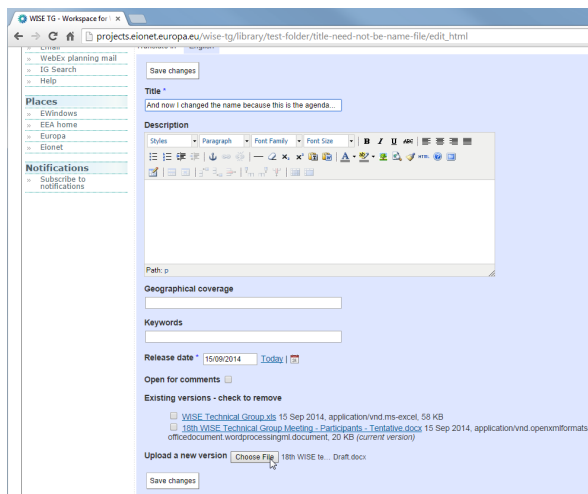
6. You can press Edit and choose another file, if you made a mistake or want to add an updated version of the file:



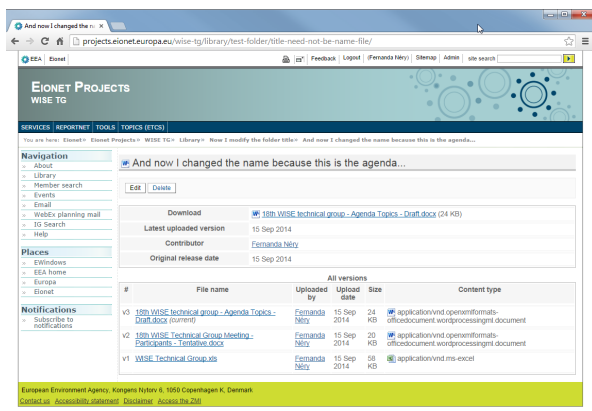
7. And Edit again (notice the two previous versions):



8. Now I modified the Title and added yet another version of the document...

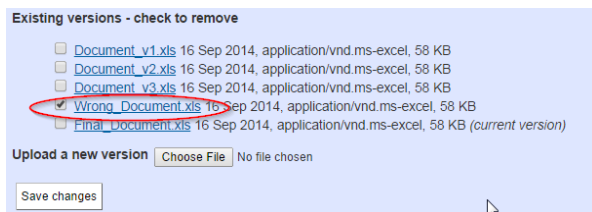


9. Notice that the URL is **always** kept stable:



To remove a previous version

1. Press **Edit** to edit to file.
2. Mark the versions you want to remove:

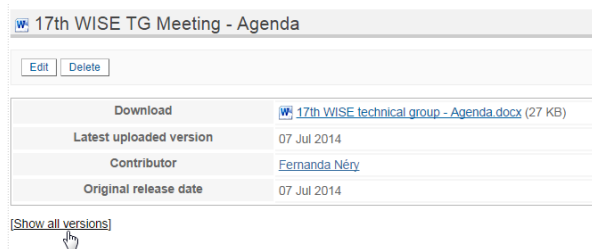


3. Press **Save changes**.
4. The wrong document is gone...




See the previous versions of a file

1. Just press **Show all versions**




2. and there it is:


17th WISE TG Meeting - Agenda

Edit

Delete

Download


17th WISE technical group - Agenda.docx (27 KB)

Latest uploaded version

07 Jul 2014



Contributor

Fernanda Nêky

Original release date

07 Jul 2014

All versions

#	File name	Uploaded by	Upload date	Size	Content type
v2	17th WISE technical group - Agenda.docx (current)	Fernanda Nêky	07 Jul 2014	27 KB	 application/vnd.openxmlformats-officedocument.wordprocessingml.document
v1	17th WISE technical group - Agenda - Final Proposal.docx	Fernanda Nêky	07 Jul 2014	24 KB	 application/vnd.openxmlformats-officedocument.wordprocessingml.document

1.2 Using the technical documentation generator

1.2.1 Introduction

The purpose of this section is to provide ‘quick’ reference information (on syntax, available tools, style conventions, etc) that may be required by technical writers or translators working with the project documentation.

The most important information is about [reStructuredText](#), the markup language used in the technical documentation (user manual, etc.).

There is also some (basic) information about [Sphinx](#), the application that automatically converts the text files and generates the documentation in HTML format or PDF format.

For example, this HTML page was originally written in reStructuredText. Please, press the *Source* link in the menu to see the original text and markup.

1.2.2 References

About Sphinx

[Sphinx](#) is a document generator based on [docutils](#) (an open-source text processing system, for processing plain text documentation into useful formats, such as [HTML](#), [LaTeX](#), [ODT](#) or [XML](#)).

In Sphinx, content (the text) is separated from presentation (formatting):

- Content is stored in plain text files using a simple markup language (reStructuredText).
- Presentation is defined using themes and templates for each type of output (HTML, LaTeX, PDF).

This separation allows the author of the document to focus on the content and not on the presentation or the output format(s). Also, if the documentation needs to be localised to another language, the text can be translated (using translation tools, if required), and all the formatting is applied later (without requiring further changes to the localised documents).

This document provides an introduction to reStructuredText (or reST), the markup language used by Sphinx. The authoritative reference is the [‘reStructuredText User Documentation’](#).

Most of the markup used by Sphinx is standard reST markup. Exceptions (i.e. markup specific to Sphinx) are signalled in the text.

In this document, some (arbitrary) style conventions are proposed to try and keep the project’s documentation as consistent as possible. Again, these conventions are signalled in the text.

Source

This document is an abridged and modified version of *Sphinx’s reStructuredText Primer*.

Sphinx's reStructuredText primer

Introduction

A reStructuredText document is simply a plain text file with some markup to specify the format or the semantics of the text.

There are two types of markup:

- inline markup: for example, `*the surrounding asterisks would mark this text as italics*`, like *this*.
- explicit markup: is used for text that need special handling, such as footnotes, tables, or generic directives. Explicit markup blocks always start with `..` followed by whitespace.

Paragraphs

The paragraph is the basic block in a reST document.

Paragraphs are simply chunks of text separated by one or more blank lines.

Indentation is significant in reST, so all lines of the same paragraph must be left-aligned to the same level of indentation.

This is a style convention.

Try to keep each line with a maximum of 78 characters. Remember that changing to next line does not create a paragraph, unless the chunks of text is separated by a blank line.

Try to keep each phrase in a different line. It improves readability and facilitates the translation process.

Remember that consecutive blank lines will be ignored in the HTML output.

Quoted paragraphs

Quoted paragraphs are created by just indenting them more than the surrounding paragraphs:

Normal paragraph.

Indented paragraph.

This is a style convention.

Each indentation level is created with 3 whitespaces. Do not use tabs.

Line breaks

Line blocks are a way of preserving line breaks (the equivalent of using `Shift+Enter` to break a line in Microsoft Word or LibreOffice Writer):

```
| These lines are  
| broken exactly like in  
| the source file.
```

Sections

Section are created by underlining (and optionally overlining) the section title with a punctuation character:

```
This is a heading  
=====
```

Any punctuation character can be used to define a section title. The underlining (and overlining) must be at least as long as the text itself. Sections must be properly nested.

This is a style convention.

Use the following punctuation characters in the section titles:

- # for Parts
- * for Chapters
- = for sections (“Heading 1”)
- – for subsections (“Heading 2”)
- ^ for subsubsections (“Heading 3”)
- " for paragraphs (“Heading 4”)

Please note that, when converting to HTML format, sections are automatically converted to an appropriate heading tag (for example: `<h2>Heading text</h2>`).

When converting to ODT or DOCX, an appropriate Heading style is applied.

Inline markup

Bold, italics, monospace

The markup is quite simple:

- use one asterisk for italics: `*text*` (the equivalent of using `Ctrl+i` in Microsoft Word or LibreOffice Writer),
- use two asterisks for strong emphasis (boldface): `**text**`
- use backquotes for text literals: ``text``

Be aware of some restrictions:

- The markup may not be nested. For example, this markup is wrong: `*italics with **bold** inside*`
- The text content within the markup may not start or end with whitespace. For example, this markup is wrong: `* text*`
- The markup must be separated from surrounding text by non-word characters (whitespace or punctuation). Use a backslash-escaped-space to work around that. For example: `thisis\ *one*\ word` is rendered like `thisisoneword`.

- If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash.

Subscript and superscript

Subscript is marked with `:sub:`subscript text``. Superscript is marked with `:sup:`superscript text``.

This is a tip.

Whitespace or punctuation is required around interpreted text, but often not desired with subscripts & superscripts. Backslash-escaped whitespace can be used; the whitespace will be removed from the processed document:

```
The chemical formula for molecular oxygen is O\ :sub:`2`.
```

To improve the readability of the text, the use backslash-escapes is discouraged. If possible, use *Substitutions* instead:

```
The chemical formula for pure water is |H2O|.
```

```
.. |H2O| replace:: H\ :sub:`2`\ O
```

Keep all substitutions together (e.g. at the end of the file).

Lists

Bulleted lists

List markup is natural: just place an asterisk at the start of a paragraph and indent properly:

```
* This is a bulleted list.
* It has two items, the second
  item uses two lines.
```

Nested lists are possible, but be aware that they must be separated from the parent list items by blank lines:

```
* this is
* a list

  * with a nested list
  * and some sub-items

* and here the parent list continues
```

Numbered lists

The same goes for numbered lists; they can also be auto-numbered using a # sign:

```
1. This is a numbered list.
2. It has two items too.

#. This is a numbered list.
#. It has two items too.
```

Definition lists

Definition lists are created as follows:

```
term (up to a line of text)
    Definition of the term, which must be indented

    and can even consist of multiple paragraphs

next term
    Description.
```

The [Sphinx](#) documentation generator provides a more flexible alternative to definition lists (see [Glossaries](#)).

Glossaries

The Sphinx `.. glossary::` directive contains a reST definition-list-like markup with terms and definitions.

See the following example:

```
.. glossary::

    environment
        A structure where information about all documents under the root is
        saved, and used for cross-referencing. The environment is pickled
        after the parsing stage, so that successive runs only need to read
        and parse new and changed documents.

    source directory
        The directory which, including its subdirectories, contains all
        source files for one Sphinx project.
```

The definitions will then be used in cross-references with the `:term:` role. For example:

```
The :term:`source directory` for this project is ...
```

In contrast to regular definition lists, a glossary supports *multiple* terms per entry and inline markup is allowed in terms. You can link to all of the terms. For example:

```
.. glossary::

    term 1
    term 2
        Definition of both terms.
```

When the glossary is sorted, the first term determines the sort order.

To automatically sort a glossary, include the following flag:

```
.. glossary::
    :sorted:
```

Field lists

Field lists are two-column table-like structures resembling database records (label & data pairs). For example:


```
:Date: 2001-08-16
:Version: 1
:Authors: - Me
          - Myself
          - I
:Indentation: Since the field marker may be quite long, the second
              and subsequent lines of the field body do not have to line up
              with the first line, but they must be indented relative to the
              field name marker, and they must line up with each other.
:Parameter i: integer
```

This is a style convention.

In this project, field lists are used to include metadata in each document.

The basic ‘**Dublin Core**’_ metadata fields should be included in the very beginning of each document (like a header to the text file), like this:

```
.. metadata-placeholder

:DC.Title:
    Document title
:DC.Creator:
    Author
:DC.Date:
    Date yyyy-mm-dd
:DC.Description:
    Abstract
:DC.Language:
    en
:DC.Format:
    text/x-rst
:DC.Rights:
    Access rights
:DC.RightsHolder:
    Copyright.
```

Note that these metadata field names are not automatically recognised by the Sphinx parser, so the text itself will not be visible in the HTML pages (for example). The metadata fields are the equivalent to the *Document properties* fields in a DOCX file or an ODT file.

`docutils` recognises a number of Bibliographic Fields (such as `docinfo`, `author`, `authors`, `organization`, `contact`, `version`, `status`, `date`, `copyright`, `field`, `topic`).

This is an advanced topic

Some metadata fields are recognised by Sphinx. For example:

- `:tocdepth:` indicates the maximum number of levels in the Sphinx sidebar table of contents for the file.
 - `:orphan:` indicates that, even if the file is not included in any `.. toctree::` directive, no warning should be produced by Sphinx.
-

Tables

The reStructuredText markup supports two basic types of tables. For *grid tables*, you have to “paint” the cell grid yourself. They look like this:

Header row, column 1	Header 2	Header 3	Header 4
(header rows optional)			
body row 1, column 1	column 2	column 3	column 4
body row 2	

Simple tables are easier to write, but limited: they must contain more than one row, and the first column cannot contain multiple lines. They look like this:

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

This is a tip.

These are the basic types of tables, which are rather clumsy. Also available (and easier to use) are *special tables*, namely list-tables and CSV-tables.

An **exceltable** extension can also be used with **Sphinx**, which allows the inclusion of XLS spreadsheets, or part of them, into a reST document.

Hyperlinks

External links

Use ``link text <http://example.com/>`_` for inline web links. If the link text should be the web address, you don’t need special markup at all, the parser finds links and mail addresses in ordinary text (with no markup).

You can also separate the link and the target definition, like this:

```
This is a paragraph that contains `a link`_.

.. _a link: http://example.com/
```

This is a tip.

The use of inline web links is discouraged, to improve the readability of the reST text.

Simple links (e.g. to institutional sites, software sites, and so on) should be kept together at the end of the text file (this is merely a way to simplify the editing procedure, and the update and verification of the links).

Internal links

To support cross-referencing to arbitrary locations in any document, the standard reST labels are used. For this to work, the label names must be unique throughout the entire documentation. There are two ways in which you can refer to labels:

- If you place a label directly before a section title, you can reference to it with `:ref:`label-name``. Example:

```
.. _my-label-ref:

Section to cross-reference
-----

This is the text of the section.

In the end of this phrase is a reference to the section title, see :ref:`my-label-
↪ref`.
```

The `:ref:` role would then generate a link to the section, with the link title being “Section to cross-reference”. This works just as well when section and reference are in different source files.

Automatic labels also work with *figures*:

```
.. _my-figure-ref:

.. figure:: my-image.png

    My figure caption
```

A reference like `:ref:`my-figure-ref`` would insert a reference to the figure with link text “My figure caption”.

The same works for *tables* that are given an explicit caption using the `table` directive.

- Labels that aren’t placed before a section title can still be referenced to, but you must provide the text for the link, using this syntax: `:ref:`Link text <label-name>``.

Using `:ref:` is advised over standard reStructuredText links to sections (like ``Section title`_`) because it works across files, when section headings are changed, and for all builders that support cross-references.

Source Code

Literal code blocks are introduced by ending a paragraph with the special marker `::`. The literal block must be indented (and, like all paragraphs, separated from the surrounding ones by blank lines):

```
This is a normal text paragraph. The next paragraph is a code sample::

    It is not processed in any way, except
    that the indentation is removed.

    It can span multiple lines.

This is a normal text paragraph again.
```

The handling of the `::` marker is smart:

- If it occurs as a paragraph of its own, that paragraph is completely left out of the document.

- If it is preceded by whitespace, the marker is removed.
- If it is preceded by non-whitespace, the marker is replaced by a single colon.

That way, the second sentence in the above example's first paragraph would be rendered as "The next paragraph is a code sample:".

Explicit Markup

Explicit markup is used in reStructuredText for most constructs that need special handling, such as footnotes, specially-highlighted paragraphs, comments, and generic directives.

An explicit markup block begins with a line starting with `..` followed by whitespace and is terminated by the next paragraph at the same level of indentation. (There needs to be a blank line between explicit markup and normal paragraphs. This may all sound a bit complicated, but it is intuitive enough when you write it.)

Directives

A directive is a generic block of explicit markup.

The directive content follows after a blank line and is indented relative to the directive start.

Basically, a directive consists of a name, arguments, options and content.

Look at this example:

```
.. contents:: This is my Table of Contents
   :depth: 2
```

The directive starts with `..` followed by one whitespace. The name of the directive is `contents` (it creates a table of contents). This directive takes one argument: the table of contents' title ("This is my Table of Contents"). The option `depth` specifies the number of section levels that are collected in the table of contents.

Options are given in the lines immediately following the arguments and are indicated by the colons. Options must be indented to the same level as the directive content.

Docutils supports the following directives:

- Admonitions: `attention`, `caution`, `danger`, `error`, `hint`, `important`, `note`, `tip`, `warning` and the generic admonition. (Most themes style only `note` and `warning` specially.)
- Images:
 - `image` - see the *images* section;
 - `figure` - an image with caption and optional legend.
- Additional body elements:
 - `contents <table-of-contents>` - a local table of contents for the sections in the current file only;
 - `rubric` - a heading without relation to the document's sections that won't be included in any table of contents;
 - `topic` and `sidebar` - special highlighted body elements;
 - `epigraph` - a block quote with optional attribution line;
 - `container` - a container with a custom class, useful to generate an outer `<div>` in HTML output.
- *Special tables:*

- `table` - a table with title;
- `csv-table` - a table generated from comma-separated values;
- `list-table` - a table generated from a list of lists.
- Special directives:
 - `include` - include reStructuredText from another file;
 - `raw` - include raw target-format markup, such as LaTeX;
 - `class` - assign a class attribute to the next element.

Table of contents

To include a table of contents within a given document, use the directive `contents`. The following example creates a local table of contents with a maximum of two levels (below the level where it is located):

```
Part II
#####

Chapter 1
*****

.. contents::
   :depth: 2
   :local:

Heading 1
=====
```

The `toctree` directive creates a table of contents that collects information from several files. The following example creates a table of contents from the sections of various documents (up to a depth of 3 levels). The `:glob:` option allows all documents in the 'chapter2' folder to be included (sorted according to their name):

```
.. toctree::
   :glob:
   :maxdepth: 3

   preamble
   chapter1/part1
   chapter1/conclusion
   chapter2/*
   references
```

Note: When building HTML pages from the default template, a `<div class="sphinxsidebar">` is created that holds a 'table of contents' with links to the document sections. The number of levels in the sidebar can be controlled. For example, placing `:tocdepth: 3` in the beginning of the document restricts the number of levels to 3.

Images

reST supports an image directive, used like so:

```
.. image:: gnu.png
   (options)
```

The file name given (here `gnu.png`) must either be relative to the source file, or absolute (which means that they are relative to the top source directory).

For example, the file `sketch/spam.rst` could refer to the image `images/spam.png` as `../images/spam.png` or as `/images/spam.png`.

The image size options (width and height) should be specified in points (`pt`), as that will best support output to different formats (HTML, LaTeX).

Figures

A figure consists of image data (including image options), an optional caption (a single paragraph), and an optional legend (arbitrary body elements):

```
.. figure:: picture.png
   :scale: 50 %
   :alt: map to buried treasure
```

This is the caption of the figure (a simple paragraph).

The legend consists of all elements after the caption. In this case, the legend consists of this paragraph and the following table:

Symbol	Meaning
.. image:: tent.png	Campground
.. image:: waves.png	Lake
.. image:: peak.png	Mountain

There must be blank lines before the caption paragraph and before the legend. To specify a legend without a caption, use an empty comment (`. .`) in place of the caption.

Special tables

The `table` directive associates a title with the following table:

```
.. table:: User list

=====
First name  Last name
=====
John        Doe
Jane        Dove
=====
```

A `list-table` is created from a uniform two-level bullet list:

```

.. list-table:: User list
   :header-rows:1

   * - First name
     - Last name
   * - John
     - Doe
   * - Jane
     - Dove

```

A csv-table is created from comma-separated values (either in the document or in an external file):

```

.. csv-table:: User list
   :header:"First name","Last name"

   "John","Doe"
   "Jane","Dove"

```

Another example of csv-table, using and external file:

```

.. csv-table:: Table 1 - Legend of the table goes here...
   :header-rows: 1
   :stub-columns: 1
   :file: ../tables/table1.csv

```

An exceltable can also be used:

```

.. exceltable:: Table 1 - Legend of the table goes here...
   :file: ../tables/tables.xls
   :sheet: table1
   :selection: A1:C20
   :header: 1

```

Using Excel tables requires an additional module *sphinxcontrib.exceltable* that is an extension for Sphinx, that adds support for including spreadsheets, or part of them, into Sphinx document. It can be installed using pip:

```
pip install sphinxcontrib-exceltable
```

Then the project `conf.py` file needs to be updated:

```
# Add ``sphinxcontrib.exceltable`` into extension list
extensions = ['sphinxcontrib.exceltable']
```

Another alternative is `xmltable` (<https://pythonhosted.org/rusty/xmltable.html>).

Footnotes

For footnotes, use `[#name]_` to mark the footnote location, and add the footnote body at the bottom of the document after a “Footnotes” rubric heading, like so:

```

Lorem ipsum [#first-footnote-name]_ dolor sit amet [#second-footnote-name]_

.. rubric:: Footnotes

.. [#first-footnote-name] Text of the first footnote.
.. [#second-footnote-name] Text of the second footnote.

```

You can also explicitly number the footnotes (`[1]_`) or use auto-numbered footnotes without names (`[#]_`).

This is a tip.

To facilitate editing, auto-numbered footnotes should **not** be used. Instead, use short descriptive names (that simplify cross-referencing).

Citations

Standard reST citations are supported:

```

Lorem ipsum [Ref]_ dolor sit amet.

.. [Ref] Book or article reference, URL or whatever.
```

Citation usage is similar to footnote usage, but with a label that is not numeric or begins with #.

When the documentation is built using the [Sphinx](#) document generator, the citations are “global”, meaning that every citation can be referenced from any .rst files. In this case, a separate file may be created (e.g. a `references.rst` file).

This is a tip.

See *Managing bibliographic citations in Sphinx* for further information.

Substitutions

reST supports “substitutions”, which are pieces of text and/or markup referred to in the text by `|name|`. They are defined like footnotes with explicit markup blocks, like this:

```
.. |name| replace:: replacement *text*
```

or this:

```
.. |caution| image:: warning.png
    :alt: Warning!
```

If you want to use some substitutions for all documents, put them into a separate file (e.g. `substitutions.txt`) and include it into all documents you want to use them in, using the `include` directive.

Be sure to use a file name extension which different from that of other source files, to avoid Sphinx finding it as a standalone document. For example, use the `.rst` file extension for the source files, and the `.txt` file extension for the files which are to be included.

This is a tip.

This is useful in technical documentation such as User’s Manuals, where a substitution file can be built for each localised version of the interface elements (menus, messages, etc), guaranteeing the consistency of the document translation with the software’s human user interface.

Warning.

Substitutions do NOT work inside directives (or inside the options of a directive).

Do not try to google for a solution (...been there). It is a design limitation: RST markup can not be nested. Period.

Comments

Every explicit markup block which isn't a valid markup construct is regarded as a comment. For example:

```
.. This is a comment.
```

You can indent text after a comment start to form multiline comments:

```
..
    This whole indented block
    is a comment.

    Still in the comment.
```

This is a style convention.

Comments can also be used as placeholders to mark places within the document. For example:

- the `.. links-placeholder` can mark the place where hyperlinks are kept together at the end of the document;
 - the `.. metadata-placeholder` can mark the place where document metadata (author, date, etc) is kept together at the beginning of the document.
-

Tools for reStructuredText

Introduction

This document presents some of the tools available for working with reStructuredText.

It is divided in the following parts:

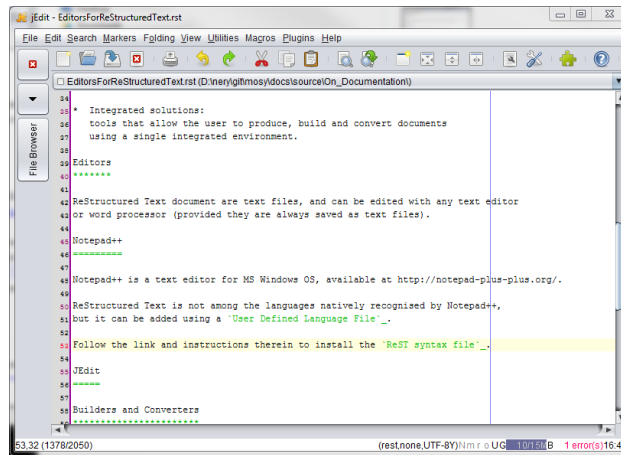
- Editors: simple text editors providing syntax highlight for reST documents.
- Builder and Converters: tools that automatically convert reST documents to other formats such as HTML, PDF, DOC, ODT, etc.
- Integrated solutions: tools that allow the user to produce, build and convert documents using a single integrated environment.

Editors

reStructuredText documents are text files, and can be edited with any text editor or word processor (provided they are always saved as text files).

JEdit

jEdit is a FOSS text editor, written in Java (so it runs in Windows, Mac OS X, Linux, etc.). ReST is among the 211 languages supported natively by jEdit.



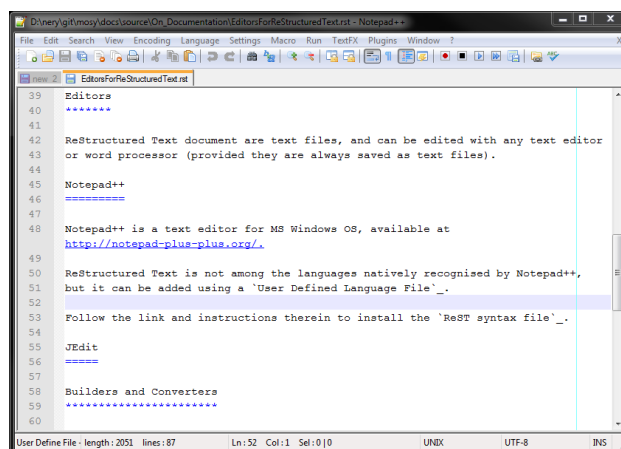
Notepad++

Notepad++ is a FOSS text editor for **MS Windows OS** only.

reStructuredText is not among the languages natively recognised by Notepad++, but it can be added using a **‘User Defined Language File’** (see install instructions below the list of available language files).

Follow the link to download the **‘ReST syntax file’**.

Notepad++ is simpler and more user friendly than jEdit.

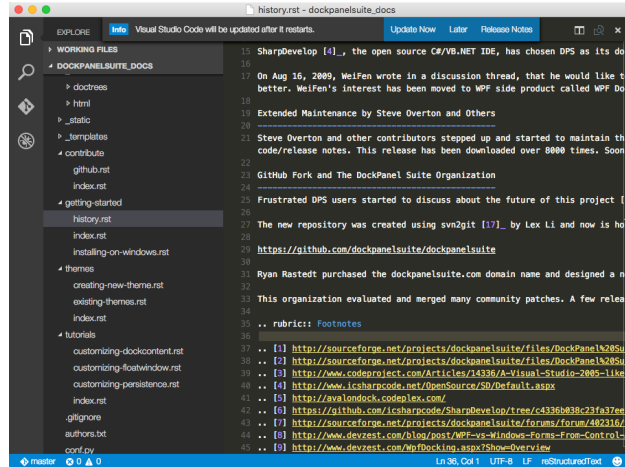


ReText

ReText is a simple editor that reads your text with Markdown or HTML markup and saves it as plain text, HTML or PDF. It is written in Python using Qt libraries.

Visual Studio Code

Visual Studio Code is a FOSS text editor, written in TypeScript (so it runs in Windows, Mac OS X, Linux, etc.). ReST is not among the languages natively supported by Visual Studio Code, but it can be added using an [extension from LeXtudio](#).



Builders and converters

Todo: Section on Builders and Converters such as Sphinx and Pandoc.

Sphinx

Sphinx is a Python documentation generator.

It requires [Python](#), which is installed by default in Linux and Mac OS X systems. For Microsoft Windows systems, see [Installing Python on Windows](#) if you need help installing Python and two useful installation utilities (*easy_install* and *pip*).

After you have Python installed, simply use the following command (in a command window):

```
easy_install -U Sphinx
```

Elevated privileges (i.e. administration rights) should not be required.

The Sphinx builder can produce a number of output formats (e.g. HTML, PDF). PDF files can be produced using the LaTeX builder (more complicated) or using the a direct PDF builder called rst2pdf (see below).

Rst2Pdf

rst2pdf is a tool for transforming reStructuredText to PDF using ReportLab. To install rst2pdf on Windows you also need [Python](#) because rst2pdf is coded in python.

Rst2pdf uses [ReportLab](#), which can be installed using:

```
easy_install reportlab
```

Again, in Windows, there may be a problem with the required Microsoft Visual Studio version. While running `setup.py` for package installations, Python 2.7 searches for an installed Visual Studio 2008. The solution is to define `VS90COMNTOOLS` variable to point to Tools directory of Visual Studio:

```
SET VS90COMNTOOLS=%VS100COMNTOOLS%
```

How to install rst2pdf on Windows?

1. Download rst2pdf source from <https://code.google.com/p/rst2pdf/downloads/list>
2. Unzip the file to an rst2pdf folder.
3. Goto the the rst2pdf folder which contains `setup.py` file.
4. Run `python setup.py install` command and it will be installed.
5. To convert any .rst file to PDF file Run `rst2pdf myfile.rst` command and you are done.
 - <http://rst2pdf.ralsina.me/>
 - User Manual: <http://ralsina.me/static/manual.pdf>

Pandoc

Integrated solutions

ReST Editor for Eclipse

The *ReST editor for Eclipse* is a plug-in for the Eclipse IDE. If `Sphinx` is installed, it can also be used to create (and build) Sphinx projects from within Eclipse. The following [presentation](#) documents the use of the editor.

This ReST editor has several advantages, namely:

- integrated spell-checking using Hunspell4Eclipse
- contextual ReST syntax help
- sections outline rearrangement

Publishing the documents online

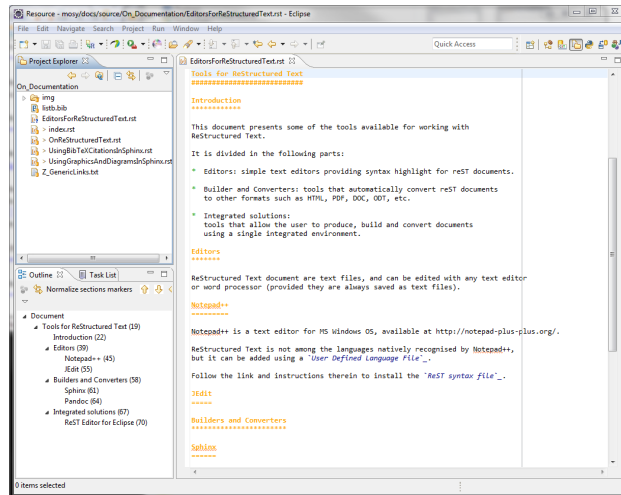
Introduction

`Sphinx` documentation can be hosted online.

Dropbox

Google drive

Note: Sources

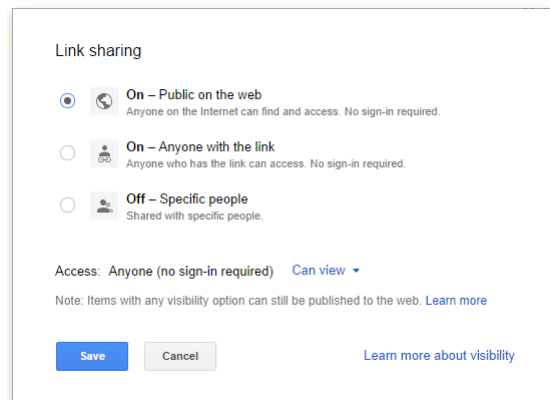


<https://kb.wisc.edu/helpdesk/page.php?id=38083>

<https://support.google.com/drive/answer/2881970?hl=en>

<https://developers.google.com/drive/web/publish-site>

1. Create a folder in Google Drive and share it as Public on the Web



2. Upload the folder containing the HTML build, Javascript, and CSS files to this folder.
3. Find the shareable link of the HTML folder and copy the unique identifier

For example, if the link is `https://drive.google.com/folderview?id=0B8AmLQ1728LmeHpwbEd1N0U4YTQ&usp=sharing` then the unique id is `0B8AmLQ1728LmeHpwbEd1N0U4YTQ`.

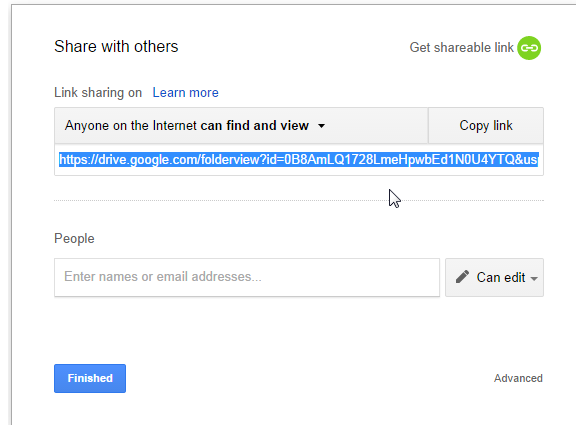
4. So the link to your documentation will be

`www.googledrive.com/host/the-unique-identifier-that-you-just-copied/index.html`

For example, the link to these pages is:

`www.googledrive.com/host/0B8AmLQ1728LmeHpwbEd1N0U4YTQ/index.html`

Read The Docs



Github.io

Note: Sources

<http://lucasbardella.com/blog/2010/02/hosting-your-sphinx-docs-in-github>

<http://daler.github.io/sphinxdoc-test/index.html>

Turning Jekyll off

Note: Source

<https://help.github.com/articles/using-jekyll-with-pages#turning-jekyll-off>

You can completely opt out of Jekyll processing by creating a file named `.nojekyll` in the root of your Page repository and pushing that file to GitHub.

This should only be necessary if your site uses directories that begin with an underscore, as Jekyll sees these as special directories and does not copy them to the final destination.

Since [Sphinx](#) puts all the static files in a `__static` folder, this needs to be done, otherwise the stylesheets, etc, won't be uploaded to the html site.

Showing source code examples in Sphinx

Standard reST literal blocks are started by `:` at the end of the preceding paragraph and delimited by indentation.

Highlight directive

The default highlighting language is Python: it can be changed using the `highlight` directive within a document:

```
.. highlight:: html
```

The literal blocks are now highlighted as HTML, until a new directive is found.

(continues on next page)

(continued from previous page)

```

::
  <html><head></head>
  <body>This is a text.</body>
</html>

```

The following directive changes the highlight language to SQL.

```

.. highlight:: sql

::
  SELECT * FROM mytable

.. highlight:: none

```

From here on no highlighting will be done.

```

::
  SELECT * FROM mytable

```

Code-block directive

The code-block directive can be used to declare the specific language to be used in a block, regardless of the highlighting language:

The following is a SQL statement.

```

.. code-block:: sql
   :linenos:

  SELECT * FROM mytable

```

Line numbers are useful for long blocks such as this one:

```

1  -- http://www.postgresonline.com/journal/index.php?/archives/97-SQL-Coding-Standards-
   ↪To-Each-His-Own-Part-II.html
2  SELECT persons.id, persons.first_name, persons.last_name, forums.category,
3         COUNT(DISTINCT posts.id) AS num_posts,
4         COALESCE(MAX(comments.rating), 0) AS highest_rating,
5         COALESCE(MIN(comments.rating), 0) AS lowest_rating
6  FROM persons JOIN posts ON persons.id = posts.author
7         JOIN forums on posts.forum = forums.id
8         LEFT OUTER JOIN comments ON posts.id = comments.post
9  WHERE persons.status > 0
10         AND forums.ratings = TRUE
11         AND comments.post_date > ( now() - INTERVAL '1 year')
12  GROUP BY persons.id, persons.first_name, persons.last_name, forums.category
13  HAVING count(DISTINCT posts.id) > 0
14  ORDER BY persons.last_name, persons.first_name;

```

Literalinclude directive

Another option is to include part of a given source code file, like this:

```
.. literalinclude:: filename
   :linenos:
   :language:
   :lines:
   :start-after:
   :end-before:
   :emphasize-lines:
```

Just below is a example:

```
Literalinclude directive
*****
```

Another option is to include part of a given source code file, like this::

Instead of using line numbers (which can change), it is possible to use the options `:start-after` and `:end-before:` that search the included file for lines containing the specified text. For example:

```
.. literalinclude:: ShowingCodeExamplesInSphinx.rst
   :language: rst
   :start-after: Instead of using
   :end-before: For example
```

produces this result:

```
use the options ``:start-after`` and ``:end-before:``
that search the included file for lines containing the specified text.
```

Pygments lexers

Syntax highlighting is done by [Pygments](#) (if installed): any of the **‘Pygments language lexers’** can be used.

The following table lists some useful lexers (in no particular order).

Lexer	Shortname
Structured Query Language	sql
PostgreSQL dialect of SQL	postgresql
PostgreSQL procedural language	plpgsql
PostgreSQL console sessions	psql
ReStructured Text	rst
TeX and LaTeX	latex
DOS/Windows batch file	bat
Windows PowerShell	powershell
Bash shell scripts	bash
Bash shell sessions	console
Cascading Style Sheets	css
HTML 4 and XHTML 1	html
XML	xml
XSLT	xslt
XQuery	xquery
JavaScript	javascript
JSON data structures	json
PHP source code	php

Continued on next page

Table 1 – continued from previous page

Lexer	Shortname
PHP embedded in HTML	html+php
Python 2	python
Python 2 tracebacks	pytb
Python console	pycon
Java	java
Configuration file in the Java's properties format	jproperties
Configuration file in the Apache config file format	apacheconf
R source code (or S, or S-plus)	r
R console	rout
Matlab	matlab
Matlab sessions	matlabsession
NumPy	numpy

Creating diagrams in Sphinx

Using Graphviz

Besides using raster images (PNG, JPG, etc.), diagrams can be included with the `'sphinx.ext.graphviz'` extension.

Graphviz is an open source graph visualisation software. Graph visualisation is a way of representing structural information as diagrams of abstract graphs and networks.

It must be installed before the extension can be used.

Due to current (2013.10) compatibility issues with PlantUML, it may be preferable to install GraphViz 2.28 instead.

Including the extension in the project configuration file

The Graphviz extension is included with Sphinx, but the extension must be enabled in the `conf.py` file:

```
extensions = ['sphinx.ext.graphviz']
```

Changing the configuration file

Extensions local to a project should be put within the project's directory structure. Set Python's module search path, `sys.path`, accordingly so that Sphinx can find them. E.g., if your extension `foo.py` lies in the `exts` subdirectory of the project root, put into `conf.py`:

```
import sys, os
sys.path.append(os.path.abspath('exts'))
extensions = ['foo']
```

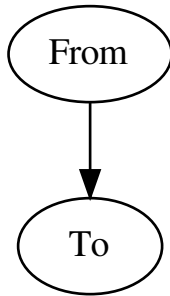
You can also install extensions anywhere else on `sys.path`, e.g. in the site-packages directory.

Examples

This code:

```
.. graphviz::  
  
    digraph {  
        "From" -> "To";  
    }
```

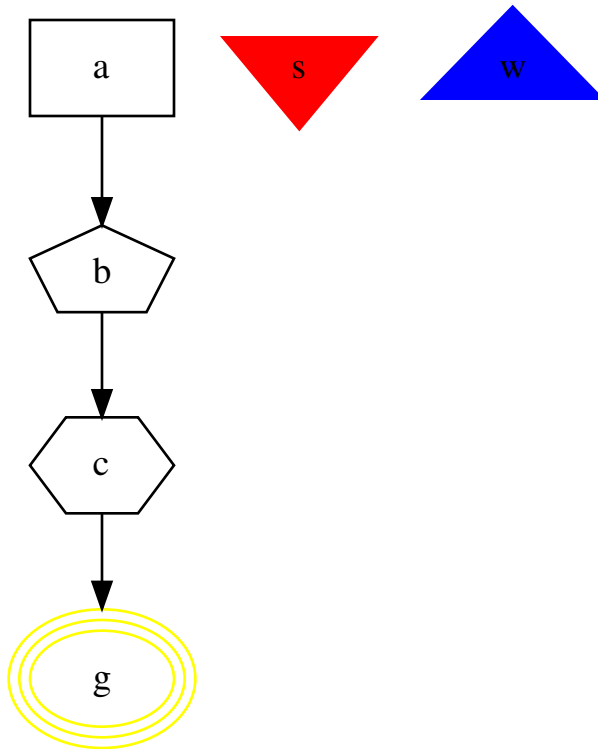
has this result:



This code:

```
.. graphviz::  
  
    digraph Flatland {  
  
        a -> b -> c -> g;  
        a  [shape=polygon,sides=4]  
        b  [shape=polygon,sides=5]  
        c  [shape=polygon,sides=6]  
  
        g [peripheries=3,color=yellow];  
        s [shape=invtriangle,peripheries=1,color=red,style=filled];  
        w [shape=triangle,peripheries=1,color=blue,style=filled];  
  
    }
```

has this result:



Numerous examples are available online:

- [https://en.wikipedia.org/wiki/DOT_\(graph_description_language\)](https://en.wikipedia.org/wiki/DOT_(graph_description_language))
- <http://www.graphviz.org/pdf/dotguide.pdf>
- <http://graphs.grevian.org/example.html>

Using PlantUML

The **‘Sphinx PlantUML extension’** (in this case a contributed extension) allows Sphinx to embed UML diagrams by using PlantUML.

PlantUML is a Java component that allows to quickly write simple UML diagrams:

- use case diagrams,
- class diagrams,
- activity diagrams,
- state diagrams,
- component diagrams,
- sequence diagrams,
- object diagram.

Diagrams are defined using a simple and intuitive language. This can be used within many other tools. Images can be generated in PNG or SVG format.

Installing the extension

The module is installed with the following command:

```
pip install sphinxcontrib-plantuml
```

Including the extension in the project configuration file

The extension must be enabled in the `conf.py` file:

```
extensions = ['sphinxcontrib.plantuml']
```

The path to the PlantUML file may have to be specified (assuming that Java itself is already in the search path):

```
plantuml = 'java -jar ../utils/plantuml.jar'
```

PlantUML requires Graphviz and an environment variable may have to be defined, pointing to the `dot` executable. For example (in Linux or OS-X):

```
setenv GRAPHVIZ_DOT /usr/local/bin/graphviz/dot
export GRAPHVIZ_DOT
```

Note: For Ubuntu users

Files with the `.sh` extension in the `/etc/profile.d` directory get executed whenever a bash login shell is entered (e.g. when logging in from the console or over ssh), as well as by the DisplayManager when the desktop session loads:

```
/etc/profile.d/*.sh
```

You can for instance create the file `/etc/profile.d/myenvvars.sh` and set variables like this:

```
export GRAPHVIZ_DOT=/usr/bin/dot
```

Note: For Windows users

Regardless of the existence of the `GRAPHVIZ_DOT` environment variable, the path to the Graphviz bin folder is apparently required to be in the `PATH` variable as well.

Examples

In the Sphinx reST documents, simply begin the PlantUML code with the `uml` directive.

This is the code for the example above:

```
.. uml::

    @startuml
    user -> (use PlantUML)

    note left of user
        Hello!
    end note
    @enduml
```

Another example:

This is the code for the example above:

```
.. uml::

    @startuml
    Alice -> Bob: Hi!
    Alice <- Bob: How are you?
    @enduml
```

Class diagram

Diagram

This is the diagram generated by the Sphinx PlantUML extension.

Code

This is the code for example above.

```
.. uml::

    @startuml

    'style options
    skinparam monochrome true
    skinparam circledCharacterRadius 0
    skinparam circledCharacterFontSize 0
    skinparam classAttributeIconSize 0
    hide empty members

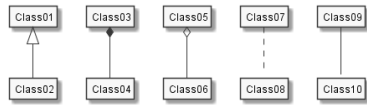
    Class01 <|-- Class02
    Class03 *-- Class04
    Class05 o-- Class06
    Class07 .. Class08
    Class09 -- Class10

    @enduml
```

Note that in docutils / Sphinx, `@startuml` and `@enduml` could be omitted. However, it is useful to keep these lines: is necessary, PlantUML can be used (outside Sphinx) to generate the PNG image files with the diagrams directly from the text file; also, if editing the code in Eclipse, the PlantUML diagrams can be previewed without the necessity of building the documentation.

Image

The image was exported using the Eclipse PlantUML plug-in. It is static, but can be resized. . .



Other examples

```

.. uml::

    @startuml

    'style options
    skinparam monochrome true
    skinparam circledCharacterRadius 0
    skinparam circledCharacterFontSize 0
    skinparam classAttributeIconSize 0
    hide empty members

    class Car

    Driver - Car : drives >
    Car *- Wheel : have 4 >
    Car -- Person : < owns

    @enduml
    
```

To declare fields and methods, you can use the symbol “:” followed by the field’s or method’s name. The system checks for parenthesis to choose between methods and fields.

```

.. uml::

    @startuml

    'style options
    skinparam monochrome true
    skinparam circledCharacterRadius 9
    skinparam circledCharacterFontSize 8
    skinparam classAttributeIconSize 0
    hide empty members

    abstract class AbstractClass {
        - privateField
        + publicField
        # protectedField
        ~ packagePrivateField
        - privateMethod()
        + publicMethod()
        # protectedMethod()
        ~ packagePrivateMethod()
    }

    class Dummy {
    
```

(continues on next page)

(continued from previous page)

```

    {static} staticID
    {abstract} void methods()
    }

class Flight {
    flightNumber : Integer
    departureTime : Date
}

package "Classic Collections" {

    abstract class AbstractList
    abstract AbstractCollection
    interface List
    interface Collection

    List <|-- AbstractList
    Collection <|-- AbstractCollection

    Collection <|-- List
    AbstractCollection <|-- AbstractList
    AbstractList <|-- ArrayList

    class ArrayList {
        Object[] elementData
        size()
    }
}

enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}

class Student {
    Name
}
Student "0..*" -- "1..*" Course
(Student, Course) .. Enrollment

class Enrollment {
    drop()
    cancel()
}

@enduml

```

Use case diagram

Code

Diagram

This is generated by the Sphinx PlantUML extension.

Code

```
.. uml::

    @startuml
    actor "Main Database" as DB << Application >>

    note left of DB
        This actor
        has a "name with spaces",
        an alias
        and a stereotype
    end note

    actor User << Human >>
    actor SpecialisedUser
    actor Administrator

    User <|--- SpecialisedUser
    User <|--- Administrator

    usecase (Use the application) as (Use) << Main >>
    usecase (Configure the application) as (Config)
    Use ..> Config : <<includes>>

    User --> Use
    DB --> Use

    Administrator --> Config

    note "This note applies to\nboth actors." as MyNote
    MyNote .. Administrator
    MyNote .. SpecialisedUser

    ' this is a text comment and won't be displayed
    AnotherActor ---> (AnotherUseCase)

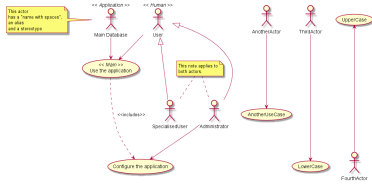
    ' to increase the length of the edges, just add extras dashes, like this:
    ThirdActor ----> (LowerCase)

    ' The direction of the edge can also be reversed, like this:
    (UpperCase) <---- FourthActor

    @enduml
```

Image

The image was exported using the Eclipse PlantUML plug-in. It is static, but can be resized...



Activity diagram (new syntax)

Diagram

Code

There are two syntaxes to create activity diagrams. The example utilises the new syntax (which is still incomplete).

```
.. uml::

@startuml

start

:first activity;

:second activity
  with a multiline
  and rather long description;

:another activity;

note right
  After this activity,
  are two 'if-then-else' examples.
end note

if (do optional activity?) then (yes)
  :optional activity;
else (no)

  if (want to exit?) then (right now!)
    stop
  else (not really)

  endif

endif

endif

:third activity;

note left
  After this activity,
  parallel activities will occur.
end note

fork
  :Concurrent activity A;
```

(continues on next page)

(continued from previous page)

```

fork again
  :Concurrent activity B1;
  :Concurrent activity B2;
fork again
  :Concurrent activity C;
  fork
    :Nested C1;
    fork again
      :Nested C2;
    end fork
  end fork
end fork

repeat
  :repetitive activity;
repeat while (again?)

while (continue?) is (yes, of course)
  :first activity inside the while loop;
  :second activity inside the while loop;
endwhile (no)

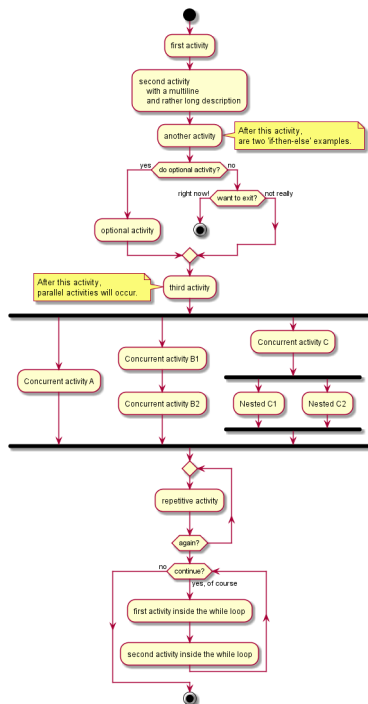
stop

@enduml

```

Image

The image was exported using the Eclipse PlantUML plug-in. It is static, but can be resized. . .



State diagram

Diagram

Generated by the Sphinx PlantUML extension.

Code

```
.. uml::

    @startuml

    [*] --> MyState
    MyState --> CompositeState
    MyState --> AnotherCompositeState
    MyState --> WrongState

    CompositeState --> CompositeState : \ this is a loop
    AnotherCompositeState --> [*]
    CompositeState --> [*]

    MyState : this is a string
    MyState : this is another string

    state CompositeState {

        [*] --> StateA : begin something
        StateA --> StateB : from A to B
        StateB --> StateA : from B back to A
        StateB --> [*] : end it

        CompositeState : yet another string
    }

    state AnotherCompositeState {

        [*] --> ConcurrentStateA
        ConcurrentStateA --> ConcurrentStateA

        --

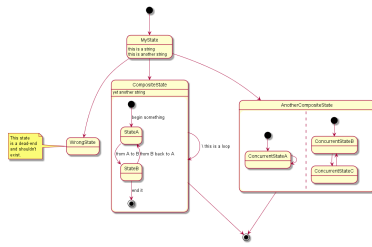
        [*] --> ConcurrentStateB
        ConcurrentStateB --> ConcurrentStateC
        ConcurrentStateC --> ConcurrentStateB
    }

    note left of WrongState
        This state
        is a dead-end
        and shouldn't
        exist.
    end note

    @enduml
```

Image

The image was exported using the Eclipse PlantUML plug-in. It is static, but can be resized. . .



GUI mockups

PlantUML can also be used for GUI mockups (see <http://plantuml.sourceforge.net/salt.html>).

Example

Code

```

.. uml::

    @startuml
    salt
    {
        Just plain text
        [This is my button]
        () Unchecked radio
        (X) Checked radio
        [] Unchecked box
        [X] Checked box
        "Enter text here  "
        ^This is a droplist^
    }
    @enduml
    
```

Managing bibliographic citations in Sphinx

Introduction

reStructuredText *Citations* are ill-adapted to parenthetical referencing (a.k.a. the ‘**Harvard System of Referencing**’ _).

An alternative is to (manually) use the ‘**authorship trigraph**’_ (common in older computer science texts).

The citation begins with 4 letters:

- one author: first 4 letters of name
- two authors: first 2 letters of author1, first 2 letters of author2
- three authors: first 2 letters of author1, first letter of author2, first letter of author3
- four authors: first letter of each author

- more than four authors: first letter of first four authors

The first letter of a name is always upper case.

After the authors' initials, put the two digits of the year (century-disambiguation is ignored).

If the symbol is exactly the same for two references, a lower case letter is attached.

To facilitate editing, citation text should be kept at the bottom of the document after a “References” rubric heading, like this:

```
.. rubric:: References

.. [BiDB79] Biskup, J.; Dayal, U.; Bernstein, P.A.: Synthesizing independent_
↪database schemas. In: ACM SIGMOD 1979 Int. Conf. On Management of Data Proceedings, ↪
↪S. 143-151.

.. [BeBe79a] Beeri, C.; Bernstein, P.A.: Computational problems related to the design_
↪of normal relational schemas. ACM Trans. Database Syst., No. 1, 1979, S. 30-59.

.. [BeBe79b] Beeri, C.; Bernstein, P.A.: Computers are stupid. ACM Trans. Database_
↪Syst., No. 4, 1979, S. 253-266.
```

A similar option is to use the BibTeX alpha style:

- one author: first 3 letters of the last name
- two to four authors: first letters of last names concatenated
- more than four authors: first letters of last names of first three authors concatenated and a “+” sign at the end.

For the examples above, the alpha style citation would be: [BDB79], [BB79a] and [BB79b], respectively.

Using Sphinx BibTeX extension

Parenthetical referencing can be produced in Sphinx using the `sphinxcontrib-bibtex` extension.

The `sphinxcontrib-bibtex` extension allows BibTeX citations to be inserted into documentation generated by Sphinx.

The extension defines a new `bibliography` directive and a new `cite` role.

These work similarly to the LaTeX's `thebibliography` environment and `\cite` command.

The references are stored in a separate plain text BibTeX format file. Currently, only the `unsrt` and `plain` BibTeX styles are supported.

Please note that the current `sphinxcontrib-bibtex` is a **beta** version.

Installing the extension

The module is installed with:

```
pip install sphinxcontrib-bibtex
```

This is a tip.

For Windows users. To facilitate the installation of 3rd party Python packages, follow the instructions on how to ‘**add Distribute and Pip to the Python installation**’.

Including the extension in the project configuration file

The Sphinx project `conf.py` file must be altered to include:

```
extensions = ['sphinxcontrib.bibtex']
```

Example

In the document, use the following syntax:

```
See :cite:`Strunk1979` for an introduction to stylish blah, blah...
```

And place the directive at the end of the document:

```
.. bibliography:: references.bib
```

The `references.bib` file should contain a **BibTeX** bibliography, including an entry for:

```
@BOOK{Strunk1979,
  title = {The Elements of Style},
  publisher = {Macmillan},
  year = {1979},
  author = {Strunk, Jr., William and E. B. White},
  edition = {Third}
}
```

Using the Sphinx Natbib Extension

A more flexible alternative is to use <http://wnielson.bitbucket.org/projects/sphinx-natbib/>

This documentation can be completed iff required in this specific project.

Using the Sphinx Thesis Resource

See also <http://jterrace.github.io/sphinxtr/html/ch-intro/index.html> for various useful adaptations/extensions of Sphinx.

Using LaTeX directly in Sphinx

For advanced users, LaTeX can also be used directly in Sphinx (when only LaTeX output is required):

```
See :raw-tex:`\cite{Strunk1979}` for an introduction to stylish blah, blah...
```

And insert the bibliography at the end of the document:

```
.. raw:: latex

\bibliographystyle{plain}
\bibliography{listb.bib}
```

Managing BibTeX bibliographies

The BibTeX files can be easily managed with [JabRef](#).

JabRef is an open source bibliography reference manager. The native file format used by JabRef is BibTeX, the standard LaTeX bibliography format.

JabRef runs on the Java (version 1.6 or newer), and should work equally well on Windows, Linux and Mac OS X.

Creating equations in Sphinx

LaTeX

The syntax for writing equations is LaTeX.

Only brief examples are included here, since LaTeX has a rather steep learning curve, and AMS LaTeX is only concerned with math support.

The following links are useful:

- See <ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf> for a not-so-short but clear syntax guide.
- See <http://www.ams.org/publications/authors/tex/amslatex> for complete references.
- See <http://mirrors.fe.up.pt/pub/CTAN/info/examples/mil/mil.pdf> for a not-so-gentle introduction to LaTeX.

MathJax

In Sphinx, the rendering (display) of the equations can be done in different ways, that will not be discussed here.

The selected option is to use the `sphinx.ext.mathjax` extension. This extension uses the JavaScript package *MathJax* to transform the LaTeX markup to readable math live in the browser.

The disadvantages are the (large) size and load time of the MathJax library.

The `mathjax_path` in the `conf.py` file indicates where the MathJax library resides. By default, this is the MathJax site, but the path can be changed no cross-site scripting is allowed.

Equation editors or previewers

Given that LaTeX syntax may be daunting, a WYSIWYG math editor can be useful, or at least an interactive previewer:

- If the objective is simply to preview the result, the online **‘Interactive LaTeX Editor’** is a very good option and includes numerous equations as examples.
- **LyX** is a user-friendly LaTeX processor that includes an equation editor.

In the long run, LyX may be the best choice: it has the same dependencies as EqualX, a larger development and user community, and does not require virtually any LaTeX knowledge.

- **EqualX** is a LaTeX equation editor (not a document processor as LyX): it can be used to create the equations and then paste the code into the ReST document.

Like **LyX**, EqualX requires a LaTeX distribution (in Linux, the dependencies are automatically installed and **TeXLive** is included in the official repositories of all major distributions; for Windows systems, **MiKTeX** is a possible alternative).

Examples

See additional examples at <http://sphinx-doc.org/ext/math.html>.

Code:

```
If :math:`\sigma_1` equals :math:`\sigma_2` then etc, etc.
```

Output:

If σ_1 equals σ_2 then etc, etc.

Code:

```
:math:`\underline{x}=[x_1, \dots, x_n]^T`
```

Output:

$\underline{x} = [x_1, \dots, x_n]^T$

Code:

```
\langle \alpha, \beta \rangle
\in
\Bigl \{
M, \text{ if }
{
l(\underline{x}) =
\frac { p(\underline{x}|M) } { p(\underline{x}|U) } \geq \frac { p(U) } { p(M) }
\geq
\frac { p(U) } { p(M) }
}
\atop
U, \text{ otherwise }
}
```

Output:

$$\langle \alpha, \beta \rangle \in \begin{cases} M, & \text{if } l(\underline{x}) = \frac{p(\underline{x}|M)}{p(\underline{x}|U)} \geq \frac{p(U)}{p(M)} \\ U, & \text{otherwise} \end{cases}$$

1.3 Using the version control system

1.3.1 Introduction

This section contains information about the main concepts of version control and basic information on how to use [Git](#), the version control system selected for source-code and technical documentation management.

Currently, this section is rather technical and unpalatable. That will change...

1.3.2 References

On Version Control

Introduction

Version Control is the management of changes to documents, computer programs, large web sites, and other collections of information. The set of files under version control is kept in a *repository*.

The **Version Control System** (VCS) is the application responsible for keeping track of the successive versions of a repository.

The basic workflow is:

1. A user clones the repository and creates a local *working copy* of the files. (The local copy can itself be a local repository under version control).
2. The user works on the local copy of the files. (Note that not every change to every file is registered as a *revision*).
3. When the user wants to (e.g. when a new version of a document is quite complete, or every day at 6:14pm ...), a group of new or modified files (or *changeset*) can be committed as a revision to the local working copy.
4. The user can also choose when to *commit* the revisions back into the original repository.
5. If the local revisions are merged into the original repository, a new revision point is created therein.

The *merge* can be performed automatically by the version control system:

- a. if the user has the required permissions to commit (*push*) to the original repository;
 - b. and if no *conflict* is detected (e.g. while the user was working in the local copy, someone else committed a revision to same files).
6. If the user does not have the required permissions to commit to the original repository, a *pull* request can be sent to the owner of the repository.

The owner reviews the proposed changes, and accepts or rejects the changeset.

The following storyboard illustrates this steps.

Storyboard #1 - The basic workflow

This storyboard depicts a simplified workflow, using a simplified hierarchy. (In real-world use, links can exist between any two different actors and repositories. The technology allows networks with arbitrary configuration.)

Network of actors

Consider a typical hierarchy:

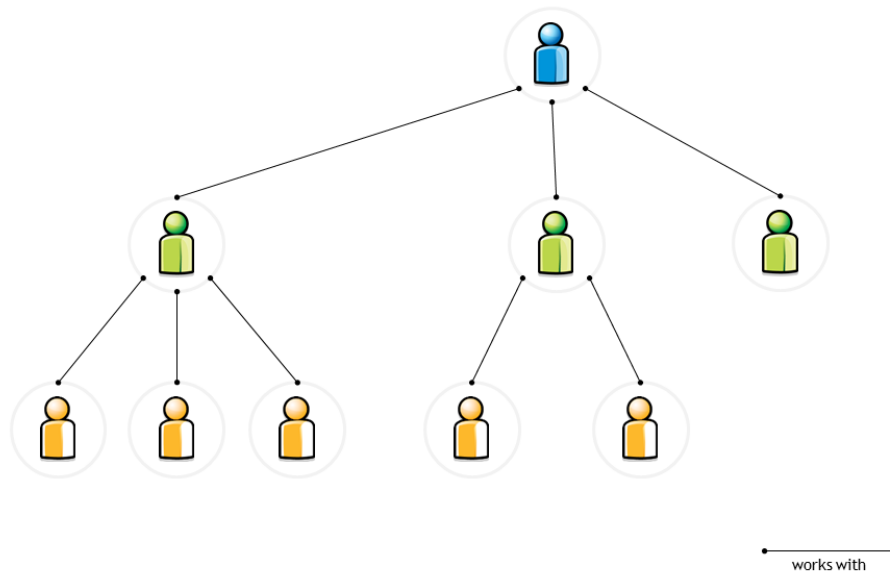
- The ‘blue’ actor only works and knows the ‘green’ actors.
- Each ‘green’ actor works with a distinct group of ‘yellow’ actors.
- The blue actor’s responsibility is to collate the green actors’ contributions (changes) and to resolve any conflicts between different changes proposed by distinct green actors.
- Each green actor’s responsibility is similar, with regard to the yellow actors.

Networks of trust

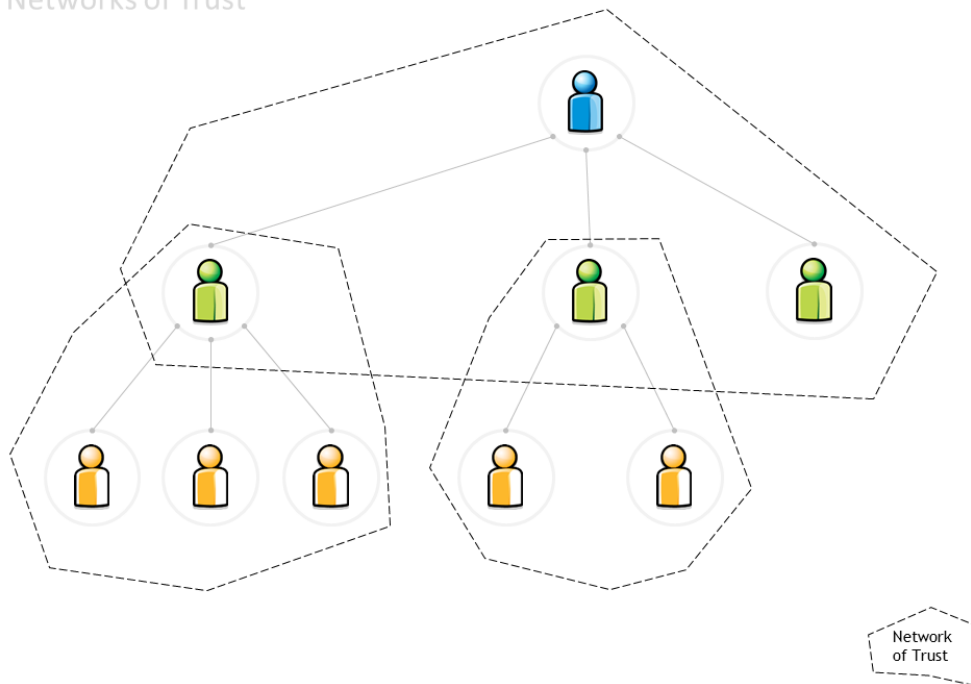
This hierarchy is a particular type of network (... a directed acyclic network or ‘tree’).

It can be viewed as 3 distinct “networks of trust”.

Network of Actors



Networks of Trust



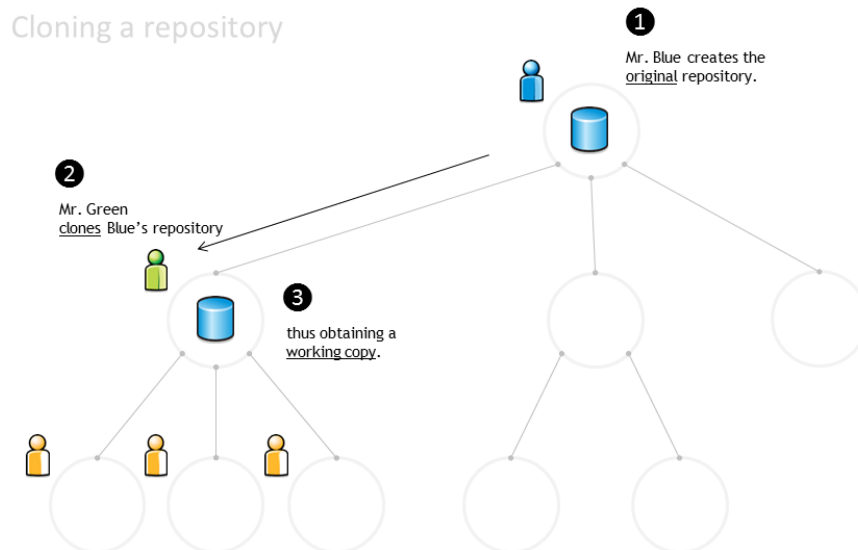
The concept of “network of trust” simplifies the work:

- The blue actor trusts the green actor to review the work done by his yellow co-workers.
 - From the blue actor’s point of view, the specific configuration of each ‘green & yellow’ network is irrelevant.
 - It is also irrelevant whether the all network is really a tree (or if a given yellow actor participates in two distinct subnetworks).
- Each actor needs only trust (and interact with) his immediate neighbourhood:
 - The green actor accepts any upstream changes approved by his blue neighbour.
 - The green actor approves (or declines) changes made by his yellow neighbours (or resolves conflicts between different changes).
- By definition, the blue actor can directly commit changes to his own blue repository of information. So can the green actors to their own green repositories.
- Each actor can also ‘pull’ into his repository any changes that his immediate neighbours have made.

Cloning a repository

The initial workflow is depicted below:

1. Mr Blue creates the **original** repository.
2. Mr Green **clones** Blue’s repository
3. thus obtaining a **working copy**.

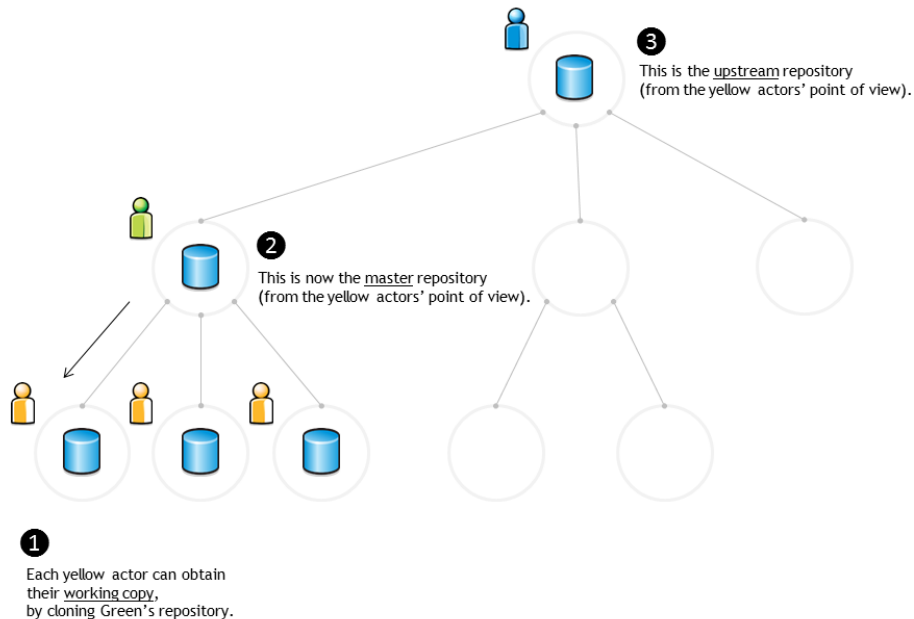


Distributed repositories

The repository is then distributed to all the team:

1. Each yellow actor can obtain their working copy, by cloning Green's repository.
2. Mr Green's repository is now the **master** repository for all the yellow actors.
3. Mr Blue's repository is now the **upstream** repository for all the yellow actors.

Distributed repositories



Synchronising repositories

The repositories must be explicitly synced. For example, suppose that:

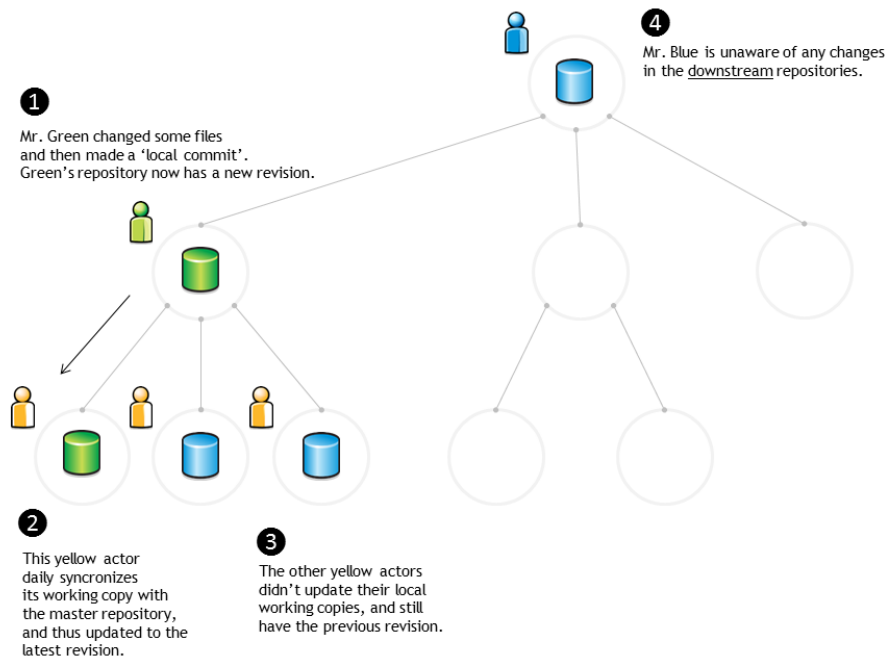
1. Mr Green changed some files and then made a 'local commit'. Green's repository now has a new revision (which does not exist in any other repository).
2. One of the yellow actors synchronises the local working copy everyday, to ensure that he has the latest files. His local working copy is updated with the latest revision made by Mr Green to the master repository.
3. The other yellow actors didn't update their local working copies, and still have the previous revision.
4. Meanwhile, Mr Blue is unaware of any changes in the downstream repositories.

Commit and merge

Changes can also be propagated upstream. Suppose that:

1. A yellow actor has changed some files and made a 'local commit'. The local repository has a new revision (jargon: 'the local repository is one commit ahead of the master repository').

Synchronizing repositories



2. The yellow actor notifies Mr Green and asks him to merge the changeset into Mr Green's repository (jargon: 'sends Mr Green a pull request').
3. Mr Green reviews the changes, approves them (or not...) and merges the changeset into his own repository.
4. Meanwhile, Mr Blue is still unaware of any changes in the downstream repositories (he has received no pull requests).

When the work assigned to Mr Green's team is ready:

1. Mr Green send a 'pull request' to Mr Blue.
2. Mr. Blue reviews and accepts the changes, and updates his repository.
3. Everyone else can synchronise their repositories to the latest version.

Storyboard #2 - Using branches to manage the document translation process

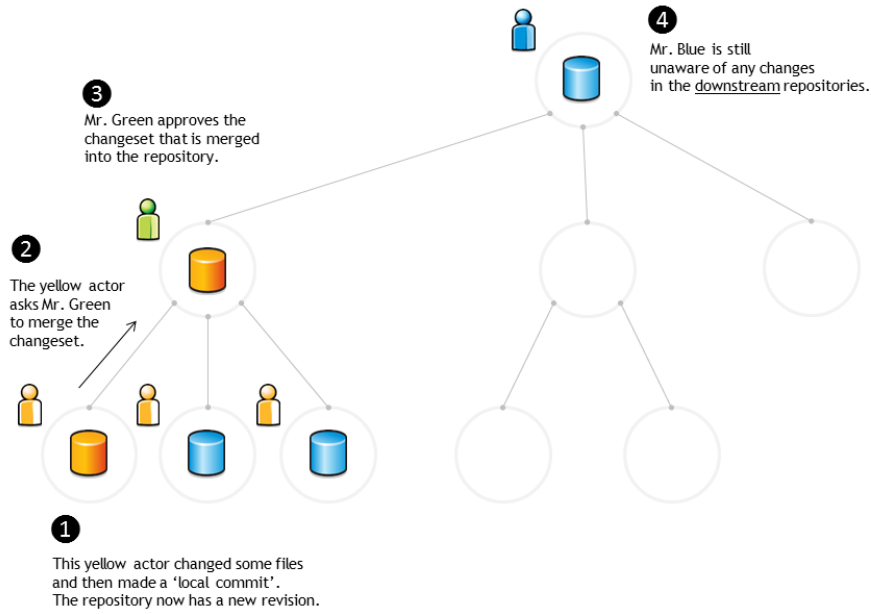
Trunk and branches

Consider the following network:

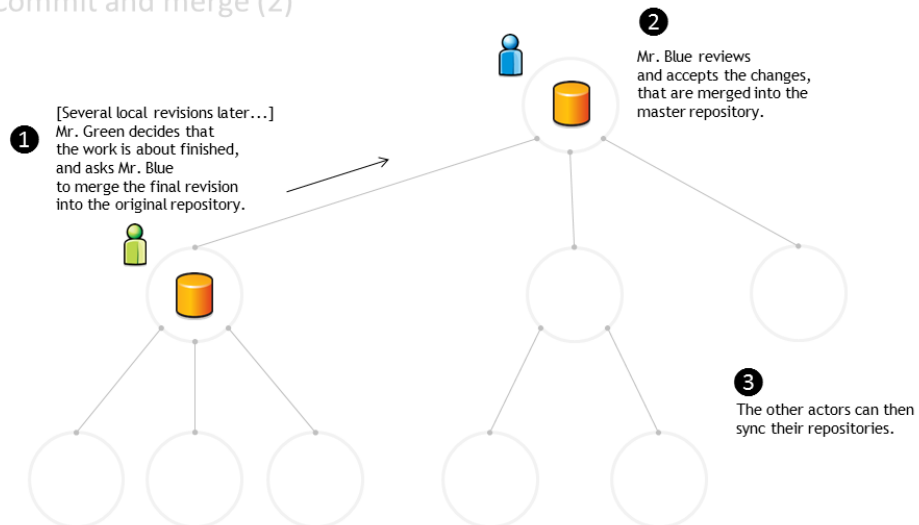
- Mr Grey is the technical writer responsible for the English version of the 'User Manual' and 'Project Handbook'. Mr Grey is also responsible for the templates and stylesheets that will be used in the various documents.
- Mr Πρσυνο is responsible for the Greek language version.
- Mr is responsible for the Bulgarian language version.
- Mr Azul is responsible for the Portuguese language version.

When the translation process begins:

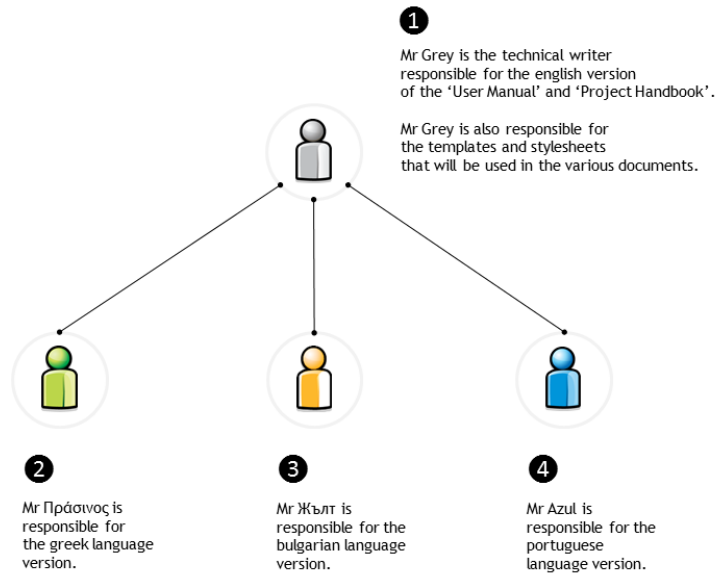
Commit and merge (1)



Commit and merge (2)



Trunk and Branches (1)



1. Mr Grey creates a repository with the English documents (e.g. one file per chapter) and with the image files (e.g. the application screenshots), templates and stylesheets required to build the final document.
2. Messrs Πράσινο, and Azul all clone the English language repository and create their own working copies.
3. Each of them also creates a local branch: a replica of the files so that each can work on translating the text to their own language while reusing the image files and the stylesheet.

In this example, the English version is the **trunk**. Each localised version is a **branch**.

When new documents are ready to be translated:

1. Mr Grey completes a new chapter and commits it to the repository.
Mr Grey also changes some of the images.
A new revision is now available.
2. Messrs Πράσινο, and Azul synchronise their working copies with the master repository (only the trunk is updated).
3. Each of them also updates the local branch.

What if Mr Πράσινο detects some spelling errors in the English version?

1. Mr Πράσινο changes the English files in the trunk, makes a local commit and notifies Mr Grey.
2. Mr Grey reviews the changes and accepts the pull request.

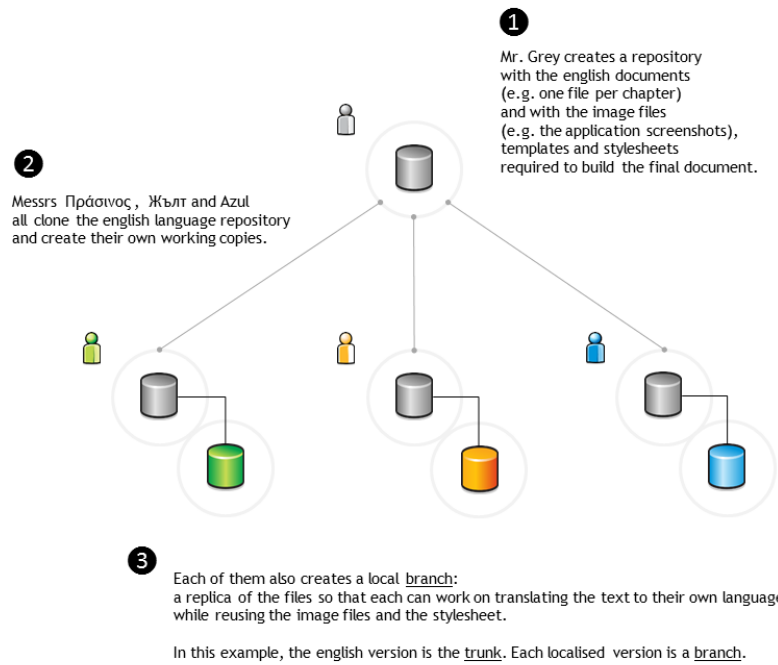
The last revision of the master repository now includes the changes made by Mr Πράσινο (but not the Greek branch).

3. Messrs and Azul synchronise their working copies with the master repository.

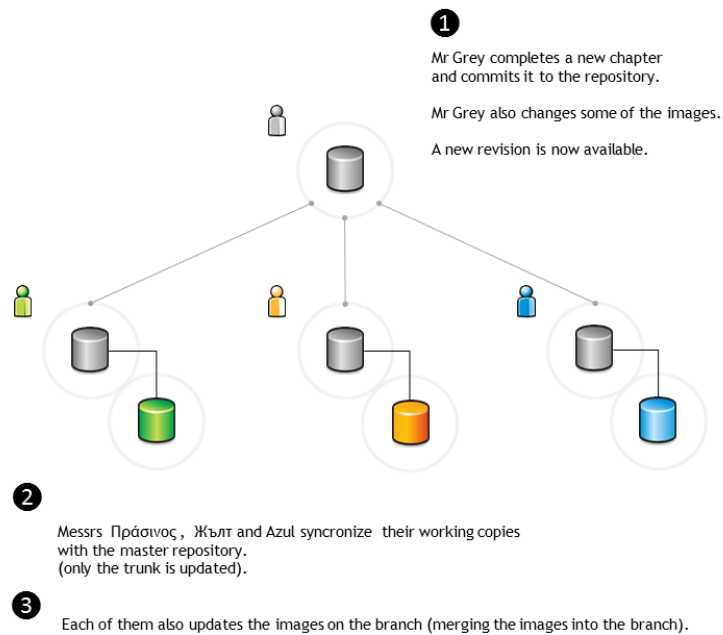
The working copies of the trunk are updated. Each team member must update their specific local branches.

Branches can function as different versions...

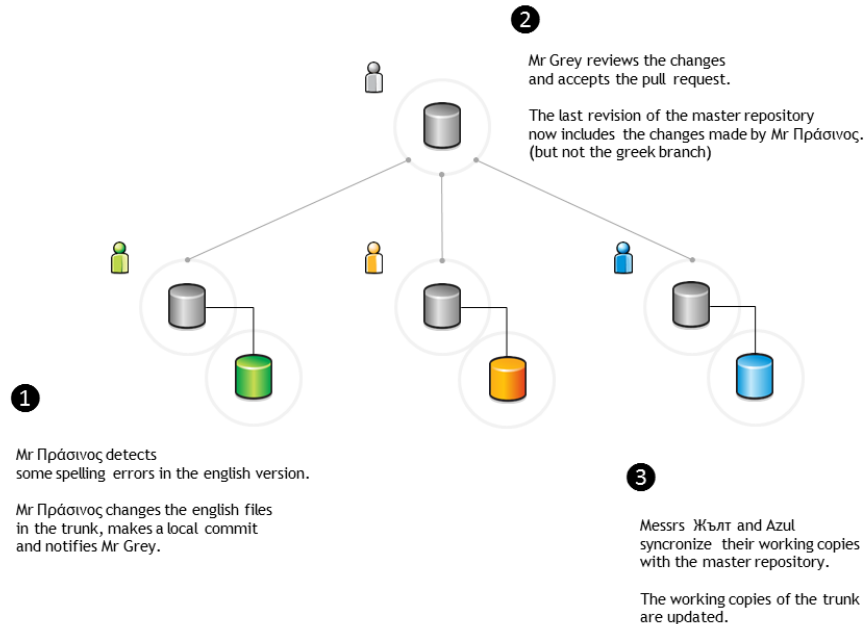
Trunk and Branches (2)



Trunk and Branches (3)



Trunk and Branches (4)



1. Mr Πρσινο changes the default stylesheet, including some styles that improve its use with the Greek alphabet.
Changes are made only in the Greek language branch.
2. Mr makes a similar change, due to the Cyrillic alphabet.
Changes are made only to the Bulgarian language branch.
3. Mr Azul dislikes the colour of chapter headings and changes the stylesheet in the Portuguese language branch.
Mr Azul decides to submit the changes to Mr Grey, so that the trunk can also be changed.
4. Mr Grey reviews the changes made by Mr Azul, but does not accept them.
The trunk stylesheet is not changed.

Glossary

baseline An approved revision of a file from which subsequent changes can be made.

branch A set of files under version control may be branched (forked) at a point in time. From that time forward, the two copies of the files may develop in different ways, independently of each other.

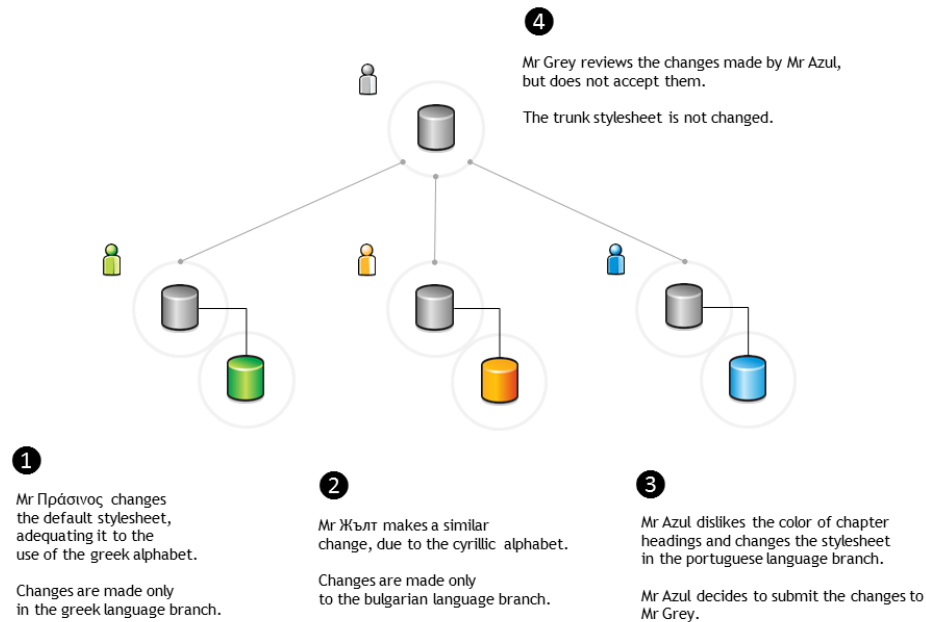
change A change (or diff, or delta) represents a specific modification to a file under version control.

changeset A collection of files that have changes.

checkout See 'clone'.

clone To clone is to create a local working copy from the repository. A user may specify a revision or obtain the latest. In centralised version control systems (with a single central repository), the term 'checkout' is also used. The term 'checkout' can be used as a noun to describe the working copy.

Trunk and Branches (5)



commit To commit is to write or merge the changes made in the working copy back to the repository. The terms ‘commit’ and ‘checkin’ can also be used as nouns to describe the revision that is created as a result of committing.

conflict A conflict occurs when different parties make changes to the same file, and the system is unable to reconcile the changes.

A user must resolve the conflict by combining the changes, or by selecting one change in favour of the other.

fork See ‘branch’.

head The most recent revision, either to the trunk or to a branch.

The trunk and each branch have their own head. HEAD is sometimes used to refer to the head of the trunk.

merge A merge is an operation in which two sets of changes are applied to a file or set of files under version control.

A user updates their working copy with changes made to the repository by other users.

A user tries to update a repository with changes made to a working copy.

repository The repository is where the files’ current and historical data are stored, often on a server.

resolve The act of user intervention to address a conflict between different changes to the same file.

revision A revision (version) is any registered “snapshot” in time of the repository.

sync See ‘update’.

trunk The trunk is the “main” line of development to the collection of information under version control, consisting only of ‘baseline’ (approved) files.

update An update (or sync) merges changes made in the original repository (by other users, for example) into the local working copy.

version See ‘revision’.

working copy A working copy is a local copy of files from a repository, made at a specific time (revision).

Version Control using Git

Before you start

The following resources contain useful information on version control systems:

- **A Visual Guide to Version Control:** a simple explanation of version control with Subversion examples.
- **A successful Git branching model:** a clear and structured workflow.

Git CheatSheet

Source

Git CheatSheet,(c) 2011, salesforce.com, inc., URL: https://na1.salesforce.com/help/doc/en/salesforce_git_developer_cheatsheet.pdf

Overview

When you first setup Git, set up your user name and email address so your first commits will record them properly:

```
git config --global user.name "My Name"
git config --global user.email "user@email.com"
```

Basic Git Workflow Example

Initialise a new git repository, then stage all the files in the directory and finally commit the initial snapshot:

```
$ git init
$ git add .
$ git commit -m 'initial commit'
```

Create a new branch named feature_A, check it out so it is the active branch, then edit and stage some files and finally commit the new snapshot:

```
$ git branch feature_A
$ git checkout feature_A
$ (edit files)
$ git add (files)
$ git commit -m 'add feature A'
```

Switch back to the master branch, reverting the feature_A changes you just made, then edit some files and commit your new changes directly in the master branch context.:

```
$ git checkout master
$ (edit files)
$ git commit -a -m 'change files'
```

Merge the feature_A changes into the master branch context, combining all your work. Finally delete the feature_A branch.:

```
$ git merge feature_A
$ git branch -d feature_A
```

Setup & Init

Git configuration, and repository initialisation & cloning.

command	description
<code>git config [key] [value]</code>	set a config value in this repository
<code>git config global [key] [value]</code>	set a config value globally for this user
<code>git init</code>	initialise an existing directory as a Git repository
<code>git clone [url]</code>	clone a Git repository from a URL
<code>git help [command]</code>	get help on any Git command

Stage & Snapshot

Working with snapshots and the Git staging area.

command	description
<code>git status</code>	show the status of what is staged for your next commit and what is modified in your working directory
<code>git add [file]</code>	add a file as it looks now to your next commit (stage)
<code>git reset [file]</code>	reset the staging area for a file so the change is not in your next commit (unstage)
<code>git diff</code>	diff of what is changed but not staged
<code>git diff --staged</code>	diff of what is staged but not yet committed
<code>git commit</code>	commit your staged content as a new commit snapshot
<code>git rm [file]</code>	remove a file from your working directory and unstage

Branch & Merge

Working with Git branches and with the stash.

command	description
<code>git branch</code>	list your branches. a * will appear next to the currently active branch
<code>git branch [branch-name]</code>	create a new branch at the current commit
<code>git checkout [branch]</code>	switch to another branch and check it out into your working directory
<code>git checkout -b [branch]</code>	create a branch and immediately switch to it
<code>git merge [branch]</code>	merge another branch into your currently active one and record the merge as a commit
<code>git log</code>	show commit logs
<code>git stash</code>	stash away the currently uncommitted modifications in your working directory temporarily
<code>git stash apply</code>	re-apply the last stashed changes

Share & Update

Fetching, merging and working with updates from another repository.

command	description
<code>git remote add [alias] [url]</code>	add a git URL as an alias
<code>git fetch [alias]</code>	fetch down all the branches from that Git remote
<code>git merge [alias]/[branch]</code>	merge a branch on the server into your currently active branch to bring it up to date
<code>git push [alias] [branch]</code>	push the work on your branch to update that branch on the remote git repository
<code>git pull</code>	fetch from the URL tracked by the current branch and immediately try to merge in the tracked branch

Inspect & Compare

Examining logs, diffs and object information.

command	description
<code>git log</code>	show the commit history for the currently active branch
<code>git log branchB...branchA</code>	show the commits on branchA that are not on branchB
<code>git log --follow [file]</code>	show the commits that changed file, even across renames
<code>git diff branchB...branchA</code>	show the diff of what is in branchA that is not in branchB
<code>git show [SHA]</code>	show any object in Git in human-readable format

Contributing on GitHub

To contribute to a project that is hosted on GitHub (or another repository hosting site, such as BitBucket) you can fork the project online, then clone your fork locally, make a change, push back to GitHub and then send a pull request, which will email the maintainer.:

```
fork project on github
$ git clone https://github.com/my-user/project
$ cd project
$ (edit files)
$ git add (files)
$ git commit -m 'Explain what I changed'
$ git push origin master
go to github and click 'pull request' button
```

Visual Git Cheatsheet

Source

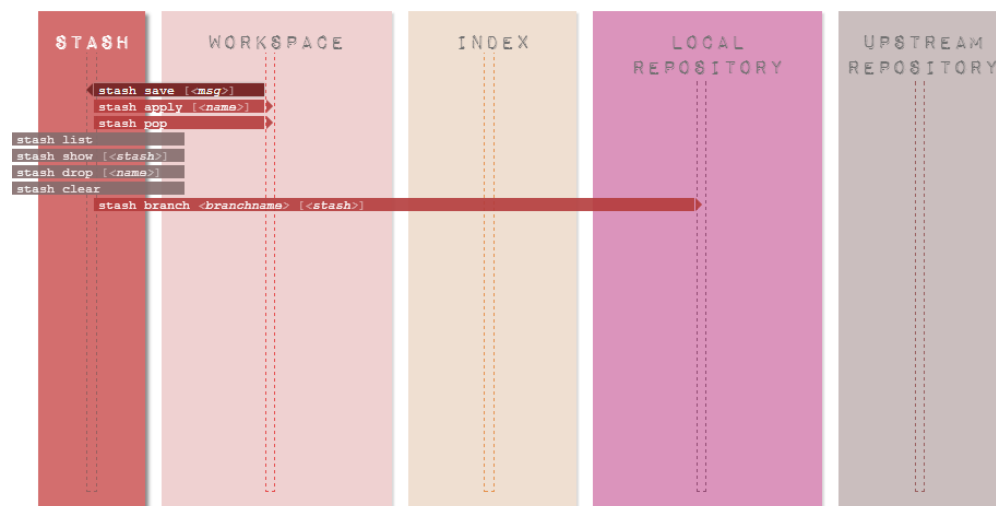
Git Cheatsheet, (c) 2009-2012, Andrew Peterson url: <http://ndpsoftware.com/git-cheatsheet.html>

A list of Git commands, categorized on what they affect.

The interactive online version provides a description for each of the commands.

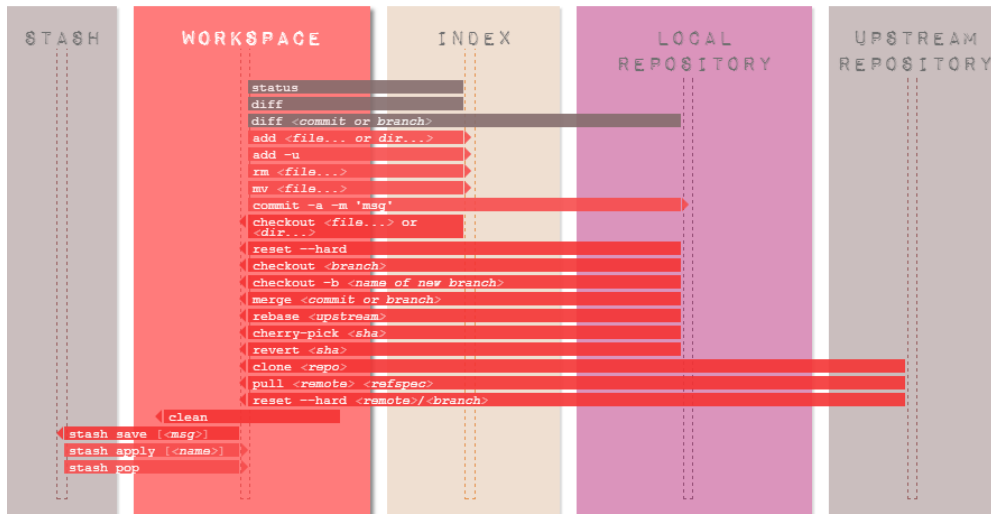
Stash

A place to hide modifications made to the workspace, while working on something else. (The stash area is not required in a “normal” workflow.)



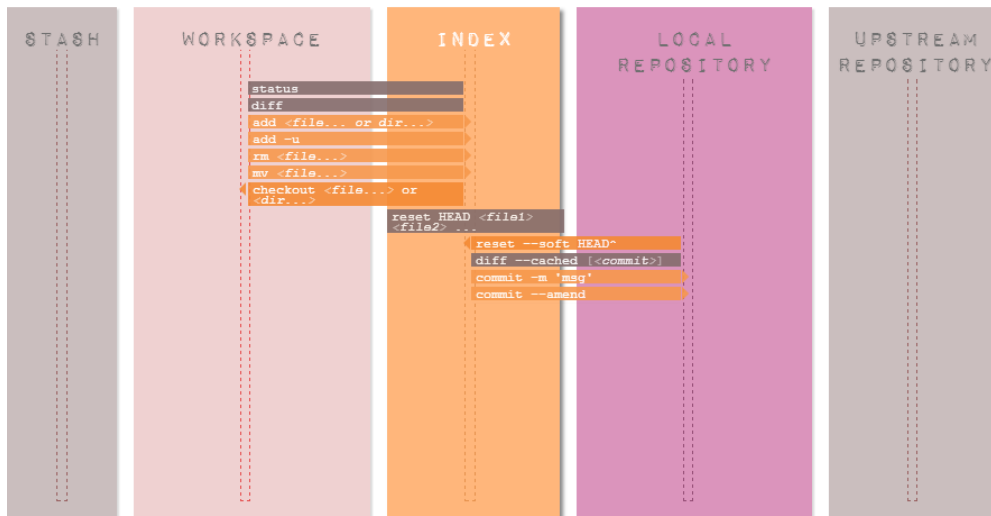
Workspace

The local working area.



Staging area

The “index”– or “staging area” – holds a snapshot of the content of the working area, and it is this snapshot that is taken as the contents of the next commit.



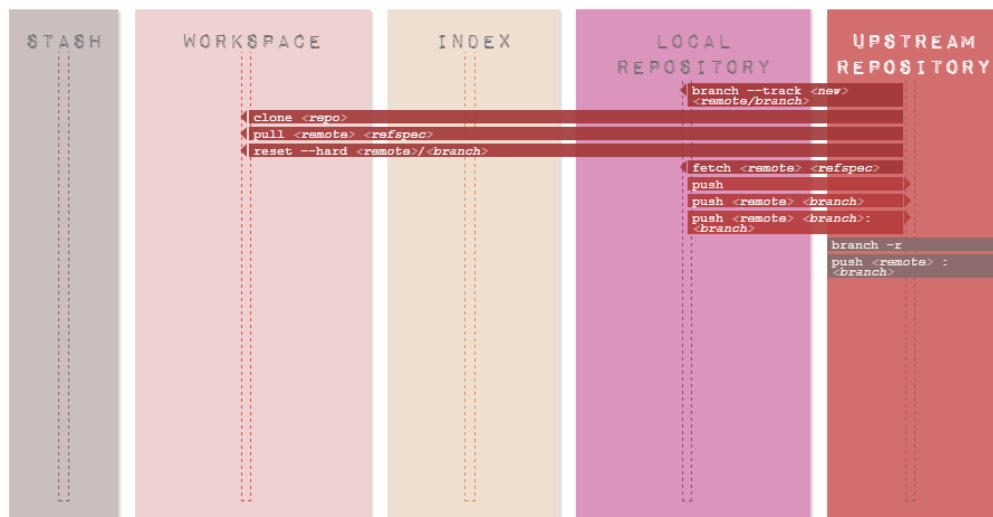
Local repository

A local area under version control. Typical branches: master, dev (for local development), feature_x, bugfix_y



Upstream repository

Typically a remote area under version control. Default name is 'origin'. Typical branches here: master, shared_feature_x, release_y.



How to...

This section include miscellaneous Git commands to perform different operations.

Set up a merge tool to resolve conflicts

Configure kdiff3 as the merge tool (in Windows):

```
$ git config --global mergetool.kdiff3.path 'C:\Program Files (x86)\KDiff3\kdiff3.exe'
$ git config --global merge.tool kdiff3
```

Invoke kdiff3:


```
$ git mergetool <file>
```

Force an update from the upstream repository

This operation will discard all changes in the local repository:

```
$ git reset --hard HEAD
$ git pull
```

Add untracked files to the set of files under version control

A pattern can be used. For example, this will add any new or untracked *.rst file:

```
$ git add $(git ls-files --other *.rst)
```

Remove multiple files from the set of files under version control

This will remove multiple files that have already been deleted from disk:

```
$ git rm $(git ls-files --deleted)
```

Alternatively, edit the .git\config file, and add the following lines:

```
[alias]
  rma = !git ls-files --deleted -z | xargs -0 git rm
```

Then run the command using the alias:

```
$git rma
```

Disable quoted file names

Special character and spaces in file names can be problematic. To disable quotes file names (Windows Unicode Support), use:

```
$ git config [--global] core.quotepath off
```

Setting up an online Git Repository

Three possible alternatives are:

- Using Atlassian [Bitbucket](#): it is free for public and private repositories (up to 5 team members), and also supports Mercurial repositories.
- Using [GitHub](#): it is free for public repositories
- Using [Dropbox](#) : is is free up to a 2GB maximum storage.

Using Bitbucket

1. Create a Bitbucket account and a new repository “projectXPTO”
2. Install Git in your computer and set the global configuration (using Git Bash):

```
$ git config --global user.name "johndoe"
$ git config --global user.email johndoe@example.com
```

3. Create a local git repository:

```
$ mkdir /path/to/your/project
$ cd /path/to/your/project
$ git init
```

4. Link the remote git repository to your local repository:

```
$ git remote add origin https://johndoe@bitbucket.org/johndoe/projectXPTO.git
```

5. Add a ReadMe file:

```
$ echo "# This is my README" >> README.md
$ git add README.md
```

6. Commit and push the first change:

```
$ git commit -m "First commit. Adding a README."
$ git push -u origin master
```

Using GitHub

The steps are similar to the ones when using Bitbucket...

1. Create a GitHub account and a new repository “projectXPTO”

(steps 2 and 3 as above)

4. Link the remote git repository to your local repository:

```
$ git remote add origin https://github.com/johndoe/projectXPTO.git
```

(steps 5 and 6 as above)

Using Dropbox

Please note that this is a more complicated solution, that is only useful if the [Bitbucket](#) or [Github](#) options cannot be used for some reason...

[Dropbox](#) is a cloud storage service provider. A Dropbox client application is available for Windows, Mac OSX, Linux and Android operating systems. The client application synchronises the content of a local Dropbox folder (in the client computer's disk) with the cloud Dropbox storage area.

A git repository is created in the local Dropbox folder and it will work if it were an “remote” upstream git repository.

Another local repository (located somewhere in the local disk, but **not** in the Dropbox folder) can then clone, push or sync with the Dropbox “remote” repository.

The rest is done automatically by the Dropbox application: the “remote” folder will be synced with online storage and will be accessible from anywhere.

Setup the “remote” and the local repositories

1. Install both Git and the Dropbox client application on the computer.
2. Go to the local Dropbox folder and create a bare repository. Open a Git Bash window:

```
$ cd ~/Dropbox
$ mkdir -p remoteRepos/ProjectXPTO
$ git init -bare remoteRepos/ProjectXPTO
```

3. Go to the local project folder, and start a local git repository:

```
$ cd ~/localRepos/ProjectXPTO
$ git init .
$ git add .
$ git commit -all -m "Initial commit"
```

4. Link the local repository to the “remote” repository on the Dropbox folder:

```
$ git remote add dropbox /Dropbox/remoteRepos/ProjectXPTO/
```

5. Push all the local changes to the “remote” repository:

```
$ git push dropbox master
```

Clone the “remote” repository to a different machine

1. Again, both Git and the Dropbox application must be installed **and** the Dropbox folders must be synced.
2. Then, clone the “remote” repository with:

```
$ cd ~/otherMachine/ProjectXPTO
$ git clone -o dropbox /Dropbox/remoteRepos/ProjectXPTO/
```

Push changes to the “remote” repository

1. Changes to the local project can be pushed back to the “remote”:

```
$ git commit -all -m "Changes made!"
$ git push dropbox master
```

Sync the local copy with the “remote” repository

1. To sync the local copy with the “remote” repository:

```
$ git pull dropbox master
```

Set up SSH for Git

Note: Sources

Mostly from: <https://confluence.atlassian.com/display/BITBUCKET/Set+up+SSH+for+Git>

Mixed with: <https://help.github.com/categories/56/articles> <https://help.github.com/articles/working-with-ssh-key-passphrases> <http://nerderati.com/2011/03/17/simplify-your-life-with-an-ssh-config-file/>

When you use HTTPS, you need to authenticate (supply a username and password) each time you take an action that communicates with the remote server. This page shows you how to use secure shell (SSH) to communicate with the Bitbucket or Github server and avoid having to manually type a password.

Step 1. Check if you have existing default Identity

The Git Bash shell comes with an SSH client. Do the following to verify your installation:

1. Double-click the Git Bash icon to start a terminal session.
2. Enter the following command to verify the SSH client is available:

```
$ ssh -v
OpenSSH_4.6p1, OpenSSL 0.9.8e 23 Feb 2007
usage: ssh [-1246AaCfGkMNnqsTtVvXxY] [-b bind_address] [-c cipher_spec]
[-D [bind_address:]port] [-e escape_char] [-F configfile]
[-i identity_file] [-L [bind_address:]port:host:hostport]
[-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
[-R [bind_address:]port:host:hostport] [-S ctl_path]
[-w local_tun[:remote_tun]] [user@]hostname [command]
```

3. If you have ssh installed, go to the next step.

If you don't have ssh installed, install it now with your package manager.

4. List the contents of your ~/.ssh directory.

If you have not used SSH on Bash you might see something like this:

```
$ ls -a ~/.ssh
ls: /c/Users/your-user-name/.ssh: No such file or directory
```

If you have a default identity already, you'll see two id_* files:

```
$ ls -a ~/.ssh
.      ..      id_rsa      id_rsa.pub  known_hosts
```

In this case, the default identity used RSA encryption (id_rsa.pub). If you want to use an existing default identity for your Bitbucket account, skip the next section and go to create a config file.

Step 2. Set up your default identity

By default, the system adds keys for all identities to the /Users/your-user-name/.ssh directory. The following procedure creates a default identity.

1. Open a terminal in your local system. Enter ssh-keygen at the command line:

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key:
```

To create a key with a name other than the default, specify the full path to the key. Enter and reenter a passphrase when prompted. Unless you need a key for a process such as script, you should always provide a passphrase. The command creates your default identity with its public and private keys.

2. List the contents of `~/.ssh` to view the key files. You should see something like the following:

```
$ ls ~/.ssh
id_rsa id_rsa.pub
```

The command created two files, one for the public key (for example `id_rsa.pub`) and one for the private key (for example, `id_rsa`).

Step 3. Create a SSH config file

1. Using a text editor, edit the `~/.ssh/config` file. Add the following entries to the configuration file using the following format:

```
Host bitbucket.org
  IdentityFile ~/.ssh/id_rsa

Host github.com
  IdentityFile ~/.ssh/id_rsa
```

Every second line is indented. That indentation (a single space) is important, so make sure you include it. The second line is the location of your private key file.

2. Save and close the file.
3. Restart the GitBash terminal.

Step 4. Update your `.bashrc` profile file

It is a good idea to configure your GitBash shell to automatically start the agent when launch the shell. The `.bashrc` file is the shell initialization file. To start the agent automatically, do the following.

1. Start GitBash.
2. Edit your `~/.bashrc` file.

Add the following lines to the file:

```
SSH_ENV=$HOME/.ssh/environment

# start the ssh-agent
function start_agent {
  echo "Initializing new SSH agent..."
  # spawn ssh-agent
  /usr/bin/ssh-agent | sed 's/^echo/#echo/' > "${SSH_ENV}"
  echo succeeded
  chmod 600 "${SSH_ENV}"
  . "${SSH_ENV}" > /dev/null
  /usr/bin/ssh-add
```

(continues on next page)

(continued from previous page)

```
}

if [ -f "${SSH_ENV}" ]; then
    . "${SSH_ENV}" > /dev/null
    ps -ef | grep ${SSH_AGENT_PID} | grep ssh-agent$ > /dev/null || {
        start_agent;
    }
else
    start_agent;
fi
```

3. Save and close the file.
4. Restart the GitBash terminal.
5. The system prompts you for your passphrase.
6. Enter your passphrase. After accepting your passphrase, the system displays the command shell prompt. Verify that the script identity added your identity successfully by querying the SSH agent:

```
$ ssh-add -l
```

After you install your public key to Bitbucket/Github, having this script should prevent you from having to enter a password each time you push or pull a repository from Bitbucket.

Step 5. Install the public key on your Bitbucket/Github account

In Bitbucket:

1. Open a browser and log into Bitbucket.
2. Choose avatar > Manage Account from the menu bar.
3. The system displays the Account settings page. Click SSH keys. The SSH Keys page displays. It shows a list of any existing keys. Then, below that, a dialog for labeling and entering a new key.

Copy the contents of the public key file into the SSH Key field. Click the Add key button. The system adds the key to your account.

In Github:

1. Goto to the account settings, everything is pretty much as above.

Return to the GitBash terminal window:

1. Verify your configuration by entering the following commands:

```
ssh -T git@bitbucket.org

ssh -T git@github.com
```

The command message tells you which Bitbucket account can log in with that key. Verify that the command returns your account name.

Step 6. Configure your repository to use the SSH protocol

The URL you use for a repository depends on which protocol you are using, HTTPS and SSH.

In Bitbucket:

- `ssh://git@bitbucket.org/accountname/reponame.git`
- `https://accountname@bitbucket.org/accountname/reponame.git`

The same goes for Github:

```
* ssh://git@github.com/accountname/reponame.git
* https://accountname@github.com/accountname/reponame.git
```

So...

1. View your current repository configuration file `.git/config`, that should similar to this:

```
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
  url = https://accountname@domain/accountname/reponame.git
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

2. Change the url:

```
[remote "origin"]
  fetch = +refs/heads/*:refs/remotes/origin/*
  url = ssh://git@domain/accountname/reponame.git
[branch "master"]
  remote = origin
  merge = refs/heads/master
```

3. Save your edits and close the file.

1.4 Commercial-Off-the-Shelf Software Selection Process

1.4.1 Overview

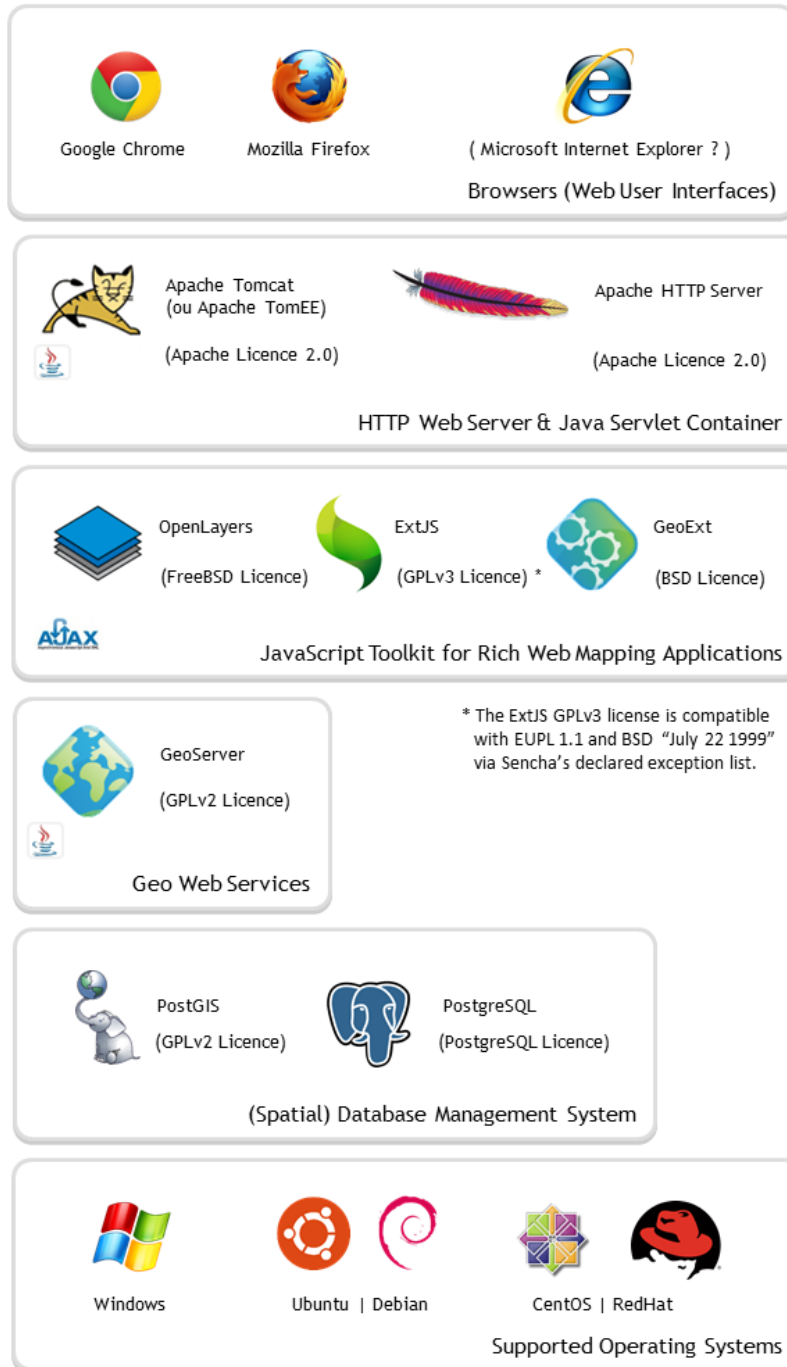
- *(Spatial) Data Infrastructure*
- *Project Management Infrastructure*

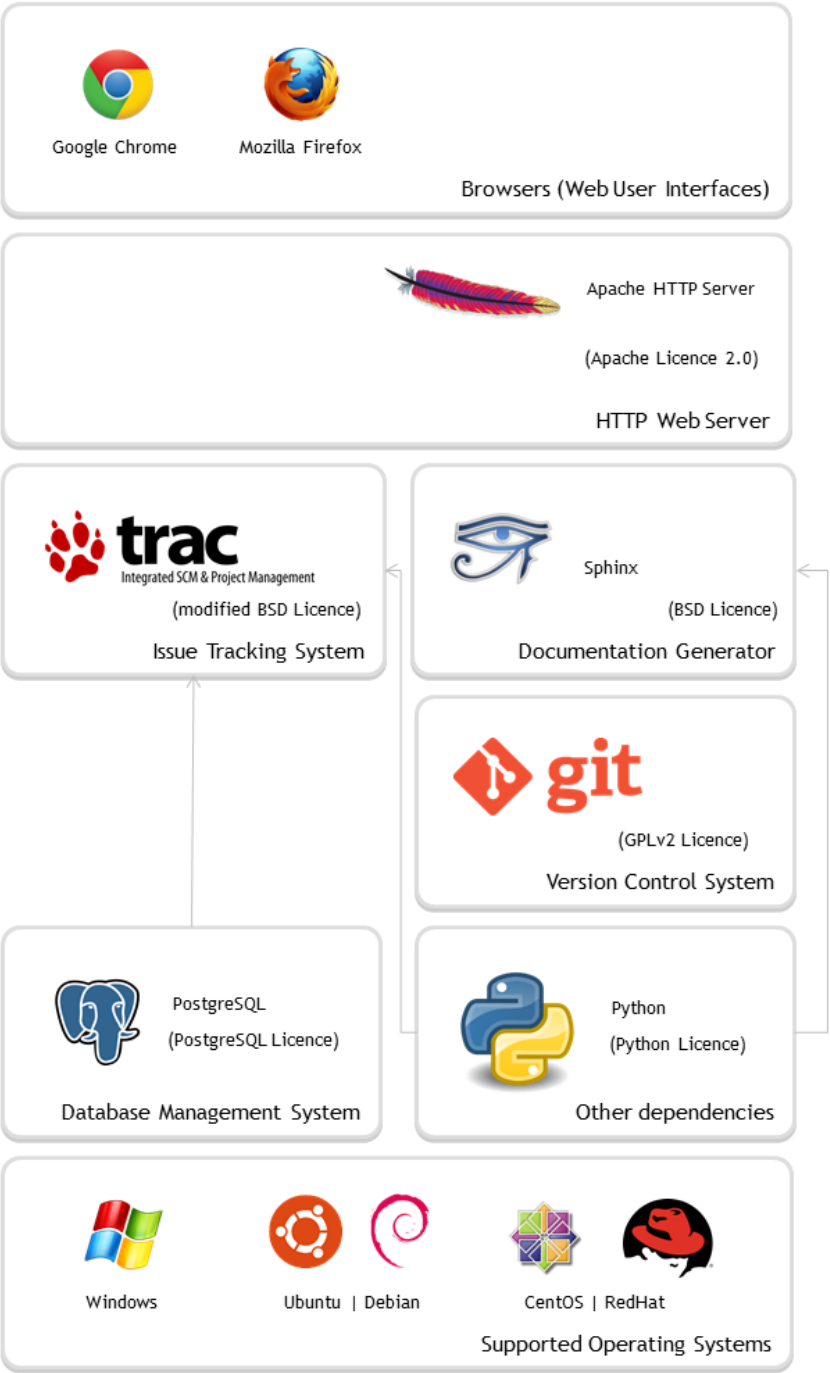
(Spatial) Data Infrastructure

Project Management Infrastructure

1.4.2 Software selection

Several commercial Off-the-Shelf (COTS) components will be required by the software solution to be developed. A basic set of components can be identified and selected at this early stage, namely a *Database Management System* and a *Geospatial Data Server*. This is still a preliminary and incomplete selection, since the functional requirements have not been fully specified and approved.





The project involves the exchange of statistical data between European Union Member States and EU institutions, such as Eurostat. The **SDMX** standards and content guidelines will be adopted where applicable. The [Open Source Software for SDMX developed by EuroStat](#) will be used ‘by default’ where required and applicable. Eurostat’s SDMX tools are thus excluded from the selection process applied to COTS components, except with regard to the portability constraint (see [Constraints](#)).

In addition, a software portfolio is selected to support the project’s *Project Management Infrastructure*. The functional requirements for these components are generic and well known for open technology development (see [HOSG09] or [Foge09]) and an integrated solution can be proposed.

The following generic technical goals for open-technology development direct the selection process:

Goals

1. Flexibility.

A component that can be used in a variety of ways tends to have more potential users, some of whom may aid the project (e.g., via bug reports and development time).

2. Portability.

A component that can be used on more platforms tends to have more potential users.

3. Modularity.

A component that is modular (e.g., with clearly-defined sub-components and perhaps support for a “plug-in” architecture) is more flexible, as well as being easier to review for correctness.

4. Use of Open Standards.

Where possible, avoid depending on interfaces that are controlled by a single vendor.

5. Reuse and collaborate with existing OTD (Open Technology Development) projects.

A project should focus on building new software, not re-implementing OTD projects that already exist. [...]

6. Avoid non-OTD dependencies.

Depend only on widely-used OTD platforms, libraries, and development tools. If a component depends on a non-OTD component, and that component then needs to be changed, it may be difficult to make the change or have that change incorporated. Similarly, it may be difficult to get support for unusual libraries and development tools. [...] If a proprietary component must be depended on, isolate it through plug-ins or an interface defined by an open standard. [...]

Where there are alternative approaches, a simple analysis of alternatives should be performed, and discussed among the project so that key issues or alternatives are not overlooked.

Source: [SWLH11]

Process

The selection process follows the common GCS process:

General COTS Selection Process

1. Define the evaluation criteria based on stakeholders’ requirements and constraints.
2. Search for COTS products.

3. Filter the search results based on a set of ‘must-have’ requirements [*“apply constraints”*]. This results in defining a short list of most promising COTS candidates which are to be evaluated in more detail.
4. Evaluate COTS candidates on the short list [*“apply factors”*]
5. Analyse the evaluation data [...] and select the COTS product that has the best fitness with the criteria. [...]

After Step 5, the selected COTS product is usually customised (a.k.a. tailored) as needed in order to reduce the mismatches it has with the requirements. A COTS product can be customised in different ways, such as using add-ons, adjusting parameters, etc.

Source: [MoRE07]

For each type of selected components, the rationale for the adopted criteria and a brief analysis of alternatives are presented in the respective sections.

The candidate set of components is trimmed and ranked using the following strategies:

Strategies to evaluate COTS products

1. *Keystone identification*, which starts by identifying a key requirement (e.g. vendor location or type of technology), and then searching for products that satisfy this keystone requirement. This allows quick elimination of a large number of products that do not satisfy the key requirement.
2. *Progressive filtering*, which starts with a large number of COTS, and then progressively defines discriminating criteria through successive iterations of product evaluation cycles, where in each cycle ‘less fit’ products are eliminated.

This strategy requires running steps 1 to 4 in the GCS process iteratively until a small number of most promising COTS products is identified from which one or more can be selected for integration into the system.

3. *Puzzle assembly*, which assumes that a COTS-based system requires fitting various components together like pieces of a puzzle. This implies that a product that ‘fits’ in isolation might not be acceptable when combined with other products. Therefore, this strategy suggests considering the requirements of each product while simultaneously remembering the requirements of other products in the puzzle.

Source: [MoRE07]

Constraints

Major constraints

The *keystone identification* strategy requires the definition of constraints. A constraint serves to limit the alternatives under consideration [East99]. Three major constraints are identified:

Licence type

Licence compatibility is a non-functional project requirement: the objective is to facilitate the sharing, reuse and future improvement of the solution to be developed, and, if required, of any existing components it may incorporate.

Following the principle of early determination of distribution policy [TrCo10], the software solution to be developed should be distributed under an OSI-approved Open Source licence, namely the European Union Public Licence v.1.1 [EUPLv1.1].

With regard to some of the COTS components, EUPLv1.1 compatibility is also a *puzzle assembly* constraint: EUPLv1.1 is a copyleft licence and all the Open Source Software for SDMX developed by Euro-

Stat is distributed under the EUPLv1.1 licence.

Two alternatives are possible:

1. To evaluate only COTS software under OSI-approved Open Source licences that are [EUPL-compatible licence](#) (either directly or through a FOSS exception list established by the licensor).
2. To evaluate also COTS software under OSI-approved Open Source licences with which the EUPLv1.1 is compatible with (either directly or through the EUPLv1.1 exception list).

Most licences allow static linking with EUPLv1.1 (combining components through compilation, copying them into the target application and producing a merged object file that is a stand-alone executable), but GNU licences (LGPL, GPL) typically do not allow dynamic linking (using components at the time the application is loaded or executed) or incorporation of source code. Since a majority of open-source components is under copyleft GNU licences, the (also copyleft) EUPLv1.1 includes an exception list or downstream compatibility list, that allows mergers between EUPLed code and:

- General Public License (GPL) v.2
- Open Software License (OSL) v.2.1, v.3.0
- Common Public License (CPL) v.1.0
- Eclipse Public License (EPL) v.1.0
- Cecill v.2.0

From COTS selection purposes, the above listed copyleft licences are considered EUPL-compatible (in really, it is the EUPLv1.1 licence that is compatible with them).

The final licence for distribution purposes must be confirmed in the end of the project, based of the type of components and their use in the system.

Portability

Cross-platform compatibility is a non-functional project requirement: again, the objective is to facilitate the adoption by different organizations with distinct IT infrastructures. The portability requirement is also a good ICT procurement practice that reduces the risk of [vendor lock-in](#) and facilitates the reuse of the components by different systems.

Only COTS components that are available for and supported on common proprietary and open source operating systems are taken under consideration. The platforms' selection procedure is described in the [Operating Systems](#) section.

Acquisition Cost

Only software with zero cost of licence acquisition is included: the project's budget contemplates infrastructure costs (i.e. for application hosting) and software development/customization but it does not allow for licence acquisition/maintenance costs.

As a result of the application of these constraints, the candidate sets are strictly based on FOSS (Free and Open Source Software) while guaranteeing compatibility with dominant proprietary and open source operating systems.

Maturity and Sustainability

In practice, the acquisition cost constraint does not influence the set of alternatives, because adequate components are either freely available, or have a 'community edition' which is free and an 'enterprise edition' with paid support services.

Indirectly, this situation derives from the application of a second set of constraints related to the maturity and sustainability of each of the FOSS components. Two evaluation models are combined:

- the Software Sustainability Maturity Model (SSMM) [Gard13]
- the Qualification and Selection of Open Source (QSOS) maturity criteria [Atos13]

Basically:

- no COTS component with an SSMM level below 4 is considered acceptable;
- and the following QSOS maturity criteria scores are set as constraints:

QSOS maturity criteria and minimum scores selected as constraints

Legacy : Project's history and heritage

- Age : { 0 : Less than three months }
- Popularity : { 0 : Very few identified users }

Activity : Activity inside and around the project

- Contributing community : { 0 : No real community nor activity (forum, mailing lists...) }
- Activity on bugs : { 0 : Low reactivity in forums and mailing lists, or no mention about bugfixes in release notes }
- Activity on features : { 0 : Few or no new features }
- Activity on releases/versions : { 0 : Very low activity on the production or development versions (alpha, beta) }

Industrialisation : Industrialisation of the project

- Services : { Existing service offerings (support, training, audit...) = 0 : No service offering identified }
- Documentation : { 1 : Documentation exists but is partly obsolete or restricted to one language or to few details }
- Source code modification : { 0 : No convenient way to propose source code modifications }

Activity on releases/versions is evaluated based on each project's website: only projects under active development are considered (at least one minor stable release in the past 12 months), only projects with stable releases are considered.

Activity on bugs and features is evaluated either directly (through the project's version release notes, bug tracker or source forge statistics) or indirectly, through FOSS statistics gathering sites (mainly [ohloh](#)).

Contributing community activity is evaluated for the end-user and developer communities. End-user activity is evaluated directly (through the project's forum, mailing lists, etc.) and indirectly, through community-based Q&A sites (mainly [stackoverflow](#) and sibling sites). Developer activity is also evaluated through source forge statistics (number of contributing developers, number of commits).

Documentation is evaluated directly at the project's website. Only projects with publicly available documentation are considered. User documentation must include at least the hardware and software requirements, installation procedure and software configuration and operation. Developer documentation must include at least source code documentation. This constraint is also a basic requirement to allow the *progressive filtering* and functional evaluation of the candidate set.

Open Standards

The selection of COTS on the basis of its implementation of open standards is a basic technical interoperability requirement.

The Open Standards listed in the recent Portuguese National Regulation on Digital Interoperability [RNID12] are adopted where applicable, if more stringent requirements are not defined by European Union legislation (see [INSPIRE]) or required by the project's functional needs.

Important request:

If more stringent requirements on open standards exist in Cyprus, Bulgaria or other interested MS, please send us the applicable documentation.

In Portugal, compliance with the Open Standards listed in the Portuguese National Regulation on Digital Interoperability is also a legal requirement for the public administration systems development and ICT procurement.

In practice, the open standards constraint did not (significantly) influence the set of alternatives, because FOSS components typically implement the relevant open standards. However, the level of support for the applicable open standards is a factor in the ranking and selection of alternatives.

Todo: RNID open standards

Include list of RNID open standards and its application as selection criteria of each type of component.

1.4.3 Factors

A factor is a criteria that enhances or detracts from the suitability of a specific alternative for the activity under consideration [East99]. Functional requirements vary with the type of component: the rationale for the adopted criteria and a brief analysis of alternatives is presented in each respective section.

Functional criteria have precedence over non-functional criteria. However, there is an obvious correlation between the maturity and sustainability score of a given COTS component and its functionality, flexibility, modularity and integration capabilities.

Furthermore, non-functional criteria related to the maturity and sustainability of FOSS projects - based on the SSMM and the QSOS model - can be applied when a 'best' choice does not emerge from the application of functional criteria.

The following nine criteria are commonly used in FOSS evaluation [Berg05]: licence type, documentation, release activity, longevity, community, support, security, integration and functionality. A short description on the application of these criteria on the selection process is given below:

- Licence

Licence type is one of the constraints used to limit the set of alternatives.

Licence compatibility with EUPLv1.1 is defined as allowing the distribution of the larger work under the EU-PLv1.1 either through incorporation of source code, static linking or dynamic linking. Licence type can thus be used as a factor to rank alternatives, by preferring licences that allow code incorporation over those that only allow static linking, and the latter over the ones that only allow dynamic linking. *Ceteris paribus*, this factor can allow a decision between tied ranking alternatives.

- Documentation

Documentation availability is a constraint. Documentation quality is a factor to decide between similar alternatives (e.g. the availability of on-line tutorials and other training material, or the existence of clear [coding style guidelines](#) for developers), as it clearly facilitates software use and adoption.

- Release Activity

Release activity is a constraint when applied to exclude inactive projects or immature projects. Release activity is also factor to assess the maturity of similar alternatives: a planned release schedule and a public feature roadmap is valued (as is evidence that such compromises are kept).

- Longevity

Project longevity is used as an indirect indicator of each project’s stability and chance of survival. This criterion is only (qualitatively) evaluated in conjunction with release activity and community activity.

- Community

Activity inside and around the project is evaluated in terms of visibility (i.e. ease of access to project website, binaries and source code, documentation, discussion lists or fora, etc.) and activity of the developer community (using indicators available at [ohloh](#), such as number of contributors, number of commits, etc.) and user community involvement, using indirect indicators such number of related papers in [Google scholar](#) (adoption by the academic community), relative number of searches recorded in [Google trends](#) (end-user interest) and number of Q&A recorded in [StackOverflow](#) sites.

- Support

- Active user and developer lists.
- Active public bug tracker.
- Paid support options available.
- SaaS providers available

- Security

- Evidence of Security Backporting.
- Existence of Enterprise (a.k.a. Long Term Support) Versions.
- Evidence of bug severity classification in public bug tracker.

- Integration

- Modularity
- Open Standards
- Collaboration with other products (e.g. reuse of standard libraries).
- Clear identification of software requirements (dependencies), which must also comply with the evaluation criteria.

- Functionality

The functionality sub-criteria depend on the component being evaluated. Functional requirements will be listed in each specific section.

Simply put, the three generic ‘no non-sense’ criteria used in the UK [Open Source Software Options for Government](#) list allow the identification of a manageable set of candidates (typically with least than 5 COTS components) to which to analysis of alternatives is applied.

Informal maturity and deployability criteria

1. Scale: “If something is working, processing, or being used millions of times per day, perhaps used by airlines to process critical transactions across the globe, that is large-scale. That should give you some confidence that it will work at that scale.”

2. Criticality: “If we can find examples of software being used in critical functions, like healthcare — when a patient’s health depends on it — or if it’s used in real-world operations by crime agencies, or in high-security environments, then we can say that this software can be made suitable for critical systems.”
3. Longevity: “If a package has been working successfully for twenty, thirty years, then it can be added to the list, because then it’s not new and unknown. If it has been in operation for many years, we know the risks.”

Source: [Tariq Rashid](#), IT Strategy & Reform, UK Government Cabinet Office

1.4.4 Total Cost of Ownership

A financial estimate of the total cost of ownership (TCO) is beyond the scope of the software evaluation and selection exercise.

Due to the project’s budgetary constraints, license acquisition cost is a binary constraint, not a factor; hardware and infrastructure requirements are similar, regardless of the specific COTS component; and operational costs, such as support and maintenance, are not readily quantifiable for an R&D project with a variable number and type of potential adopters: nevertheless, the selection process does try and minimise the solution’s TCO, through the application of the maturity and deployability criteria.

The selection of COTS, either proprietary or FOSS, is similar in that it must cover a project’s functional requirements. However, the total cost of a given solution is distributed differently.

The licence acquisition cost should be proportional to the provided functionality: thus, in non-free COTS software the least expensive version that provides the necessary and sufficient functionality is generally chosen. (The same principle applies to commercial services, such as technical support and maintenance services, in either FOSS or proprietary solutions).

Given that a budget ceiling on software licence acquisition always exists, the affordable solution may not be the most expandable one (e.g. to support development beyond a particular project’s time frame).

When selecting FOSS solutions, the need to reduce long-term costs, such as staff training or technology migration, justifies the choice of components that provide more flexibility even when they may look ‘over-sized’ in terms of the features or the complexity strictly required for a given project.

1.4.5 Additional References

European Interoperability Framework recommendations on Open Standards

Recommendation 22

When establishing European public services, public administrations should prefer open specifications, taking due account of the coverage of functional needs, maturity and market support.

[...]The level of openness of a formalised specification is an important element in determining the possibility of sharing and reusing software components implementing that specification. [...] If the openness principle is applied in full:

- All stakeholders have the same possibility of contributing to the development of the specification and public review is part of the decision-making process;
- The specification is available for everybody to study;

- Intellectual property rights related to the specification are licensed on FRAND terms [Fair, reasonable and non discriminatory] or on a royalty-free basis in a way that allows implementation in both proprietary and open source software.

Recommendation 21.

Public administrations should use a structured, transparent and objective approach to assessing and selecting formalised specifications. [...]

Definitions

Formalised Specifications : Formalised specifications are either standards pursuant to EU Directive 98/34 or specifications established by ICT industry fora or consortia

Standard : As defined in European legislation (Article 1, paragraph 6, of Directive 98/34/EC), a standard is a technical specification approved by a recognised standardisation body for repeated or continuous application, with which compliance is not compulsory and which is one of the following:

- international standard: a standard adopted by an international standardisation organisation and made available to the public,
- European standard: a standard adopted by a European standardisation body and made available to the public,
- national standard: a standard adopted by a national standardisation body and made available to the public.

Standards developing organisation : A chartered organisation tasked with producing standards and specifications, according to specific, strictly defined requirements, procedures and rules. Standards developing organisations include:

- recognised standardisation bodies such as international standardisation committees such as the International Organisation for Standardisation (ISO), the three European Standard Organisations: the European Committee for Standardisation (CEN), the European Committee for Electrotechnical Standardisation (CENELEC) or the European Telecommunications Standards Institute (ETSI);
- fora and consortia initiatives for standardisation such as the Organisation for the Advancement of Structured Information Standards (OASIS), the World Wide Web Consortium (W3C) or the Internet Engineering Task Force (IETF).

Source: [EIFv2.0]

QSOS maturity criteria and default scores

Legacy : Project's history and heritage

- Age : { 0 : Less than three months; 1 : Between three months and three years; 2 : More than three years }
- History : { 0 : The software has many problems which can be prohibitive; 1 : No major crisis, or unknown history; 2 : Good past experience in crisis management }
- Core team : { 0 : Very few identified core developers; 1 : Few active core developers; 2 : Important and identified core development team }
- Popularity : { 0 : Very few identified users; 1 : Usage can be detected; 2 : Many known users and references }

Activity : Activity inside and around the project

- Contributing community : { 0 : No real community nor activity (forum, mailing lists...); 1 : Community with significant activity; 2 : Strong community with vivid activity in forums, with many contributors and supporters }
- Activity on bugs : { 0 : Low reactivity in forums and mailing lists, or no mention about bugfixes in release }

notes; 1 : Existing activity but without any clearly dened process or with long resolution times; 2 : Strong reactivity based on roles and task assignments }

- Activity on features : { 0 : Few or no new features; 1 : Product’s evolution is led by a dedicated team or by users, but without a clearly stated process; 2 : Feature request process is industrialized, an associated roadmap is available }
- Activity on releases/versions : { 0 : Very low activity on the production or development versions (alpha, beta); 1 : Activity on production or development versions (alpha, beta) with frequent minor corrective versions; 2 : Important activity with frequent corrective versions and planned major versions linked with the roadmap }

Governance : Project’s strategy

- Copyright owners : { 0 : Rights are being held by a few individuals or commercial entities; 1 : Rights are uniformly held by many individuals; 2 : Rights are held by a legal entity or a foundation that the community trust (ex: FSF, Apache, ObjectWeb) }
- Roadmap : { 0 : No roadmap is published; 1 : Roadmap without planning; 2 : Versioned roadmap with planning and delay measurements }
- Project management : { 0 : No clear and apparent project management; 1 : Project managed by an individual or a single commercial entity; 2 : Strong independance of the core team, rights held by a recognized entity }
- Distribution mode : { 0 : Dual distribution with a commercial version along with a functionally limited free one; 1 : Subparts are only available under proprietary license (core, plugins...); 2 : Completely open and free distribution }

Industrialization : Industrialization of the project

- Services : Existing service offerings (support, training, audit...) { 0 : No service offering identified; 1 : Limited service offering (geographically, to a single language, to a single provider or without warranty); 2 : Rich ecosystem of services provided by multiple providers, with guaranteed results }
- Documentation : { 0 : No user documentation; 1 : Documentation exists but is partly obsolete or restricted to one language or to few details; 2 : Documentation up to date, translated and possibly adapted to several target readers (enduser, sysadmin, manager...) }
- Quality assurance process : { 0 : No QA process identified; 1 : Existing QA processes, but they are not formalized or equipped; 2 : QA process based on standard tools and methodologies }
- Source code modification : { 0 : No convenient way to propose source code modifications; 1 : Tools are provided to access and modify the code (eg SCM, forge...) but are not really used by core team to develop the product; 2 : The contributing process is well dened,exposed and respected, it is based on clearly dened roles }

Source: [Atos13]

References

1.4.6 Project Management Infrastructure

A software portfolio was selected to support the project’s technical infrastructure for collaboration, namely:

- a [version control](#) system to keep track of changes to the project’s documentation (and source code);
- a [documentation generator](#) to automate the production of user manuals (and documents such as this one);
- an [issue tracking](#) and generic [project management](#) system.

The following topic provides an overview of the **required components** and **key functionalities** for the technical infrastructure.

It contains an abridged version of section 2.2 in [SWLH11] with minor terminology changes and editing.

A similar but more detailed overview is available in [Foge09] (Chapter 3).

Technical Infrastructure for Collaboration

Since collaboration among widely-distributed contributors is key to OTD (Open Technology Development), projects must establish a project site with the technical infrastructure needed for collaboration. The project site must enable the shared development of the software, test suites, and documentation (including user, installation, administration, and design documentation), though the details of how these occur often vary between projects.

It must be possible for all potential contributors and users to use the tools easily. For example, if security restrictions make it too difficult for people to participate, they will not participate.

Projects should prefer to use widely-used OSS collaboration tools that work well with any standards-compliant web browser. Unusual tools create an unnecessary barrier to entry, as they require users to learn how to use new tools instead of simply contributing. (Even if users have learned how to use a tool, users will be more willing if the tool is widely used because their learning time is amortised.)

OSS tools should be strongly preferred; they can be configured for special needs, tend to be inexpensive to deploy, and tend to be especially good at OTD-style collaboration since they are often used for that purpose.

Maximising access via standards-compliant web browsers increases the ability for others to interact with the product, e.g., they can interact from the field.

Contractors must expect that this technical infrastructure means that the government and other contractors will have continuous access to intermediate progress. This transparency is by design.

Key Functions

The central project site must support the collaboration of ongoing improvements and should provide the following functions:

1. *Web Site*. The central project site must provide a single starting point for those interested in the project, enabling people to learn about the project and find all related information. This is normally a web site with a simple fixed URL.
2. *Bug and Feature Tracking*. The central project site must provide a mechanism for users to submit bug reports and feature requests, and for developers to determine how (or if) to resolve them.

This is often implemented through specialised tools such Bugzilla, Trac, or Redmine, but other tools (such as wikis) can be used. There may need to be a special process for reporting security vulnerabilities to prevent their disclosure before a repair is available.

3. *Version Control System (VCS)*. The central project site must provide a mechanism for tracking changes, including at least the software and often test suites and some documentation. It should at least provide a method for seeing and tracking the “main development branch” and each major release. The VCS must make it possible to see who made each change, when, and what the change was.

There are two major types of VCS systems: centralised (e.g., [Subversion](#) aka SVN) and distributed (e.g., [Git](#) and [Mercurial](#)). Distributed systems (such as [Git](#)) have significant advantages and should be preferred for VCS of OTD projects.

4. *Community interaction*. The central project site must provide a mechanism for users and developers to discuss issues. Mailing lists and wikis tend to be easier to use. Be sure to archive discussions in a way that later participants can find them, or important discussions will be lost.
5. *Release downloads*. The central project site must provide a mechanism for download of major releases.

Public access and classification control

Where possible, determine which components can be released to the public as OSS ahead-of-time, and establish the project outside in the public from the beginning.

There is no legal requirement wait until the project is “feature complete” before this release occurs, and if it is to be public anyway, the sooner it is publicly released the better.

Some components are classified, and thus their development may only take place on systems authorised for classified processing.

Where possible, divide components based on their sensitivity to limit what must be protected by security classification control.

Projects should seriously consider using a distributed VCS (such as “git”) where there may be varying levels of classification. Distributed VCS software make it much easier to create separate external branches and later provide them to others for merging if approval is granted.

Hosting

Each project must determine if it will be based on some existing collaboration hosting service that provides pre-configured functionality, or if it will establish its own system, obtain the necessary tools, and host the project itself. Whatever the decision about hosting, you must be able to compete and change who does hosting support down the line. Do not be locked into a single supplier.

A key criterion for the evaluation of hosting services is to determine how difficult it would be to copy the data (e.g., bug reports, source code, etc.) elsewhere so that if the hosting service is inadequate, the project can easily move.

Small projects are often well-served by using an existing collaboration hosting service, as they cannot justify the resources to specially configure their infrastructure.

Ideally, if a project is to be publicly available as OSS, it should be established as a public project before the design is created.

Source: [SWLH11]

1.4.7 Version Control System

Requirements

The selected version control system is [Git](#).

Sourcecode hosting sites, such as [Github](#) or [BitBucket](#), provide an easily deployable and inexpensive alternative to in-house hosting.

Rationale

(See overview in *Project Management Infrastructure*)

Version Control or revision control is the management of of changes to documents, computer programs, large web sites, and other collections of information. The set of files under version control is kept in a *repository*.

The *Version Control System* (VCS) is the application responsible for keeping track of the successive versions of a repository.

Without a repository under version control, project information quickly gets scattered and duplicated over file systems and e-mail attachments, rendering it inconsistent and unmanageable. If coupled with clear operating procedures, a VCS can vastly simplify the management of changes to project documentation and source code.

In this project, a version control is required:

- to manage the technical documentation (requirements, analysis, design, code and API documentation);
- to manage the source code of the different components;
- to manage the end user documentation production and translation.

Additional requirements are:

- The documentation and source code repository must be under the same VCS.
- The documentation and source code repository should be integrated with the project management / issue tracking system.

Analysis of alternatives

The following table contains a comparison of 4 version control products compatible with the established *COTS selection constraints*:

- Bazaar
- Git
- Mercurial
- and Subversion

	Software	Bazaar	Git	Mercurial (HG)	Subversion (SVN)
Licence	OSI Licence EUPL Compatibility	GPLv2 Dynamic Linking	GPLv2 Dynamic Linking	GPLv2 Dynamic Linking	Apache Yes
Hosting Services (examples)	https://bitbucket.org https://github.com/ https://launchpad.net/ http://sourceforge.net/	No No Yes Yes	Yes Yes No Yes	Yes No No Yes	No Experimental No Yes
General features	Repository model Concurrency model Storage method Scope of change Revision IDs	Distributed Merge Snapshot Tree Pseudorandom	Distributed Merge Snapshot Tree SHA-1 hashes	Distributed Merge Changeset Tree Numbers, SHA-1 hashes	Client-server Merge or lock Changeset and Snapshot Tree Numbers
Network Protocols	SSH HTTP/HTTPS	Yes Yes	Yes Yes	Yes Yes	Yes Yes
User Interfaces (examples)	Web ...via Redmine or Trac CLI GUI	Yes Yes Yes TortoiseBzr	Yes Yes Yes TortoiseGit	Yes Yes Yes TortoiseHg	Yes Yes Yes TortoiseSVN
IDE integration (examples)	Eclipse VisualStudio	BzrEclipse, QbzrEclipse Bzr-visualstudio	Egit MS Visual Studio Tools for Git Git Source Control Provider	Mercurial Eclipse HgSccPackage	Subclipse Subversive AnkhSVN, VisualSVN
Features	Atomic commits Merge tracking File renames Event hooks Signed Revisions (OpenPGP) End of line conversions Unicode filename support	Yes Yes Yes Yes Yes Yes Yes	Yes Yes (Similarity based) Yes Yes Yes Yes	Yes Yes Yes Yes Yes Yes (OS dependent)	Yes Yes Yes Yes No Yes Yes

Table legend

Feature	Description
Repository model	In a client-server model, users access a master repository via a client; typically, their local machines hold only a working copy of a project tree. Changes in one working copy must be committed to the master repository before they are propagated to other users. In a distributed model, repositories act as peers, and users typically have a local repository with version history available, in addition to their working copies.

Concurrency model Describes how changes to the working copies are managed to prevent simultaneous edits from causing nonsensical data in the repository. In a lock model, changes are disallowed until the user requests and receives an exclusive lock on the file from the master repository. In a merge model, users may freely edit files, but are informed of possible conflicts upon checking their changes into the repository, whereupon the version control system may merge changes on both sides, or let the user decide when conflicts arise.

Source: [version control software comparison](#) (adapted)

Evaluation

The major difference between the products is the centralised ([Subversion](#)) or distributed ([Bazaar](#), [Git](#), [Mercurial](#)) nature of the repository. Decentralised systems are relatively recent (after 2005), but have gained wide acceptance, as they allow a more flexible project organisation (e.g. collaboration within smaller teams that contribute to a larger project) and to not require that every user is connected to a single online repository (on the internet or corporate intranet). Decentralised systems can also support ‘centralised-like’ workflows, simply by defining which is the canonical (authoritative) repository, where the ‘official’ or approved versions are kept.

Feature-wise, a clear “winner” does not emerge from the evaluation:

- Online hosting services exist for all the products (i.e. it is possible to store the repository on the web).
- All systems support secure network protocols such as SSL or HTTPS (besides various proprietary protocols).
- Various user interfaces are available: web interfaces (including integration with issue tracking systems such as Redmine or Trac), bash-like command line interfaces (CLI) and various native or 3rd-party graphical user interfaces (GUI).
- For software development purposes, all version control systems can be coupled to integrated development environments (IDE), using plug-ins. Examples are listed for two common multilanguage IDEs: [Eclipse](#) and Microsoft [Visual Studio](#).
- Major features are similar among products (though neither the commands nor the typical workflow are).

The final selection is made by evaluating software adoption in real-world development projects.

Real-world use

Basic information on developer community activity is available through the following [Ohloh indicators on version control systems](#).

Software adoption is evaluated using a simple criterion:

- Which is the version control system used by the different open-source COTS evaluated for this project?

Type ↓	Software ↓	Bazaar	Git	Mercurial (HG)	Subversion (SVN)
Operating System	Linux kernel		Yes		
Database	PostgreSQL		Yes		
	PostGIS		Yes		Yes
HTTP Server	Apache HTTP Server				Yes
Java Application Server	Apache Tomcat				Yes
GIS (library)	Proj.4				Yes
GIS (library)	GEOS		Yes		Yes
GIS (library)	GDAL				Yes
GIS (library)	GeoTools				Yes
GIS (Server)	GeoServer		Yes		
GIS (library)	OpenLayers		Yes		Yes
GIS (desktop)	QuantumGIS		Yes		
Version Control	Git		Yes		
	Subversion				Yes
Version Control GUI	TortoiseGit		Yes		
	TortoiseSVN				Yes
Issue Tracking	Redmine		Yes	Yes	Yes
	Trac		Yes		
IDE	Eclipse		Yes		Yes
Code Editor	JEdit		Yes		Yes
Diff tools	Kdiff3		Yes		
	Kompare				Yes
Document generators	Doxygen		Yes		
	Sphinx		Yes		
	Pandoc		Yes		

A similar generic trend is clear in the [results of the Eclipse Community Survey 2012](#).

Conclusion

Based on the software adoption results, [Git](#) is clearly the recommended version control system.

1.4.8 Documentation Generator

Requirements

The selected documentation tool is [Sphinx](#).

If strictly required, developer documentation may be generated from source code using [Doxygen](#).

If required for dissemination purposes, file conversion into common open and proprietary formats can be accomplished with [Pandoc](#).

[ReadTheDocs](#) (either through the online service or using a local deployment) can be used for automated documentation generation using [Sphinx](#) and [reStructuredText](#) documentation stored in a [Git](#) repository.

Rationale

The main objective is to find a tool adequate for user documentation, that:

- Is easy to work with, by non-technical writers
- Can support the translation process (from EN to PT, CY, BG, etc.)
- Can generate outputs in different formats (HTML, PDF)
- Can produce print quality documents, using different stylesheets
- Can also support the creation of developer documentation from source code (e.g. API documentation).

All documentation must be kept under version control: differences or changes between versions must be readily identifiable. (For example, if the English version of a user manual is updated, changes are required in the localised versions, in the specific files, paragraphs or lines that changed).

Fast identification of changes and differences is a built-in feature of the *version control system* if the documentation is stored in plain text files. This excludes the use of binary formats, such as Microsoft Word documents ([DOC](#)), or zipped XML formats, such as ISO/IEC 26300:2006 OpenDocument ([ODT](#)) or Microsoft Office Open XML ([DOCX](#)). (Conversion tools from and to these formats must nevertheless be available).

By using plain text files, the documentation content can be produced using a simple text editor (i.e. does not require dedicated tools such as word processors or desktop publishing tools). However, a long or steep learning curve should be avoided, which excludes markup languages such as [DocBook](#) or [LaTeX](#), that are overly complex.

A *lightweight markup language* (such as [Markdown](#), [reStructuredText](#) or [Wiki Markup](#)) is thus required: “*simple syntax, designed to be easy for a human to enter with a simple text editor, and easy to read in its raw form*”.

Other basic feature requirements are:

- separation of content (structured text) and presentation (formatting styles),
- support for style sheet languages such as Cascading Style Sheets (CSS) and/or the Extensible Stylesheet Language (XSL),
- support for HTML/XML templating engines in the automated documentation build processes, and LaTeX templates for printed documentation (as a common intermediate step for generating print-quality PDF output).

Analysis of alternatives

The following table contains a comparison of 5 documentation generators:

- [ApiGen](#)
- [Doxygen](#)
- [Javadoc](#)
- [ROBODoc](#)
- and [Sphinx](#)

	Software	ApiGen	Doxygen	Javadoc	ROBODoc	Sphinx
Licence	OSI Licence	New BSD	GPLv2	GPLv2	GPLv3	BSD
	EUPL Compatibility	Yes	Dynamic Linking	Dynamic Linking	No	Yes
	Output Formats					
Must	HTML	Yes	Yes	Yes	Yes	Yes
Should	PDF	-	Yes	via Doclets	-	Yes
Should	LaTeX	-	Yes	via Doclets	Yes	Yes
Could	Compiled HTML Help	-	Yes	via Doclets	-	Yes
	Supported Languages					
Should	Java	-	Yes	Yes	Yes	via javasphinx
Should	JavaScript	-	via perldoxygen	-	Yes	Yes
Should	Python	-	Yes	-	-	Yes
Could	C/C++	-	Yes	-	Yes	Yes
Could	C#	-	Yes	-	-	-
Could	Perl	-	-	-	Yes	-
Could	PHP	Yes	Yes	-	Yes	via phpdomain
Could	Ruby	-	-	-	Yes	via rubydomain

Evaluation:

- Support for specific programming languages – i.e technical documentation automatically generated from source code – is shown in the results table above.
- For mere syntax highlighting, Sphinx uses the [Pygments](#) library (under BSD licence) that supports all the listed programming languages.
- ROBODoc was excluded on the basis of *licence compatibility*.
- ROBODoc and Javadoc are not under active development – see [ohloh](#).
- Javadoc is typically used for Java code documentation and is not really a general purpose documentation generator.
- ApiGen and Javadoc are dominated alternatives when compared to Doxygen and Sphinx.

Real-world use

Basic information on developer community activity is available through the following [Ohloh indicators on documentation tools](#).

Since the only non-dominated alternatives are [Doxygen](#) and [Sphinx](#), software adoption and real-world use cases were only investigated for these products.

Extensive lists of [projects using Doxygen](#) and [projects using Sphinx](#) are available online.

The following conclusions can be drawn:

- Doxygen is mainly used for developer documentation, such as API documentation, although some open-source projects also use it for user documentation (PostgreSQL, PostGIS, GEOS, GDAL).

- Sphinx is extensively used for user manuals, tutorials and books, but can also be used for developer documentation, either through direct support or through additional ‘language domains’, such as [javadoc](#).
- Doxygen output can be incorporated in Sphinx documentation, via the [BREATHE](#) extension (under BSD licence).

The following links illustrate the use of [Sphinx](#) for different (technical) purposes:

- [Documentation Generators](#)¹.
- [Managing Multilingual Documentation with Sphinx and Transifex](#)
- [Documenting Multiple Programming Language APIs with Sphinx](#)
- [Writing Technical Documentation with Sphinx, Paver, and Cog](#)
- [Brandon Rhodes PyCon 2013 Sphinx tutorial source code](#)

Conclusions

- [Sphinx](#) is better for user documentation.
- The [PANDOC](#) library (under GPLv2 licence) can be used to convert [RST](#) to a large number of formats (including [DOCX](#) and [ODT](#)).
- The [javadoc](#) extension to the Sphinx documentation system adds support for documenting Java projects. It includes a Java domain for writing documentation manually and a [javadoc-apidoc](#) utility which will automatically generate API documentation from existing Javadoc markup.
- [Doxygen](#) can complement Sphinx (for developer documentation and programming language not supported by Sphinx).
- [ReadTheDocs.org](#) is an online hosting service for [Sphinx](#) documentation. The service can be connected to a repository under version control, using either [Bazaar](#), [Git](#), [Mercurial](#) or [Subversion](#). Web hooks are supported (when changes are committed, the documentation is automatically rebuilt), also with two online repository hosting services, [GitHub](#) and [BitBucket](#). The [ReadTheDocs](#) software is available under an MIT licence and a local deployment can be used (if required).

1.4.9 Issue Tracking System

Requirements

Note: Important note

This document refers to the outcome of the preliminary software selection process.

Following the procurement process, and based upon the preference expressed by the infrastructure provider and accepted by the development contractor, a decision was reached to use [Redmine](#) instead of [Trac](#).

The project management and issue tracking system proposed for this project is [Trac](#).

Alternatives such as [Redmine](#) or [Bugzilla](#) can be adopted, if required or preferred by the infrastructure provider or already in use by the development contractor.

Online issue tracking systems, namely those associated with sourcecode hosting sites such as [github](#), [bitbucket](#) or [sourceforge](#) are an easily deployable and inexpensive alternative.

¹ An overview and list of bookmarks which is itself written in [reStructuredText](#), build in Sphinx, stored on a [BitBucket](#) repository under [Mercurial](#) version control and published online using [ReadTheDocs.org](#).

Rationale

(See overview in *Project Management Infrastructure*)

Generically, an ‘issue’ is any task that must be accomplished within a given time frame, has a responsible party/user and a predefined or an *ad-hoc* workflow. ‘Issues’ can be software feature requests, bug reports, etc., but can also be RFC (Request for Comments) on existing documents, meeting agendas, etc..

The issue tracking system is the software product responsible for maintaining a list of (prioritised) open issues and dispatching them to the responsible parties based on the selected workflow.

Some issue tracking systems have features similar to project management applications and can be used for generic project management, in addition to their main role in supporting the software development process and interactions with end-users.

If the product is to be used also as the project’s initial website and project management tool, then two complementary features are useful:

- a wiki to allow non-technical (authorised) users to add, modify or delete the online content, using a *lightweight markup language*.
- a forum (message board) to centralise and archive project-related messages and discussions (a customised type of ‘issue’ can be used for this purpose).

Besides the generic *COTS selection constraints*, two integration requirements apply:

- **Git must** be one of the supported version control system, as most tickets/issues will likely refer to or affect documentation and source code
- **PostgreSQL should** be one of the supported database back-ends: this is not a functional requirement, but consolidates the software portfolio and allows the reuse of limited DBA and infrastructure resources (should in-house hosting be adopted).

A *puzzle assembly strategy* is also applied to the software dependencies of each product: if it requires a software component that is also required by another component, then a smaller software portfolio can be defined (requiring less training, less administration skills, etc. and reducing the TCO).

Other features are discussed in the next section.

Analysis of alternatives

The following table contains a comparison of 4 issue tracking products compatible with the established *COTS selection constraints*:

- Bugzilla
- Redmine
- Request Tracker
- and Trac

	Software	Bugzilla	Redmine	Request Tracker	Trac
Licence	OSI Licence	MPL	GPLv2	GPLv2	(modified) BSD
	EUPL compatible	Yes	Dynamic Linking	Dynamic Linking	Yes
	Implementation language	Perl	Ruby on Rails	Perl	Python
Hosting	SaaS	Yes	Yes	Unknown	Yes
	Database				
Must	PostgreSQL	Yes	Yes	Yes	Yes
Could	SQLite	Yes	Yes	Yes	Yes
Could	MySQL	Yes	Yes	Yes	Yes
	Revision control system				
Should	Subversion	Yes	Yes	Yes	Yes
Must	Git	Yes	Yes	No	Yes
Could	Mercurial	Yes	Yes	No	Yes
Could	Bazaar	Yes	Yes	No	Yes
	Authentication methods				
Must	Form based	Yes	Yes	Yes	Yes
Must	LDAP	Yes	Yes	Yes	Yes
Could	OpenID	Yes	Yes	Yes	Yes
	Input				
Must	Web	Yes	Yes	Yes	Yes
Should	Mylyn (Eclipse)	Yes	Yes	No	Yes
Should	Email	Yes	Yes	Yes	Yes
Could	SOAP	Yes	No	No	No
Could	REST	Yes	Yes	Yes	No
	Notification				
Must	Email	Yes	Yes	Yes	Yes
Could	RSS	Yes	Yes	Yes	Yes
Could	Atom	Yes	Yes	No	No
Could	Twitter	No	Yes	No	No
	Other Features				
Must	Unicode support	Yes	Yes	Yes	Yes
Must	Full-text search	Yes	Yes	Yes	Yes
Must	Custom Fields	Yes	Yes	Yes	Yes
Must	Customizable workflow	Yes	Yes	Yes	Yes
Should	Multiple Projects	Yes	Yes	Yes	Partially
Should	Wiki	Add-on, Media Wiki	Yes	Add-on, Media Wiki	Yes
Should	Forum / Blog	No	Yes	No	No
Could	Test planning integration	Yes	Yes	No	Yes

Table legend

Feature	Description
Implementation language	Indicates the programming language used for product implementation, which has implications of the product's software dependencies.
Cloud services	Indicates if cloud-based service providers are available, that would allow the product to be used as SaaS (Software as a Service).
Database:	Lists supported FOSS database back-ends (other proprietary DBMS such as Oracle are also supported)
Revision control system	Lists the supported version control systems.
Authentication methods	Basic form-based authentication will be required for external users (the various project partners, the development contractor); LDAP for internal users.
Input methods	Interactive web-based is required for end-users; e-mail can be useful for the technical team; Mylyn integration can be useful if the development team uses the Eclipse IDE (e.g. for Java development) or simply if the Eclipse Platform is used for project and task management by team members.
Notification methods	E-mail based notification is required, other options may be nice-to-have.
Basic features	Unicode support, the possibility to add/customise fields, and full-text searches are basic requirements (that were mainly used to exclude software options from the candidate set).
Multiple projects	Support for multiple projects is not required for the project, but allows the reuse of the same platform at organisation level.
Wiki	Collaborative editing of project pages is not required for issue tracking, but may be a convenient feature, specially for non-technical documentation (and end-users).
Forum / blog	Again, this feature is not required for issue tracking, but can provide a convenient platform for team members and end-users. The forum should be monitored by technical elements, in order to identify potential issues/bugs that should raise a formal issue tracking workflow. Similarly, a blog can be used to provide end-users with (non-technical) feedback on new features, bugs resolved, etc.

Evaluation

The functional evaluation is not conclusive: except for [RT](#), the alternatives meet the established criteria.¹²³

All products require the following components:

- Database back-end: either [PostgreSQL](#) or [MySQL](#) ([SQLite](#) can only be used for very small projects, due to concurrency limitations).
- Apache HTTP Server (or equivalent): only [Trac](#) is shipped with a built-in web server.
- A [Mail Transfer Agent](#) (such as [Exim](#) or [Postfix](#)) or simply access to an e-mail account on an [SMTP](#) server (to send email notifications).

Additional dependencies are related to each product implementation language:

- Bugzilla requires [Perl](#) and various Perl modules.
- Trac requires [Python](#) and several Python modules.

¹ The <http://landfill.bugzilla.org/> site can be explored for demonstration purposes.

² The <http://www.redmine.org/> site is, itself, powered by Redmine. The <http://demo.redmine.org/> site can be explored for demonstration purposes.

³ Demonstration sites for various Trac versions are available at <http://trac.edgewall.org/wiki/SandBox>.

- Redmine runs on [Ruby on Rails](#) and requires various Rails modules.

From a software portfolio point-of-view, Trac might be preferable since [Sphinx](#), the selected *documentation generator* is also a Python application with similar dependencies (as is [ReadTheDocs](#) that can be used to automate a server-side documentation build process).

The three candidate have different wiki capabilities:

- Bugzilla has a [MediaWiki](#) add-on available. [MediaWiki](#) (used by [Wikipedia](#)) requires a web server such as the [Apache HTTP server](#), PHP 5.3.2+, and a database back-end such as [PostgreSQL](#). [Wiki Markup](#) is used as a markup language: [reStructuredText](#) is not currently supported. Basically, wiki support for Bugzilla would require an addition component (PHP) and wouldn't allow the use of the same markup language in all the project's documents and pages.
- Trac provides wiki functionality with support for two markup languages: [WikiFormatting](#) or [reStructuredText](#) (the latter requires two additional Python modules: [docutils](#) and [pygments](#)).
- Redmine also provides wiki functionality, using [Textile](#). Support for [reStructuredText](#) is only partial, through a plug-in, that requires the installation of [Python](#) + [docutils](#) or [pandoc](#).

Again, from an integration (with the documentation generator) and software portfolio point-of-view, Trac is preferable.

[Trac](#) does not support multiple projects, which can be a limitation at organisational level (e.g. the same platform/installation might be needed for distinct subprojects). An unsupported community contributed plug-in ([SimpleMultiProject](#)) exists that extends the basic fields and reports available in Trac by including a “project” category with associated milestones.

Real-world use

Basic information on developer community activity is available through the following [Ohloh indicators on issue tracking systems](#).

The following table is analogous to *version control system adoption table* and identifies the issue tracking system used by the different open-source COTS evaluated for this project.

Type ↓	Software ↓	Bugzilla	Redmine	Trac	Other
Operating System	Linux kernel	Yes			
Database	PostgreSQL PostGIS			Yes	Message board Yes
HTTP Server	Apache HTTP Server	Yes			Yes
Java Application Server	Apache Tomcat	Yes			Yes
GIS (library)	Proj.4			Yes	Yes
GIS (library)	GEOS			Yes	Yes
GIS (library)	GDAL			Yes	Yes
GIS (library)	GeoTools				JIRA
GIS (Server)	GeoServer				JIRA
GIS (library)	OpenLayers			Yes	Yes
GIS (desktop)	QuantumGIS		Yes		
Version Control GUI	TortoiseGit TortoiseSVN				code.google.com code.google.com
IDE	Eclipse	Yes			Yes
Code Editor	JUnit				sourceforge (allura)
Diff tools	Kdiff3 Kompare				sourceforge (allura) KDE bugtracking
Document generators	Doxygen Sphinx Pandoc	Yes			Bitbucket Github

FOSS development projects provide most of the available real-world use examples listed at each project's web site.

The [Bugzilla users community](#) is notoriously large and includes high-profile projects such as the [Linux Kernel](#), [Mozilla](#), [Eclipse](#) and the [Apache Software Foundation](#) projects.

The [Redmine users community](#) is much smaller than either Bugzilla's or Trac's, although [Google trends on issue tracking systems](#) show an increasing level of end-user interest in the past years. The [stackoverflow](#) Q&A site shows a similar trend.

The [Trac users community](#) includes well-known projects such as [WordPress](#), [Plone](#), [Django](#), [jQuery](#) or [Transifex](#). Trac is also used by most [OSGEO](#) and other FOSS GIS projects - such as [Proj](#), [GDAL](#), [PostGIS](#), [GRASS](#), [OpenLayers](#), [JOSM](#) or [OpenStreetMap](#). Trac is also the [sourceforge classic tracker](#) (the current sourceforge 2.0 version runs on the [Allura](#) platform).

An additional source of information on software adoption is available at <https://launchpad.net/bugs/bugtrackers>. As of June 2013, Trac was the most common issue tracker, used by 400 out of about 1700 listed projects.

Conclusions

- [Bugzilla](#) is best used as a dedicated bug tracking system and would require additional components (e.g. [MediaWiki](#)) to support a generic project site.
- [Redmine](#) has a versatile and user-friendly interface, for administration and end-users. It has an increasing and active community, which is nevertheless much smaller than the user and developer communities of either Bugzilla or Trac.
- All things considered (features, integration, community size and activity), [Trac](#) is an equilibrated option.
- It should be noticed that many projects currently use online issue tracking services managed by service providers (SaaS), instead of maintaining an in-house issue tracking platform.
- Similarly, most [sourcecode hosting services](#) (e.g. [Github](#), [Bitbucket](#) ou [sourceforge](#)) provide issue tracking facilities.

1.4.10 Questionnaire Management System

Requirements

Questionnaire management systems

A review of the functionality available in Blaise 4.5, the Survey Processing System developed by Statistics Netherlands, was made in order to try and guarantee that the fundamental requirements were duly identified.

Data model basics

Field definition A field definition includes a field name and a field type that defines valid values. Usually it will have question text. It can have a description to document the field, a field tag, and special attributes.

Rules There are four types of rules: routing instructions, edit checks (hard and soft), computations, and layout instructions.

- Routing instructions describe the data entry order of the fields and the conditions under which they will be eligible.

For computer-assisted interviewing, the route specifies the order and conditions in which the fields are asked.

- Edit checks determine whether a specified statement is true for the values of the fields involved. If it is false, the instruction will generate an error. What will be done with the error depends on the application at hand. Two kinds of edits are supported. The CHECK instruction defines a hard error, something that must be fixed before the form can be considered clean. The SIGNAL instruction defines a soft error, which is a

possible problem. It can be suppressed or the values of the involved fields may be changed. A CHECK is the default.

- Computations determine proper routes to process fields, carry out complex checks, or derive values.
- Layout instructions determine the placement of data entry fields displayed in the Data Entry Program.

Fields

A field is the basic element of the data set in Blaise. Fields can have specific types of definitions. Examples of definition types include strings, numbers, or dates. You can create your own user-defined types of fields.

You specify fields in the FIELDS section:

```
DATAMODEL Commute1 "The National Commuter Survey, first example."
FIELDS
  Name "What is your name?" : STRING[20]
  Town "In which town do you live?" : STRING[20]
  Gender "Are you male or female?" :
    (Male, Female)
  MarStat "What is your marital status?" :
    (NevMarr "Never married",
    Married "Married",
    Divorced "Divorced",
    Widowed "Widowed")
  Children "How many children have you given birth to?" : 0..10
  Age "What is your age?" : 0..120
```

In its most simple form, each field definition consists of an identifying name and a specification of the valid values. A longer text between double quotes will usually be inserted between the name and the value definition. This text may serve to state a question, as description, or to document the field.

The example above shows string, numeric and enumerated data variables.

Rules

```
DATAMODEL Commute1 "The National Commuter Survey, first example."
FIELDS
  <---see example above-->
RULES
  Name
  Town
  Gender
  MarStat
  Age
```

The five fields will be processed in this order. Field names can also be asked, subject to a condition:

```
IF (Gender = Female) AND (Age > 12) THEN
  Children
ENDIF
```

This means that the field Children will only be processed if the field Gender has the value Female and Age is greater than 12.

Checking

Checks are conditions that have to be satisfied. You can state the check in terms of what the correct relationship between fields should be:

```
MarStat = NevMarr "he/she is too young to be married!"
```

The specification instructs the system to check whether the field MarStat has the value NevMarr. If not, then the edit is invoked. You can attach text between double quotes to a condition. Such a text will be used as an error message if the condition is not satisfied.

Checks can be subject to conditions:

```
IF (Age < 15)
  "If age of respondent is less than 15" THEN
    MarStat = NevMarr
    "then he/she is too young to be married!"
ENDIF
```

The check MarStat = NevMarr will only be carried out if the field Age has a value less than 15. The application will reject entries in which people younger than 15 years are married.

Blocks

The following specification contains two block definitions: the block BPerson and the block BWork:

```
DATAMODEL Commute2 "The National Commuter Survey, example 2."
  BLOCK BPerson "Demographic data of respondent"
    FIELDS
      <-- see example above -->
    RULES
      <-- see example above -->
  ENDBLOCK

  BLOCK BWork "Data about work and commuting"
    FIELDS
      Working "Do you have a paid job?" : (Yes, No)
      Descrip "Short description of your job" : STRING[40]
      Distance "What is the distance to your work (in km)?" : 0..300
      Travel "How do you travel to your work?" : SET [3] OF
        (NoTravel "Do not travel, work at home",
         PubTrans "Public bus, tram or metro",
         Train "Train",
         Car "Car or motor cycle",
         Bicycle "Bicycle",
         Walk "Walk",
         Other "Other means of transportation")
      Commuter "Are you a commuter?" : (Yes, No)
    RULES
      Working
      IF Working = Yes THEN
        Descrip Distance Travel
      ENDIF

      IF (Working = Yes) and (Distance > 10) THEN
        Commuter := Yes
```

(continues on next page)

(continued from previous page)

```

    ELSE
        Commuter := No
    ENDIF
ENDBLOCK

FIELDS
    Person : BPerson
    Work : BWork;
RULES
    Person
    Work
ENDMODEL

```

A block is like a sub-data model with fields and rules. A block is a special kind of field type. You define a field with the block name as field type:

```

FIELDS
    Person: BPerson

```

Field tags

Questions on paper questionnaires are often identified using numbers. A field tag is specified between parentheses after the field name and before the first text:

```

FieldName (FieldTag) "Text" / "Description" : FieldType

```

A field tag may consist of letters and digits, although usually only digits will be used (to represent question numbers). The following example illustrates the use of field tags:

```

Working (101) "Do you have a paid job?" : (Yes, No)
Descrip (102) "What is your job description" : STRING[40]

```

Field types

String type

A string field accepts any text as value, provided the length of the text does not exceed the specified maximum length. An example:

```

Name "What is your name? " : STRING[30]

```

You can test the contents of, or make an assignment to, a string field using single quote marks:

```

IF SectionName = 'Household' THEN
    Labell := 'Household Roster'
ENDIF

```

It is common to test or assign values between string fields:

```

IF Name1 <> Name2 THEN
    Name1 := Name2
ENDIF

```

To test for an empty string you can use either the word `EMPTY` or two single quote marks with no space between them:

```
IF Name = EMPTY THEN...
IF Name = ' ' THEN
```

OPEN type

For long, open-ended text responses of variable length you can use the `OPEN` type. `OPEN` type fields are useful when the interviewer must record verbatim answers from respondents.

Integer type

Integer fields can be defined in several ways:

```
Age1 "What is your age?" "Age of respondent" : 0..120
Age2 "What is your age?" "Age of respondent" : INTEGER[3]
```

Decimal or real type

To define a field that accepts decimal numbers, you can use the following type declarations:

```
Ticket1 "How much did you pay for your train ticket?" : 0.00..10.00
Ticket2 "How much did you pay for your train ticket?" : REAL[5, 2]
Ticket3 "How much did you pay for your train ticket?" : REAL[5]
```

Enumerated type or precode

An enumerated field can take as a value one of the items in a list of items. The item is known as a category identifier. A simple example is:

```
Gender "What is your gender?" : (Male, Female)
```

Set type

To allow a respondent to choose more than one item from a list of answers, use a `SET` field. A `SET` field may also be known as a multiple precode or code-all-that-apply. Consider the following example:

```
Travel "How do you travel to your work?" : SET [3] OF
  (NoTravel "Do not travel, work at home",
   PubTrans "Public bus, tram or metro",
   Train "Train",
   Car "Car or motor cycle",
   Bicycle "Bicycle",
   Walk "Walk",
   Other "Other means of transport")
```

The format is the same as that of an enumerated field with the addition of the reserved words `SET [3] OF`. This indicates that, at most, three different category values can be selected. By specifying `SET OF` without a number between brackets, you allow all items to be picked from the list.

Stored values of enumerated and set fields

The values in enumerated fields and SET fields are stored as code numbers. The first item in the list is assigned code 1, the second gets code 2, and so on. You can specify your own code numbers to overrule this coding scheme. You do that by adding numbers in parentheses between category names and category texts. Consider the following example:

```
Travel "How do you travel to your work?" : SET [3] OF
  (NoTravel (0) "Do not travel, work at home"
  PubTrans "Public bus, tram or metro",
  Train "Train",
  Car (4) "Car or motorcycle",
  Bicycle "Bicycle",
  Walk "Walk",
  Other (9) "Other means of transport")
```

This means that NoTravel is coded as 0. Subsequent values are coded in ascending order until a new code is encountered. Here PubTrans is coded as 1 and Train as 2. The codes of the other values are 4 for Car, 5 for Bicycle, 6 for Walk, and 9 for Other.

Date and time type

A date field is a special field type that only accepts dates as values. For example:

```
Birth "What is your date of birth?" : DATETYPE
TimeTravel "At what time do you start work?" : TIMETYPE
```

Classification type

To perform hierarchical coding, a classification type is needed.

Arrayed fields

An arrayed field defines a series of fields of the same type.

TYPE section

With the TYPE section, you can define reusable type definitions (or just types), then use them in later sections:

```
TYPE
  TTravel = SET [3] OF
    (NoTravel "Do not travel, work at home",
    PubTrans "Public bus, tram or metro",
    Train "Train",
    Car "Car or motor cycle",
    Bicycle "Bicycle",
    Walk "Walk",
    Other "Other means of transport")
FIELDS
  TravelNow "How do you travel to your work at the moment?" : TTravel
  TravelThen "How did you travel to your work one year ago?" : TTravel
RULES
```

(continues on next page)

(continued from previous page)

```
TravelNow
TravelThen
```

A Block may be considered a FieldType.

Type libraries

Survey organisations will find it profitable to standardise the types that are used in two or more surveys. This will make programming quicker and make it easier to compare survey results. Interviewers and data editors will also appreciate the common standards between surveys. Types can be shared by many applications in two different ways.

They can be incorporated with an INCLUDE statement or they can be stored in a pre-compiled type library.

The type library is prepared just like a data model except that the first key word is LIBRARY. A very simple type library:

```
LIBRARY MyLib
TYPE
    TYesNo = (Yes, No)
    TMarStat = (Single, Married, Divorced)
ENDLIBRARY
```

To use the type library from a data model, name the type library in a LIBRARIES section. Then, for any field in the data model, refer to any type in the usual manner. For example:

```
DATAMODEL UsesLib
LIBRARIES MyLib
FIELDS
    WillYou : TYesNo
    Married : TMarStat
ENDMODEL
```

Languages

Blaise supports the use of different languages. This is particularly important if you want to interview people speaking different languages. During interviewing, the interviewer can change to a different language with a function key. This requires you to specify the question texts in all languages you might intend to use. The format for a field definition for two languages is as follows:

```
FieldName    "Text1" "Text2" /
             "Description1" "Description2" : FieldType
```

You specify the question texts (Text1 and Text2) in different languages, and specify the descriptions after the slash in the same order. Here is a concrete example:

```
Name    "What is your name?" "Wat is uw naam?" /
        "Name of respondent" "Naam van de respondent" : STRING[30]
```

Enhancing texts

Hard spacing and line breaking, font, font size, colour, bold, and underline. Variable text fills. Auxiliary fields (screen labels, etc.)

Routing rules

Route field methods

Blaise has four methods: CLASSIFY, ASK, SHOW and KEEP. ASK is the default.

CLASSIFY The CLASSIFY routing method puts a classification type field on the route, to show and edit it.

ASK The method ASK shows the field on the screen so the user can enter a value.

SHOW The method SHOW shows the field and its value on the screen. In the FormPane (page), the user will not be able to land on it and will not have the opportunity to change the value. You typically SHOW a field to give information to the user, as with a label in the page. It is possible to use the mouse to land on a show field, but you can not change its value.

KEEP The method KEEP means that the field is not shown on the screen. The user will not be aware that this field exists. The value of the field will be stored in the Blaise data file if the field is defined in the FIELDS section, but the KEEP is considered to be at the end of the RULES section.

If you SHOW or KEEP a field, its value must be computed. If a field is computed but otherwise not mentioned in the RULES, the KEEP method will automatically be applied at the point of computation.

Preventing return to some questions

You can use the KEEP method to prevent users from returning to some questions after they have been answered. This is called a wall. This might be done for reasons of confidentiality.

Rationale

Analysis of alternatives

1.4.11 Dynamic charts on web browser usage statistics

Warning: This version requires internet access.

Note: Daily usage values are represented in the fourth and fifth of the following charts. There is a clear weekly pattern of use that reflects the slower rate of adoption of non-Microsoft browsers in enterprise environments: on weekends, users that have [IE](#) in their workplace switch to [Chrome](#) or [Safari](#) in their domestic computers. The final chart, with an example of operating systems usage statistics (for internet browsing) in Bulgaria, clearly shows the slower rate of technology turn-over in work environments. Windows XP is still has a very large usage share specially on workdays (note the time-series anomaly around Christmas holidays), although the end of security [support for Windows XP](#) in April 2014 will soon force companies and domestic users to upgrade their computers or operating systems.

1.4.12 OSGEO list of webmapping tools

Note: For completeness sake, the original of the following list was retrieved from http://wiki.osgeo.org/wiki/A_comprehensive_list_of_webmapping_toolkits.

Software	Last release (date)	Description	Licence	Map engines and protocols	Language
AtlasMapper	1.3 (2012-11) Inactive	The AtlasMapper is a cross platform JavaScript/Java mapping application that allows a catalogue of map layers to be easily browsed, layered, re-styled and compared side-by-side in a web browser.	GPLv3	WMS, NCWMS, ArcGIS Server, XYZ / OSM	Java, JavaScript (GetExt, Openlayers, Ext JS)
CartoWeb	3.5.0 (2008-09-04) Inactive	CartoWeb is a comprehensive and ready-to-use Web-GIS as well as a convenient framework for building advanced and customised applications.	GPLv2.0+	See MapServer	PHP
Chameleon	2.4.1 (2006.09.06) Inactive	Chameleon is a distributed, highly configurable, environment for developing Web Mapping applications. It is built on MapServer as the core mapping engine and works with all MapServer supported data formats through a regular MAP file.	MIT	See MapServer , WMS	PHP
collective.geo.plone (Plone Maps)	Active	The goal is to provide a comprehensive set of tools to manage and publish geospatial data into the Plone , using existing and proven technologies as much as possible.	GPLv2.0	(See OpenLayers , Polymaps)	Python, JavaScript, OpenLayers
dbox	0.9a1 (2006-05-08) Inactive	dbox is really a collection of DHTML-based libraries for building highly interactive web-based mapping applications. The tools are meant to work directly with the MapServer web mapping system. They provide relatively autonomous functionality without restricting overall design. In fact, they were designed to be used with old fashioned elements like tables.	?	MapServer	JavaScript, DHTML
Dracones	1.1.3 (2010-12-06) Inactive	Dracones is a MapServer -based web mapping framework. Core components: A lightweight map widget, with a smooth navigation interface; Map layers with interactive behaviours, like mouse selection or tooltip (mouseover) information; Flexible query/extension mechanism; Handy other services like map image export, and history navigation	BSD	See MapServer , WMS	PHP, Python
Flexlayers	Unknown (2009) Inactive	Partial port of the OpenLayers mapping API from its native JavaScript to ActionScript 3	LGPLv3	WMS, WMS-C, and WFS	ActionScript 3
Fusion	2.2.0 (2011-06-01) Active	Fusion is a web-mapping application development framework for MapGuide OS and MapServer built primarily in JavaScript. It allows non-spatial web developers to build rich mapping applications quickly and easily.	MIT	See MapServer , MapGuide	PHP, JavaScript (Openlayers)
GeoExt	1.1 (2011.12.22) Active	GeoExt is a JavaScript Toolkit for Rich Web Mapping Applications that combines the OpenLayers library (for mapping components) with the Ext JS JavaScript Framework. See for example GeoExplorer .	BSD-3-Clause	See OpenLayers	JavaScript, (OpenLayers, Ext JS)

Continued on next page

Table 2 – continued from previous page

geojsp	Unknown (Unknown) No public source-forge	geojsp is an open source (GPL) component that integrates geographic elements in your business intelligence infrastructure (geo-BI): thematic maps; flow maps; indicators; graphics (bar charts, pie chart, etc.)	GPL	See Open-Layers	Java
GeoMondrian	1.0 (Unknown) Last commit in 2012-04	GeoMondrian is a spatially-enabled version of Pentaho Analysis Services Mondrian . It provides a consistent integration of spatial objects into the OLAP data cube structure, instead of fetching them from a separate spatial database, web service or GIS file.	EPL	See Open-Layers	Java
GeoMOOSE	2.6.1 (2012-02-12) Active: OSGEO Incubation	GeoMOOSE is a Web Client JavaScript Framework for displaying distributed cartographic data. GeoMOOSE has a number of strengths including modularity, configurability, and delivers a number of core functionalities in its packages. GeoMOOSE is built MapServer , OpenLayers and the Dojo Toolkit.	MIT	MapServer , see Open-Layers	JavaScript, PHP
GeoPrisma	1.6.3 (2012-11-27) No information	GeoPrisma is a Web mapping application featuring Access Control (Permissions) to geospatial data (Resources). A single Resource can be served by multiple different geodata servers (Services), such as WMS, TileCache, MapServer, FeatureServer, etc.	BSD-3-clause	see Open-Layers	PHP
GeoShield	0.3.0 (2012-04) Immature	GeoShield is a project born to offer a centralised way to define security access-control to geoservices. It acts like a proxy, intercepting all the communications between clients and OGC compliant services	BSD-3-clause	See GeoServer , WMS, WFS, WPS, SOS	Java
GeoWebCache	4.0 (2013-07-09) Active	GeoWebCache is a Java web application used to cache map tiles coming from a variety of sources such as OGC Web Map Service (WMS). It implements various service interfaces (such as WMS-C, WMTS, TMS, Google Maps KML, Virtual Earth) in order to accelerate and optimise map image delivery. It can also recombine tiles to work with regular WMS clients.	LGPL	WMS, WMS-C, TMS, WMTS	Java
GisClient	3.5.3 (2012-06-2012) Inactive?	Web authoring tool configurator for MapServer that enables the user both to build Mapfiles and to provide OpenLayers maps.	GPLv3	See Mapserver	AJAX, Javascript, PHP/MapScript

Continued on next page

Table 2 – continued from previous page

GPAAMP (Global Protected Areas Assessment and Monitoring Pilot)	0.7 (2010-10) Inactive	Supporting viewing and download services, it functions primarily as a means for visualising information from disparate sources delivered through standards-based web services. It is based on the stack ExtJS, OpenLayers and GeoExt.	Apache Licence 2.0	See Open-Layers	Javascript (GeoExt, Open-Layers, ExtJs)
Heron Mapping Client	0.73 (2013-06-07) Active	GeoExt is a toolkit that combines OpenLayers with Ext JS to help build desktop style GIS apps on the web with JavaScript. The Heron MC leverages these frameworks by providing high-level components and a convention to quickly assemble applications merely through configuration (“Look ma no programming”).	GPLv3	See Open-Layers	Javascript (GeoExt, Open-Layers, Ext JS)
HSLayers	3.4.0 (2012-11-02) Inactive	HSLayers combines OpenLayers (for mapping part) and ExtJS (for the graphical user interface) for complete desktop-like WebGIS toolkit in JavaScript. It also has several server-side little script for support of the web interface.	GPL	See Open-Layers , MapServer , WCS and WFS	JavaScript, Python
i3geo	Unknown (Unknown) Active	i3geo provides a set of navigation tools, generation of analysis, sharing and generation of maps on demand.	GPLv2.0+	See MapServer	PHP, JavaScript
ka-Map	1.0 (2007-05-02) Inactive	ka-Map is an open source project that is aimed at providing a JavaScript API for developing highly interactive web-mapping interfaces using features available in modern web browsers.	MapServer licence	MapServer	PHP, JavaScript
leaflet	0.6 (2013-06-26) Active	Leaflet is a modern, lightweight JavaScript library for making tile-based interactive maps for both desktop and mobile web browsers.	BSD-2-Clause	tilled layers, WMS	JavaScript
Mapbender	3.0.0 (2013-05-29) Active: OSGeo project	Mapbender is the back office software and client framework for spatial data infrastructures. It provides a data model and web based interfaces for displaying, navigating and querying OGC compliant map services. It is based on the frameworks Symfony2 , jQuery and OpenLayers	GPL, BSD	WMS , WFS	PHP, JavaScript
MapFish	2.2 (2011-06-27) Inactive	MapFish is a flexible and complete framework for building rich web-mapping applications. It emphasises high productivity, and high-quality development.	BSD-3-Clause	WMS , WFS	Python, JavaScript
MapProxy	1.5.0 (2012-12-05) Active	It caches, accelerates and transforms data from existing map services and serves any desktop or web GIS client. MapProxy is a tile cache solution, but also offers many new and innovative features like full support for WMS clients.	ASLv2.0	WMS , TMS	Python

Continued on next page

Table 2 – continued from previous page

MapQuery	0.1 (2011-07-28) Inactive	MapQuery is a jQuery plug-in that you can use to add mapping to your website. Whether you quickly want to add a simple map to a page, or build a feature rich web application, MapQuery is just the thing you need.	MIT	See Open-Layers	JavaScript
Modest Maps	Unknown (Unknown) Very low activity	Modest Maps is a small, extensible, and free library for designers and developers who want to use interactive maps in their own projects. It provides a core set of features in a tight, clean package with plenty of hooks for additional functionality	BSD	?	JavaScript
OpenLayers	2.13.1 (2013-07-09) Active 3.0.0alpha (2013-07-12) Active	OpenLayers makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds.	BSD-2-Clause	ArcGIS Server, GML, Google Maps, KML, MapGuide, MapServer, TMS, Bing Maps, WFS, WMS, etc.	JavaScript
OpenScales	2.2 (2012-07-11) Active	OpenScales is an open source (LGPL) mapping framework written in ActionScript 3 and Flex that enables developers to build Rich Internet Mapping Applications.	Lesser GPL	GML, KML, OSM, TMS, WFS, WMS and others.	ActionScript, Flex, AIR
p.mapper	4.3.1 (2013-04-04) Active	The p.mapper framework is intended to offer broad functionality and multiple configurations in order to facilitate the setup of a MapServer application based on PHP/MapScript.	GPLv2+	MapServer	JavaScript, PHP
Polymaps	2.5.1 (2011-06-14) Inactive	Polymaps is a free JavaScript library for making dynamic, interactive maps in modern web browsers. It provides speedy display of multi-zoom datasets over maps, and supports a variety of visual presentations for tiled vector data, in addition to the usual cartography from OpenStreetMap, CloudMade, Bing, and other providers of image-based web maps.	BSD 3-clause	Bing Maps, Cloud-made, OSM	JavaScript
ReadyMap Web SDK	?	ReadyMap SDK is a free JavaScript library for embedding 3D maps in a web page. Build 3D maps using ReadyMap's API, or turn your OpenLayers or Leaflet maps into 3D globes.	GPLv3	See Openlayers	JavaScript (osgjs, jQuery)

Continued on next page

Table 2 – continued from previous page

TileCache	2.11 (2010-10-15) Inactive	TileCache provides a Python-based WMS-C server, with pluggable caching mechanisms and rendering backends. In the simplest use case, TileCache requires only write access to a disk, the ability to run Python CGI scripts, and a WMS you want to be cached.	BSD-3-Clause	Mapnik, MapServer, WMS	Python
TileMill	0.10.1 (2012-10-10) Active	TileMill is an application for making beautiful maps. Whether you're a journalist, web designer, researcher, or seasoned cartographer, TileMill is the design studio you need to create compelling, interactive maps.	BSD	Mapnik	CSS-like map styling language
Thematic Mapping Engine	2009 In-active	TME enables you to visualise global statistics on Google Earth. The engine returns a KMZ file that you can open in Google Earth or download to your computer.	GPLv3	KML	JavaScript , PHP

1.4.13 Web Server and Servlet Container

Requirements

The selected COTS are:

- [Apache HTTP Server](#), as a web server
- [Apache Tomcat](#), as a servlet container

Should a full J2EE application server be required, the JBoss Application Server is an adequate alternative.

1.4.14 Rationale

The web server will deliver static resources (static pages, such as the documentation, JavaScript, CSS, images, etc.), provide a security layer between the application server(s) and the client applications (e.g. browsers), allow for load balancing (if required) and act as a reverse proxy (routing incoming requests to specific application servers).

A web server is required by several of the other software components. For example:

- [Trac](#), the issue tracking system, is a Python application that requires a [WSGI](#), [FastCGI](#) or [AJP](#)-capable web server (preferably WSGI), in production environments.
- [Geoserver](#), the geospatial data server, is a Java application that requires a Java servlet container application.
- The map visualisation component is based on [JavaScript](#) libraries that support the web mapping applications.
- [Limesurvey_](#), is a PHP application for questionnaire management that can provide the required functionality for microdata collection.
- phpPgAdmin is a PHP application that provides a web-based administration console for PostgreSQL (similar to phpMyAdmin for MySQL): although is not required, it is very useful for data management purposes.
- Similarly, publication of the project documentation pages requires a web server, and the Sphinx, the documentation generator may also be deployed to such a server.
- ... and so on.

In most cases, the final selection of the web server to be used in a production environment will be made by each organisation's ICT department or provider. In the scope of this project, the objective is to select an adequate option, though not necessarily the one that a specific MS may choose to use in its particular production environment.

In this perspective, the choice of web server is similar to the choice of supported operating systems or supported web browsers:

- it must be a common, well-known, stable, enterprise-strength and commercially supported COTS;
- it must comply with the *Constraints* set on licence compatibility, portability and acquisition cost;
- it must be part of a reference platform on which the other COTS components (e.g. GeoServer, SDMX tools, etc.) have been tested on by each of their development teams.
- and it must support the range of foreseeable project requirements.

1.4.15 Analysis of alternatives

The following table presents an edited excerpt of an existing [comparison of web server software](#), from which all options non-compliant with the defined constraints were excluded.

Some of the listed options are grayed out:

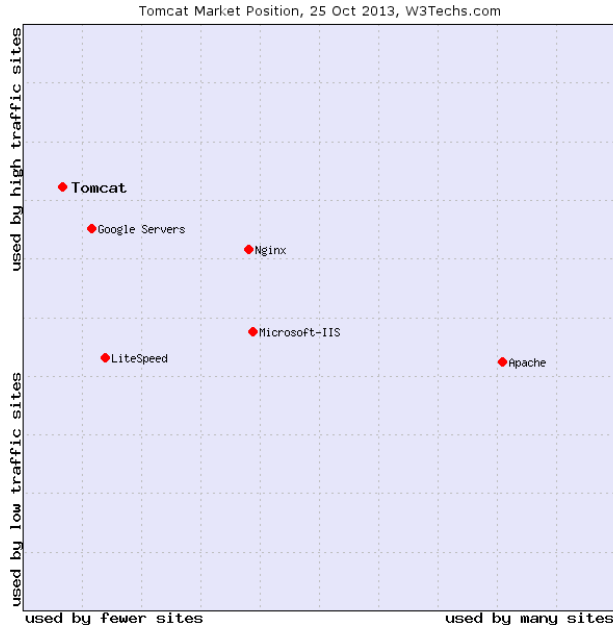
- AOLServer and Xitami are no longer actively developed;
- the Cherokee Windows built is broken in the last versions (i.e. there is no Windows version currently available)
- httpd, NaviServer, or Mongoose have small development teams, and are dominated alternatives (as to maturity and adoption) when compared to Apache, nginx or Jetty.

Software	License	Version	Release date	SaaS	Basic access authentication	Digest access authentication	SSL/TLS https	CGI	FCGI	Python WSGI	JavaServlets	IPv6	Windows	Linux	OS X
AOLServer	Mozilla	4.5.1	2009-01-06	Yes	Yes	No	Yes	Yes	No	No	No	Unknown	Yes	Yes	Yes
Apache HTTP Server	Apache-2.0	2.4.6	2013-07-22	Yes	Yes	Yes	Yes	Yes	Yes	Via modules	To AppServer via AJP	Yes	Yes	Yes	Yes
Apache Tomcat	Apache-2.0	7.0.42	2013-01-16	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes
Cherokee HTTP Server	GPL-2+	1.2.102	2013-01-17	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Broken	Yes	Yes
Jetty	Apache-2.0 EPL-1.0	9.0.0	2013-03-08	Yes	Yes	Yes	Yes	Yes	Unknown	No	Yes	Unknown	Yes	Yes	Yes
lighttpd	BSD-3-Clause	1.4.32	2012-11-21	Yes	Yes	Yes	Yes	Yes	Yes	No	To AppServer via AJP	Yes	Yes	Yes	Yes
Mongoose	GPL-2	3.8	2013-07-16	Unknown	Yes	Yes	Yes	Yes	Unknown	Unknown	Unknown	No	Yes	Yes	Yes
NaviServer	GPL2.0+ MPL-1.1	4.99.5	2013-06-08	Yes	Yes	No	Yes	Yes	No	No	No	Unknown	Yes	Yes	Yes
nginx	BSD-2-Clause	1.5.3	2013-07-30	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Xitami	GPL	5.0a0	2009-01-19	Yes	Yes	Unknown	paid version	Yes	Unknown	No	Unknown	Unknown	Yes	Yes	Yes

For the remaining competitors, the adoption statistics made available by [w3techs.com](#) and based on the top 10 million websites in the [Alexa](#) 3 month average ranking, are as follows:

- 65.4% of sites use **Apache HTTP Server**;
- 14.5% of sites use **nginx** (rank #3)
- 0.4% of sites use **Apache Tomcat** (rank #6)
- 0.1% of sites use **Jetty** (rank #9)

The following image presents the average market position of the three top FOSS options (as well as the two most frequent proprietary options, Microsoft IIS and LiteSpeed, and the unpublished custom Linux-based Web server that Google uses for its online services).



Conclusions

Apache HTTP Server is the dominant web server. It can support Python applications via **mod_wsgi** (the Python WSGI adapter module for Apache), as well as CGI and FastCGI applications.

The Apache HTTP Server also supports AJP, the Apache JServ Protocol: it can thus act as a reverse proxy (routing the requests that reach the web server to a Java application server, for example Apache Tomcat).

Given that a Java web container will be required (by components such as GeoServer), an alternative would simply be to use Apache Tomcat (or Jetty) as a web server. However, globally, this solution is less flexible and, during the project, a WSGI-compliant web server would still be required for the project management infrastructure.

In conclusion, the use of Apache HTTP Server plus Apache Tomcat as a Java web container provides adequate and flexible deployment options.

Note that it is assumed that only a Java web container will be required, not a full Java EE Application Server.

Should such requirement exist, Apache TomEE (that reuses Tomcat as its web container part) might be used. Alternative solutions are GlassFish or JBoss (recently renamed as WildFly).

The available usage statistics¹ for JBoss or GlassFish are based on survey data and eventual survey bias is not documented. However, JBoss consistently appears with a larger usage share in the surveys, for comparable functionality (see for example the small [comparative of Tomcat, JBoss, GlassFish and Jetty](#), published by the developers of JRebel).

JBoss incorporates a Tomcat 6.0 derivative (fork) as its web container, and

1.4.16 Operating Systems

Requirements

Commercial Off-the-shelf (COTS) components **must** be available for:

¹ See, for example: <http://plumbr.eu/blog/most-popular-application-servers>, <http://zeroturnaround.com/rebellabs/java-ee-productivity-report-2011/#servers>, <http://zeroturnaround.com/rebellabs/developer-productivity-report-2012-java-tools-tech-devs-and-data/4/>.

- Microsoft Windows Server and Windows 7
- [Ubuntu](#) 12.04.02 LTS
- [CentOS](#) 6

Software components to be developed **must** be supported/tested/accepted in:

- Microsoft Windows Server and Windows 7
- [Ubuntu](#) 12.04.02 LTS **or** [CentOS](#) 6

Rationale

Portability

Cross-platform compatibility is a non-functional project requirement: the objective is to facilitate the sharing, reuse and future improvement of the software solution to be developed (and of existing software components it may incorporate).

For the end users, the majority of the interfaces will probably be web-based user interfaces (WUI): thus web browser compatibility will be a more stringent requirement than operating system compatibility.

Nevertheless, when software is available for different platforms, potential users may install and test the server-side components in their local machines, thus promoting reuse and adoption.

Analysis of alternatives

The selected software components must have cross-platform compatibility and must be available for and supported on the most common operating systems. Thus, it must be determined which are the *most common operating systems*.

Except for proprietary operating systems (e.g. Microsoft Windows and Mac OS X) no reliable market share statistics exist. Most FOSS (Free and Open Source Software) operating systems can be freely downloaded and installed and do not require any kind of user registration: thus, only indirect estimates of their use are available.

- For the server segment, the [W3Techs report on web servers operating systems](#) provides a rough estimate:
 - 65% are UNIX systems, 50% of which are Linux distributions
 - 35% are Microsoft Windows systems
 - less than 0.1% are Mac OS systems.

Among the 3 major [Linux families](#), the relative proportion is:

- 57% for the ‘Debian family’, including [Debian](#) (approximately 58% of the Debian-based distributions) and [Ubuntu](#) (42%)
- 39% for the ‘Red Hat family’, including [CentOS](#) (70%), [Red Hat](#) (23%) and [Fedora](#) (7%);
- 1.8% for the ‘Slackware family’, that includes [SUSE](#) (99.9%).

These values do not include servers that are not exposed to the Internet (e.g. database servers in corporate intranets).

- For the desktop/mobile user segment, the [Wikipedia request statistics](#) provides another rough estimate:
 - 68% are Microsoft Windows systems
 - 20% are Apple Mac OS or iOS systems
 - 7% are Linux systems (including Android)

If only non-mobile Linux systems are considered, the available sample again reveals the 3 major Linux families:

- 96% for the ‘Debian family’, represented by Ubuntu (98% of the Debian-based distributions), Mint and Debian (both with less than 1%);
- 3% for the ‘Red Hat family’, divided among Fedora (59%), Mandriva (22%), CentOS (10%) and Red Hat (9%);
- less than 2% for the ‘Slackware family’, represented by SUSE (99.9%) and OpenSUSE (0.1%).

These values reflect a specific type of use (Wikipedia queries) by (mostly) domestic users. However, the usage ranking of operating systems is similar to that obtained in other indirect sources like [Google Trends on operating systems](#) or the [StatOWL Operating System Market Share](#) report. Based on the StatOWL’s data, the relative proportion of the 7 most frequent Linux distributions is:

- 97% for the ‘Debian family’, including Ubuntu (98% of the Debian-based distributions) Mint (1%) and Debian (<1%);
- 2% for the ‘Red Hat family’, including Fedora (40%), CentOS (40%) and Red Hat (20%);
- 1% for the ‘Slackware family’, represented only by SUSE.

The global patterns that may be inferred from these data sources are:

- The operating systems’ usage share is completely different on the server segment and the user segment.
- On the server segment, Microsoft Windows and Linux distributions such as Debian, Ubuntu and CentOS are common.
- On the user segment, proprietary systems (Microsoft Windows, Mac OS X) are dominant. The (largely) dominant Linux distribution is Ubuntu.

These patterns support the following conclusions:

- Server-side software components **should** be available for:
 - Microsoft Windows Server 2008 R2
 - The ‘Debian family’ and ‘Red Hat family’ Linux distributions;
 - Mac OS X support is not a requirement, but may be a nice-to-have feature for developers.
- Client-side software components **should** be available for:
 - Microsoft Windows (at least for Windows 7);
 - The ‘Debian family’ distributions, at least for Ubuntu 12.04.2 LTS (a Debian-based distribution that has Long Term Support releases with longer support life-cycles);
 - The ‘Red Hat family’ distributions, at least for CentOS 6 (a free Red Hat Enterprise Linux clone that has a longer life-cycle than Fedora);
 - Mac OS X (although this OS is uncommon in the European public administration).

These conclusions can further be synthesised in the following **must** requirements:

- Software components **must** be available for:
 - Microsoft Windows Server 2008 R2 and Windows 7
 - Ubuntu 12.04.02 LTS
 - CentOS 6
- Software components to be developed **must** be tested/accepted in:
 - Microsoft Windows Server 2012 and Windows 7

- Ubuntu 12.04.02 LTS **or** CentOS 6

Regarding the two selected Linux operating systems:

- Software distribution in Ubuntu is based on DEB packages (as in all of the ‘Debian family’); in CentOS, it is based on RPM packages (as in all of the ‘Red Hat family’). Software packaging requirements are distinct for the two types (DEB and RPM): no packaging requirements will be defined for the project, other than the common .tar.gz (compressed tar archives) file type for source code distribution (when applicable/required).
- Both operating systems are gratis, but commercial technical support is available for Ubuntu (e.g. by its main developer, [Canonical](#)) and CentOS is basically a free clone of the commercial Red Hat distribution (so any organisation requiring commercial support can acquire and use the [Red Hat](#) distribution).

1.4.17 Database Management System

Requirements

The selected spatial object-relational database management system is [PostgreSQL 9.2+](#) with [PostGIS 2.0+](#).

Rationale

The database management system must comply with the [Constraints](#) set on licence compatibility, portability and acquisition cost.

Compliance with ANSI-SQL:2008 [[ISO.IEC_9075:2008](#)] is a basic technical interoperability requisite.

- In Portugal, this is also a legal requisite for Public Administration IT systems development and procurement as established by the National Regulation on Digital Interoperability (Open Standards). [[RNID12](#)]

Project specific functional requirements (further discussed in the analysis of alternatives) are:

- Support for spatial data types and spatial analysis
- Support for (or coupling to) statistical computing tools
- Support for key-value data storage and/or XML data storage

Analysis of alternatives

Progressive filtering is based on maturity and sustainability criteria that are not specific to this project, but guarantee that the chosen solution can be used at enterprise level in distinct projects, indirectly reducing the cost of support, database administration, etc.

Detailed information on the nature, characteristics and market share of major RDBMS products is available in Section B (Databases) of the Competitive Assessment in the [European Commission’s Report on the Oracle/Sun Microsystems merger](#).

The [proprietary DBMS market share](#) estimates vary. The following ranking is based on annual revenue:

- [Oracle](#) (circa 50%),
- [IBM DB2](#) (circa 20%),
- [Microsoft SQL Server](#) (circa 17%).

Three additional open source RDBMS are identified in the [EC Report](#):

- [MySQL](#)
- [PostgreSQL](#)

- [Ingres](#)

[MySQL](#) (and its various clones and forks, such as [MariaDB](#), [Drizzle](#) and [Percona Server](#)) has the largest market share of open source RDBMS.

Oracle's evaluation of the relative technical merits of the three open source RDBMS is quoted in the [EC Report](#):

The notifying party argues that the database products Ingres and PostgreSQL are also available under open source licenses and are technically superior to MySQL, in particular with regard to higher-end enterprise usage targeting existing Oracle customers. Therefore if any open source database product were able to exercise a competitive constraint on Oracle, it would be Ingres or PostgreSQL rather than MySQL.

—§662

Presently, [Ingres](#) could not be included in the candidate set, as the community version does not appear to be under active development. [Ingres](#) utilises a dual licensing model: the open source (community) version is available without charge to the end users under GPLv2; a proprietary version is commercialised by [Actian](#). The information available at the [Ingres Community](#) site does not provide a clear separation between the the community and the enterprise versions. Furthermore, an analysis the [Ingres commit activity](#) on the [Ingres source code repository](#) indicates that no commits have been made to the main branch in the past 12 months. Regarding the [IngresGeospatial](#) component, the available roadmap also indicates that there is no recent activity on the project.

The [EC report](#) provides an overview of the total cost of ownership (TCO), based on 3rd party evaluations performed in 2009, stating that:

MySQL and PostgreSQL are comparable in terms of price for the smallest deployments, and have by far the lowest prices for the largest. While the price levels of MySQL and PostgreSQL are directly comparable with each other, none of the other [proprietary] vendors comes close to either of these software packages for a large deployment.

—§243 et seq.

[OpenVirtuoso](#), another object relational database management system, meets the minimum maturity criteria and is included in the candidate set of alternatives (OpenVirtuoso is not mentioned in the EC report that focuses on direct competitors to Oracle and MySQL).

The candidate set includes:

- [MySQL](#)
- [PostgreSQL](#)
- and [OpenVirtuoso](#).

Spatial data support

Support for the spatial components is a project-specific requirement that, in practice, allows a decision between the 3 candidates.

The selected RDBMS **must** support spatial data:

- OGC Well-Known Text (WKT) and OGC Well-Known Binary (WKB) datatypes [[ISO_19125-1:2004](#)]
- Geography Markup Language (GML) import/export [[ISO_19136:2007](#)] [[ISO.IEC_13249-3:2011](#)]
- Spatial Reference System transformations [[ISO.IEC_13249-3:2011](#)]
- DE-9IM spatial operators (Clementini/Egenhofer Operators) [[ISO_19125-2:2004](#)]

[MySQL](#) currently lacks a full implementation of the spatial components.

The spatial components of [OpenVirtuoso](#) are only available in the proprietary version of the program.

[PostgreSQL](#) + [PostGIS](#) is the selected option:

- PostGIS is the extension to PostgreSQL that allows spatial objects to be stored in the database, using the OGC WKT and WKB formats. (PostgreSQL is an object relational database that allows the definition of complex user datatypes).
- Spatial data can be exported to GML, the XML grammar defined by OGC as a data modelling and open interchange format for geographic information. GML is used in the EU INSPIRE data models and data exchange specifications. [Inspire].
- Support for the Dimensionally Extended 9 Intersection Model (**DE-9IM**) basically guarantees that all relevant topological predicates (functions to check topological relations between two geometries) are implemented.
- PostGIS also supports **GeoJSON** and **TopoJSON**, that may be useful for map visualisation purposes (e.g. using Javascript libraries such as **D3.js**).
- Finally, PostGIS includes support for GiST-based R-Tree spatial indexes.

Note: If or where only a simple SQL database engine is required, while maintaining support for spatial data and operators, the recommended option is **SQLite + SpatiaLite**.

SpatiaLite is available under an **MPLv1.1** licence, which is fully compatible with EUPLv1.1.

SQLite is under a **Public Domain** licence, that is not an **OSI-approved licence**: that may require the acquisition of licence from the **Hwaci** (depending on the specific legal requirements of a Member State).

Statistical computing support

Another project-specific requirement is also supported by PostgreSQL: the support for statistical analysis.

At least two alternatives exist (see [Conw11] for examples):

- Using PL/R, a loadable procedural language extension that supports writing PostgreSQL functions and triggers in the **GNU R** statistical computing language;
- Accessing the database using **RPostgreSQL**, the **GNU R** interface to the PostgreSQL database system.

Basically this means that, if required:

- The statistical capabilities provided by GNU R can be called by the PostgreSQL database in a way that database programmers are familiar with;
- The data stored in the PostgreSQL database can be accessed by GNU R in a way that statisticians are familiar with.

Implicit in the above discussion in the adoption of **GNU R** (GPLv2) should any ‘out-of-the-ordinary’ statistical procedures be required (e.g. for statistical disclosure control).¹

Key-value and/or XML data storage

Support for either key-value data storage or XML data storage is a (plausible) functional project requirement.

A common data model at national level can be specified based on well-known *a priori* requirements, namely the need to produce the Eurostat THB indicators data cubes and, consequently, to store the appropriate variables at microdata level.

¹ A discussion/comparison of statistical software packages is beyond the scope of this document. In this context, it is also deemed unnecessary as the GNU R Project is the *de facto* standard in statistical computing.

However, it is likely that different countries will need/want to store additional variables regarding victims, traffickers, routes, etc., to comply with national reporting requirements. These needs can not be foreseen and accommodated into a single and canonical relational database model. Two alternatives exist (which may complement each other):

- a [NoSQL](#)-like approach, whereby the additional data is stored as a semi-structured set of key-value pairs;
- an [SDMX](#)-like approach, whereby the structure of the data is described using an XML schema and the data itself is stored in XML structured text.

PostgreSQL provides basic functionality to support either option: key-value data storage using the [hstore](#) module (that defines the `hstore` datatype and functions) and XML data storage using the native XML datatype and functions (that use the [xmllib2](#) parser and toolkit).

Note on portability and support

1. PostgreSQL is included by default in the main Linux distributions (Debian/Ubuntu and Red-Hat/CentOS/Fedora/Scientific), although the version maintained in each Linux distribution's official repository may not be the last one. However, all PostgreSQL releases are available in the PostgreSQL Apt Repository (for the Debian family) and the PostgreSQL Yum Repository (for the Red Hat family).

PostgreSQL is also available for Microsoft Windows and Mac OS X.

2. As with the selected FOSS operating systems, both PostgreSQL and PostGIS are free of charge, but commercial versions and technical support are available for PostgreSQL (*inter alia*, by one of main developers, [EnterpriseDB](#)) and for PostGIS (*inter alia*, by one of the main developers, [Refractions Research](#)).

Real-world use

The following table lists a sample of projects running on PostgreSQL + PostGIS solutions.

The selection is (completely) biased toward spatial data systems: it focuses on national mapping and cadastral (land surveying) agencies, spatial data infrastructures, and security and emergency response systems.

However, spatial data types are more complex than text, numeric or date data types and spatial databases generally support a heavier workload than non-spatial databases. So, even if a given system will not have a major spatial component, these examples can be seen as 'stress testing' results for the proposed solution.

Organization, system or project	References
Eurogeographics (MCA)	2
Gestione Integrata e Interoperativa dei Dati Ambientali (SDI), Italy	34
Geodatastyrelsen (MCA), Denmark	5
Global Disaster Alert and Coordination System (SDI), UN & EC-JRC	6
Global Monitoring for Environment and Security , EC-JRC	7
Institut national de l'information géographique et forestière (MCA), France	8
Instituto Geográfico Nacional (MCA), Spain	9
Landesamt für Vermessung und Geoinformation Bayern (MCA), Germany	10
Maanmittauslaitos (MCA), Finland	11
Ordnance Survey (MCA), UK	12
Paikkatietoikkuna (SDI), Finland	13
Police Crime Map , UK	14
Publieke Dienstverlening op de Kaart (SDI), Netherlands	15
Statens Kartverk (MCA), Norway	1617

MCA: Mapping/Cadastral Agency

SDI: Spatial Data Infrastructure

References

1.4.18 Open Standards in the Portuguese National Regulation on Digital Interoperability

Data formats

Comments

All the listed open standards are recognised as applicable to different components of the software solution, except XMPP.

² URL: http://inspire.jrc.ec.europa.eu/events/conferences/inspire_2011/presentations/workshops/269/ESDIN_D5_2_EuroGeographics_Technical_Architecture.pdf

³ URL: http://www.minambiente.it/export/sites/default/archivio/allegati/INSPIRE_state_of_play_2011_ITALY.pdf

⁴ URL: <http://www.gdmc.nl/zlatanova/Gi4DM2010/gi4dm/Pdf/p116.pdf>

⁵ URL: http://www.eurosdnet.net/workshops/PostGIS/7_Rasmussen_de_Martino_Nielsen_KMS_Denmark.pdf

⁶ URL: <http://meetingorganizer.copernicus.org/EGU2012/EGU2012-7404.pdf>

⁷ URL: <http://rslab.disi.unitn.it/papers/R72-JSTAR-Brunner.pdf>

⁸ URL: <http://postgis.net/2012/10/18/ign>

⁹ URL: http://inspire.jrc.ec.europa.eu/events/conferences/inspire_2012/presentations/143.pdf

¹⁰ URL: http://www.fig.net/pub/fao/floss_cadastre.pdf

¹¹ URL: <http://www.paikkatietoikkuna.fi/web/en/open-spatial-data>

¹² URL: http://www.eurosdnet.net/workshops/PostGIS/8_Bennett_Ordnance_Survey_UK.pdf

¹³ URL: <http://www.oskari.org/trac/wiki/DocumentationBackend>

¹⁴ URL: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/78964/Open_Source_Options_v2_0.pdf

¹⁵ URL: http://www.tudelft.nl/fileadmin/Faculteit/LR/Documenten/Onderwijszaken/lecture_postgis.pdf

¹⁶ URL: <http://joinup.ec.europa.eu/sites/default/files/studies/IDABC-case-study-Norwegian%20Mapping%20Authority-final-version.pdf>

¹⁷ URL: http://www.eurosdnet.net/workshops/PostGIS/6_Pedersen_Statkart_Norway.pdf

Table

Standard	Description	Status	Deadline	Reference
SQL	Structured Query Language	Mandatory	2013.02.06	http://www.w3schools.com/sql/default.asp
PNG	Portable Network Graphics	Recommended	2013.02.06	http://www.w3.org/TR/PNG
SVG	Scalable Vector Graphics	Mandatory	2013.02.06	http://www.w3.org/TR/SVG
XML	Extensible Markup Language	Mandatory	2013.02.06	http://www.w3.org/TR/REC-xml/
XSLT 2.0	XSL Transformations	Mandatory	2013.02.06	http://www.w3.org/TR/xslt20/
XSD	XML Schema Definition	Mandatory	2013.02.06	http://www.w3.org/TR/xmlschema-0/ http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/structures.html http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html
XSL 1.1	Extensible Stylesheet Language	Mandatory	2013.02.06	http://www.w3.org/Style/XSL/
XMPP	Extensible Messaging and Presence Protocol	Mandatory	2013.02.06	http://xmpp.org/rfcs/rfc6120.html
UTF-8	8-bit Unicode Transformation Format	Mandatory	2013.02.06	http://tools.ietf.org/html/rfc3629

Document formats

Comments

All the listed open standards are recognised as applicable to the documentation of the project and the file-based data exchange requirements.

Table

Standard	Description	Status	Deadline	Reference
ODF 1.1	Open Document Format v1.1 (Second Edition)	Mandatory	2013.02.06	http://docs.oasis-open.org/office/v1.1/OS/AvailableDocuments/2013.02.06/OtherDocuments/2014.06.01/ODF-OpenDocument-v1.1.pdf
PDF 1.7	Portable Document Format	Mandatory	2013.02.06	www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf
XML 1.0	Extensible Markup Language	Mandatory	2013.02.06	www.w3.org/TR/REC-xml/
HTML 4.01	Hypertext Markup Language	Mandatory	2013.02.06	www.w3.org/TR/html401/

Web interface technologies

Comments

Todo: Atom ? RSS ?

Table

Standard	Description	Status	Deadline	Reference
ATOM 1.0	Atom Syndication Format 1.0	Mandatory	2013.02.06	http://tools.ietf.org/html/rfc4287
CalDav	Calendaring Extensions to web DAV (CalDAV)	Mandatory	2014.06.01	http://tools.ietf.org/html/rfc4791
CSS2.1	Cascading Style Sheets 2.1	Mandatory	2013.02.06	http://www.w3.org/TR/REC-CSS2
HTML 4.01	Hypertext Markup Language	Mandatory	2013.02.06	http://www.w3.org/TR/html401/
HTTP/1.1	Hypertext Transfer Protocol	Mandatory	2013.02.06	http://tools.ietf.org/html/rfc2616
HTTPS	Hypertext Transfer Protocol Secure	Mandatory	2013.02.06	http://tools.ietf.org/html/rfc2818
Javascript 1.5	Javascript 1.5	Recommended		http://www.ecma-international.org/publications/files/ECMA-ST-ARCH/ECMA-262,%203rd%20edition,%20December%201999.pdf
WCAG 2.0 – level «A»	Web Content Accessibility Guidelines 2.0 – level «A»	Mandatory	2013.02.06	http://www.w3.org/TR/WCAG20
WCAG 2.0 – level «AA»	Web Content Accessibility Guidelines 2.0 – level «AA»	Mandatory	2013.02.06	http://www.w3.org/TR/WCAG20
WCAG 2.0 – level «AA» or «AAA»	Web Content Accessibility Guidelines 2.0 – level «AA» or «AAA»	Recommended		http://www.w3.org/TR/WCAG20
WCAG 2.0 – level «AAA»	Web Content Accessibility Guidelines 2.0 – level «AAA»	Recommended		http://www.w3.org/TR/WCAG20
WebDAV	Web Distributed Authoring and Versioning Access Control Protocol	Recommended		http://tools.ietf.org/html/rfc3744
XML 1.0	Extensible Markup Language	Mandatory	2013.02.06	http://www.w3.org/TR/REC-xml/
XSL v1.1	XML stylesheet language XSL v1.1	Mandatory	2013.02.06	http://www.w3.org/TR/2006/REC-xsl11-20061205

Streaming protocols

Comments

Not applicable.

Table

Standard	Description	Status	Deadline	Reference
RTSP	Real Time Streaming Protocol	Mandatory	2013.02.06	www.ietf.org/rfc/rfc2326.txt

Electronic mail protocols**Comments**

Not directly required. Obligations shall be complied with, where applicable.

Table

Standard	Description	Status	Deadline	Reference
IMAP 4	Internet Message Access Protocol	Mandatory	2013.02.06	tools.ietf.org/html/rfc3501
MIME	RFC 2045, 2046, 2047 – Multipurpose Internet Mail Extensions	Mandatory	2013.02.06	tools.ietf.org/html/
POP3	RFC 1939 – Post Office Protocol	Mandatory	2013.02.06	www.ietf.org/rfc/rfc1939.txt
POP3S, IMAPS	RFC 2595 Using TLS with IMAP, POP3 and ACAP	Recommended	2013.02.06	tools.ietf.org/html/rfc2595
SMTP	Simple Mail Transfer Protocol – RFC 5321	Mandatory	2013.02.06	www.ietf.org/rfc/rfc2821.txt
SMTPS	RFC 3207 SMTP Service Extension for Secure SMTP over Transport Layer Security - http://www.ietf.org/rfc/rfc3207.txt	Recommended	2013.02.06	www.ietf.org/rfc/rfc3207.txt

Geographical information systems**Comments**

WFS and WMS are applicable and required. WCS is not required in this project.

No requirement has currently been identified that strictly requires WPS.

WMTS-client applications will be required, WMTS-server applications probably will not.

Table

Standard	Description	Status	Deadline	Reference
WCS	Web Coverage Service	Mandatory	2013.02.06	www.opengeospatial.org/standards/wcs
WFS	Web Feature Service	Mandatory	2013.02.06	www.opengeospatial.org/standards/wfs
WMS	Web Map Service	Mandatory	2013.02.06	www.opengeospatial.org/standards/wms
WPS	Web Processing Service	Mandatory	2013.02.06	www.opengeospatial.org/standards/wps

Communication protocols

Comments

Not directly required. Recommendation shall be followed, where applicable.

Table

Standard	Description	Status	Deadline	Reference
IPV6	Internet Protocol, Version 6 (IPv6)	Recommended	2013.02.06	http://tools.ietf.org/html/rfc2460

Network security

Comments

Obligation shall be complied with, where applicable.

Table

Standard	Description	Status	Deadline	Reference
TLS 1.0	Transport Layer Security	Mandatory	2014.01.01	http://tools.ietf.org/html/rfc2246

Data exchange and service integration and orchestration

Comments

Warning: Note on REST?

Table

Standard	Description	Status	Deadline	Reference
BPMN 2.0	Business Process Model and Notation	Recommended		http://www.omg.org/spec/BPMN/2.0
HTTP/1.1	Hypertext Transfer Protocol	Mandatory	2013-02-06	http://tools.ietf.org/html/rfc2616
HTTPS	Hypertext Transfer Protocol Secure	Mandatory	2013-02-06	http://tools.ietf.org/html/rfc2818
LDAP	Lightweight Directory Access Protocol	Mandatory	2013-02-06	http://www.ietf.org/rfc/rfc1777.txt
SAML 2.0	Security Assertion Markup Language 2.0	Mandatory	2013-02-06	http://docs.oasis-open.org/security/saml/v2.0/
SOAP 1.1	Simple Object Access Protocol 1.1	Mandatory	2013-02-06	http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
WS-Addressing 1.0	Web Services Addressing	Mandatory	2013-02-06	http://www.w3.org/TR/ws-addr-core/
WS-RM 1.1	WS-Reliable Messaging 1.1	Recommended		http://docs.oasis-open.org/ws-rx/wsrn/200702/wsrn-1.1-spec-os-01.pdf
WS-Security 1.2	Web Services Security 1.2	Recommended		http://docs.oasis-open.org/ws-sx/ws-security-policy/v1.2/ws-securitypolicy.html
WS-Security Username Token Profile 1.0	WS-Security Username Token Profile 1.0	Recommended		http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

1.4.19 Geospatial Data Server

Requirements

The selected geospatial data server is [GeoServer 2.3.2+](#)

Iff required:

- [GeoWebCache 1.3.0+](#) can be deployed as a standalone or integrated map tiles server.
- [GeoNetwork 2.10.0+](#) (which itself includes an integrated [GeoServer](#) server), can be used a spatial data catalogue system.

Rationale

The purpose of the geodata server component is simply to provide a layer of standardised geoweb services between the database and the various possible client applications (GIS desktop clients, web mapping applications, other geodata servers, etc.).

The geospatial data server must comply with the [Constraints](#) set on licence compatibility, portability and acquisition cost.

It must also be compatible with / supported on the software stack that was previously selected:

- Microsoft Windows, Ubuntu, CentOS (see [Operating Systems](#))
- PostgreSQL + PostGIS (see [Database Management System](#))

Compliance with Open Geospatial Consortium (OGC) WMS, WFS and WFS-T standards is a basic technical interoperability requisite.

- In Portugal, compliance with the OGC Web Feature Service (WFS) Interface Standard and the OGC Web Map Service (WMS) Interface Standard is also a legal requisite established by the National Regulation on Digital Interoperability (Open Standards). [RNID12]
- In the European Union, and under the provisions of the INSPIRE Directive, Member States must share spatial datasets relevant for environmental policy (34 spatial data themes, that include *inter alia* geographical names, administrative units, addresses, and population and demography).

Member States are also required to create web services for accessing these datasets, and these must be OGC-compliant services (see [INSPIRE Implementing Rules](#) for details)

Analysis of alternatives

Again, progressive filtering was based on maturity and sustainability criteria.

To simplify the triage of candidate solutions, only [OSGEO Foundation](#) projects were considered. The [OSGEO incubation process](#) is essentially a peer-reviewed evaluation inspired by the CMMI maturity levels [CMMI10]: graduated projects comply with an adequate [set of criteria](#) focused on maturity and sustainability aspects.

The candidate set includes:

- [deegree](#) (initial release in 2002 OSGEO graduated project in 02.2010), under a LGPLv2 licence.
- [GeoServer](#) (initial release in 2002, OSGEO graduated project in 03.2013), under a GPLv2+ licence.
- [MapServer](#) (first release as FOSS in 1999, OSGEO graduated project in 12.2008), under a Mapserver licence.¹

Non-functional considerations

1. [deegree](#) and [GeoServer](#) are written in Java, a factor that weights positively in the current selection procedure: given that EuroStat's SDMX tools are available for the Java platform, a compatible software stack (e.g. the Java servlet container) can be used and a less disparate portfolio of software and software administration skills is required. The same applies to programming skills, should Java be adopted as the implementation language for any additional software components.²
2. Similarly, an alternative solution based on [MapServer](#) and [TinyOWS](#) can be deemed more adequate in projects where:
 - a PHP web development framework is adopted;
 - the PHP MapScript component is used/required for the developed of interactive mapping applications.
3. Currently, there is considerably less available documentation (notably 3rd parties tutorials, books, etc.) and service providers for [deegree](#) than for [GeoServer](#) or [MapServer](#). A similar trend is observable by comparing the activity on the [discussion lists](#) of the three products.

Also, there are no publicly available comparative benchmarks of [deegree](#) against the other geodata server products.

¹ The [MapServer](#) licence is an MIT like permissive licence.

² Geoserver incorporates *GeoTools*, an open source (LGPL) Java code library which provides standards compliant methods for the manipulation of geospatial data, that is used in several GIS applications.

4. Performance benchmarks³ of MapServer vs. GeoServer show superior throughput of MapServer when using FastCGI on Linux Operating Systems (MapServer results on Windows machines are erratic – specially when there is an higher number of concurrent requests – and, for some request types, throughput is inferior to GeoServer’s when there are less than 4 concurrent clients).⁴.

As a result of this preliminary evaluation, *deegree* was considered a dominated solution in the candidate set. The evaluation is not based on the product’s intrinsic features and capabilities. It is merely the result of a lower score on QSOS maturity criteria and also a smaller user community and relative scarcity of real-world use cases (the notable exception being the choice of *deegree* to support the [EU INSPIRE geoportal](#)).

Support for OGC W*S interfaces

The basic type of geoweb services to be considered are:

- Web Map Services (see [WMS overview](#)): allow the visual display of spatial data (without necessarily providing access to the features that comprise those data).
- Web Feature Services (see [WFS overview](#)): read-only vector spatial features (i.e. points, lines or polygons, typically with associated alphanumeric information);
- Transactional WFS (WFS-T): editable vector spatial features where editing permissions can be controlled through user authentication and authorisation (the mechanism of authorisation and the supported granularity of control varies in different products).

MapServer does not support WFS-T, as stated in the current MapServer 6.2.1 documentation:

This is just a basic WFS (read-only): transaction requests are not supported and probably never will given the nature of MapServer. GeoServer or TinyOWS is recommended for those needing WFS-T support.

Source: http://mapserver.org/ogc/wfs_server.html

TinyOWS is a WFS server that has recently been incorporated into the MapServer Suite, to address the lack of WFS-T services. Currently, PostGIS is the only spatial database back-end supported by TinyOWS. Such limitation is not relevant from this specific project but diminishes the reusability of the component (e.g. potential users may care to publish other geospatial data stored in common proprietary databases, such as Oracle Spatial and Graph or ESRI ArcSDE spatially-enabled databases).

In *Geoserver*, fine-grained access authorisation (e.g. for edit operations) can be delegated in and controlled by the database system⁵, an option currently under discussion in the MapServer community⁶.

As stated before, *deegree* provides the required W*S interfaces, albeit directed towards a rather more technical user profile than GeoServer, due to the relative paucity of documentation and graphical user interfaces.

Conclusions

The evaluation results are:

- MapServer+TinyOWS or *deegree* are adequate and sufficient solutions for the identified W*S requirements

³ Benchmark data are available at http://wiki.osgeo.org/wiki/FOSS4G_Benchmark.

For products supporting WMTS services (‘GoogleMaps-like’ tiled map services), the following links provide useful information: <http://www.esdm.co.uk/mapserver-and-geoserver-and-tilecache-comparison-serving-ordnance-survey-raster-maps>, <http://www.esdm.co.uk/further-load-testing-of-geoserver-and-mapserver-and-tilecache> and <http://developmentseed.org/blog/2010/oct/19/qa-mapnik-performance-just-important-its-beauty/>.

⁴ It should be noted that, for high-demand image map services (WMS), both products were outperformed by *mapnik*. *Mapnik* is not fully supported in Microsoft Windows operating systems.

⁵ URL: <http://docs.geoserver.org/stable/en/user/data/database/sqlsession.html#using-sql-session-scripts-to-control-authorizations-at-the-database-level>

⁶ URL: <https://github.com/mapserver/tinyows/issues/43>

- GeoServer is equally adequate, better documented (than deegree), easier to configure and publish services (than deegree or MapServer) and supports a larger number of database back-ends (than TinyOWS), namely PostGIS, H2, ArcSDE, DB2, MySQL, Oracle, Microsoft SQL Server and SQL Azure.

The following excerpt provides an adequate overview of the 3 products:

GeoServer, MapServer and Deegree are open-source map server products focusing on Internet mapping applications using OGC webGIS standards. These OGC interoperability standards such as WMS, WFS and WFS-T allow for the cross-platform exchange of geographic information over the Internet. Using these standards, map data stored in Oracle Spatial, PostGIS or ArcSDE databases can be accessed over the Internet with a standard web browser or GIS client software. With WMS, map data can be accessed and displayed as an image that can be overlaid with GIS data from other data sources to produce composite maps. With WFS, users can access the actual geographic features in vector format, while WFS-T allows for creation, deletion and updating of features. MapServer, GeoServer and Deegree are server-based “map engines” to display spatial data (maps, images or vector data depending on the OGC web service) over the Internet to users based on their requests. [...] MapServer has proved to be a very mature and reliable product to distribute maps from GIS data sources over the Internet through the WMS, WCS and other OGC interoperability standards. GeoServer and Deegree are more recent projects built with Java technology. While comparable to MapServer in many ways, GeoServer and Deegree go further by supporting transactional WFS services, allowing users to insert, delete and modify geographical data at the source from remote locations through the Internet.

—Source: [Piep10]

Real-world use

The following table lists a sample of projects using GeoServer.

Organization, system or project	References
Instituto Geográfico Nacional (MCA), Spain	⁹
Gestione Integrata e Interoperativa dei Dati Ambientali (SDI), Italy	¹¹ ¹²
Global Disaster Alert and Coordination System (SDI), UN & EC-JRC	¹⁰
Global Monitoring for Environment and Security, EC-JRC	⁸
Paikkatietoikkuna (SDI), Finland	¹³
UK Location Programme	⁷

Final notes

Note on support for map tile services

Support for WMTS services is not considered a *must-have* requirement for this project: the spatial information can be stored as vector data in the database, and served using WMS or WFS services. It is not foreseeable that a large number of concurrent requests will require a Map Tile server. Consumption of 3rd-party map tile services (e.g. Mapquest, Google Maps, etc.) will be required in the web mapping component, but it is unlikely that a WMTS server will be strictly necessary. If, for performance reasons, a map tile server becomes necessary, two simple options exist:

⁹ URL: http://inspire.jrc.ec.europa.eu/events/conferences/inspire_2012/presentations/143.pdf
¹¹ URL: http://www.minambiente.it/export/sites/default/archivio/allegati/INSPIRE_state_of_play_2011_ITALY.pdf
¹² URL: <http://www.gdmc.nl/zlatanova/Gi4DM2010/gi4dm/Pdf/p116.pdf>
¹⁰ URL: <http://meetingorganizer.copernicus.org/EGU2012/EGU2012-7404.pdf>
⁸ URL: <http://rslab.disi.unitn.it/papers/R72-JSTAR-Brunner.pdf>
¹³ URL: <http://www.oskari.org/trac/wiki/DocumentationBackend>
⁷ URL: <http://data.gov.uk/sites/default/files/Data-Publisher-How-To-Guide-Establish-a-Reference-Implementation-for-an-INSPIRE-View-Service-using-a-GeoServ.pdf>

- Geoserver includes an integrated version of [GeoWebCache](#), a tiling server that runs as a proxy between a map client and map server, caching (storing) tiles as they are requested, eliminating redundant request processing and reducing response time,
- [GeoWebCache](#) (under a LGPL licence) can also be deployed as a standalone product to implement WMS-C, WMTS, TMS or Google Maps KML service interfaces (e.g. to expose some map services to the public through a different server).

Note on support for spatial catalogue services

Support for [CSW](#) services is not considered a *must-have* requirement for this project, although metadata should be stored for the various datasources. If a spatial data catalog is required, the obvious option is to use another Java application, [Geonetwork](#), which was developed for, and is used by, various United Nations offices and programmes ([FAO](#), [OCHA](#), [UNEP](#) and [WFP](#)).

Note on portability

All required GIS components are available for and supported on Linux distributions:

- Currently, the ‘Debian family’ has the most inclusive and up-to-date distribution of FOSS GIS applications and libraries. Repositories maintained by the [DebianGIS](#) project and the [UbuntuGIS](#) project (the latter does the repackaging for the Ubuntu distribution, and also includes some software packages that do not comply with the stricter [Debian Policy](#)).
- For the ‘Red Hat family’, RPM repositories are available via the [ELGIS](#) project and the [EPEL](#) Fedora project (for packages that are not part of the standard Red Hat Enterprise Linux distribution).

Output formats

The following excerpt lists output formats supported by GeoServer in response to W*S requests. As can be noticed, some of the output formats (GeoJSON, GML, KML, SVG) are also supported directly by [PostGIS](#). However, the use of a web service interface (either WMS or WFS) provides an additional level of abstraction that isolates client applications from the database and provides a standardized grammar for obtaining the information (e.g. allowing the use of desktop GIS applications to explore the data, or the development of independent data visualizations tools).

GeoServer output formats

Image Outputs

All image outputs can be initiated from a WMS getMap request on either a raster, vector or coverage data.

Format	Description
KML	KML (Keyhole Markup Language) is an XML-based language schema for expressing geographic data in an Earth browser, such as Google Earth or Google Maps. KML uses a tag-based structure with nested elements and attributes. For GeoServer, KML files are distributed as a KMZ, which is a zipped KML file.
JPEG	WMS output in raster format. The JPEG is a compressed graphic file format, with some loss of quality due to compression. It is best used for photos and not recommended for exact reproduction of data.
GIF	WMS output in raster format. The GIF (Graphics Interchange Format) is a bitmap image format best suited for sharp-edged line art with a limited number of colors. This takes advantage of the format’s lossless compression, which favors flat areas of uniform color with well defined edges (in contrast to JPEG, which favors smooth gradients and softer images). GIF is limited to an 8-bit palette, or 256 colors.

1.4. Commercial-Off-the-Shelf Software Selection Process

TIFF	WMS output in raster format. TIFF (Tagged Image File Format) is a flexible, adaptable format for handling multiple data in a single file. GeoTIFF contains geographic data embedded as tags within the TIFF file.
------	---

Text Outputs

Format	Description
AtomPub	WMS output of spatial data in XML format. The AtomPub (Atom Publishing Protocol) is an application-level protocol for publishing and editing Web Resources using HTTP and XML. Developed as a replacement for the RSS family of standards for content syndication, Atom allows subscription of geo data.
GeoRss	WMS GetMap request output of vector data in XML format. RSS (Rich Site Summary) is an XML format for delivering regularly changing web content. GeoRss is a standard for encoding location as part of a RSS feed. supports Layers Preview produces a RSS 2.0 documents, with GeoRSS Simple geometries using Atom.
GeoJSON	JavaScript Object Notation (JSON) is a lightweight data-interchange format based on the JavaScript programming language. This makes it an ideal interchange format for browser based applications since it can be parsed directly and easily in to javascript. GeoJSON is a plain text output format that add geographic types to JSON.
CSV	WFS GetFeature output in comma-delimited text. CSV (Comma Separated Values) files are text files containing rows of data. Data values in each row are separated by commas. CSV files also contain a comma-separated header row explaining each row's value ordering. GeoServer's CSVs are fully streaming, with no limitation on the amount of data that can be outputted.

Data Outputs

All data outputs are initiated from a WFS GetFeature request on vector data.

Format	Description
GML2/3	GML (Geography Markup Language) is the XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic data sharing. GML2 is the default (Common) output format, while GML3 is available from the "All Formats" menu.
Shapefile	The ESRI Shapefile, or simply a shapefile, is the most commonly used format for exchanging GIS data. GeoServer outputs shapefiles in zip format, with a directory of .cst, .dbf, .prg, .shp, and .shx files.

References

1.4.20 Web Mapping

Requirements

The selected web mapping library is [OpenLayers](#) 2.13+.

If required:

- The web mapping solution can be supported by compatible toolkits, such as [GeoExt](#) 1.1+.
- The web mapping solution can be based on the customisation of existing web map clients, such as [GeoExplorer](#).

Additional [JavaScript](#) libraries can be used if required, subject to the global constraints on COTS selection.

Rationale

The purpose of the web mapping component is to support data visualisation (thematic mapping) and geocoding (finding the geographic coordinates associated within geographical names or administrative units).

The web mapping components must comply with the *Constraints* set on licence compatibility, portability and acquisition cost.

For thematic visualisation purposes, the webmapping component must support the *OGC W*S protocols* previously identified.

For geocoding purposes, the webmapping component must support access to the *GeoNames* service.

Analysis of alternatives

The preliminary candidate set was based on an updated version of the *OSGEO list of webmapping tools*.

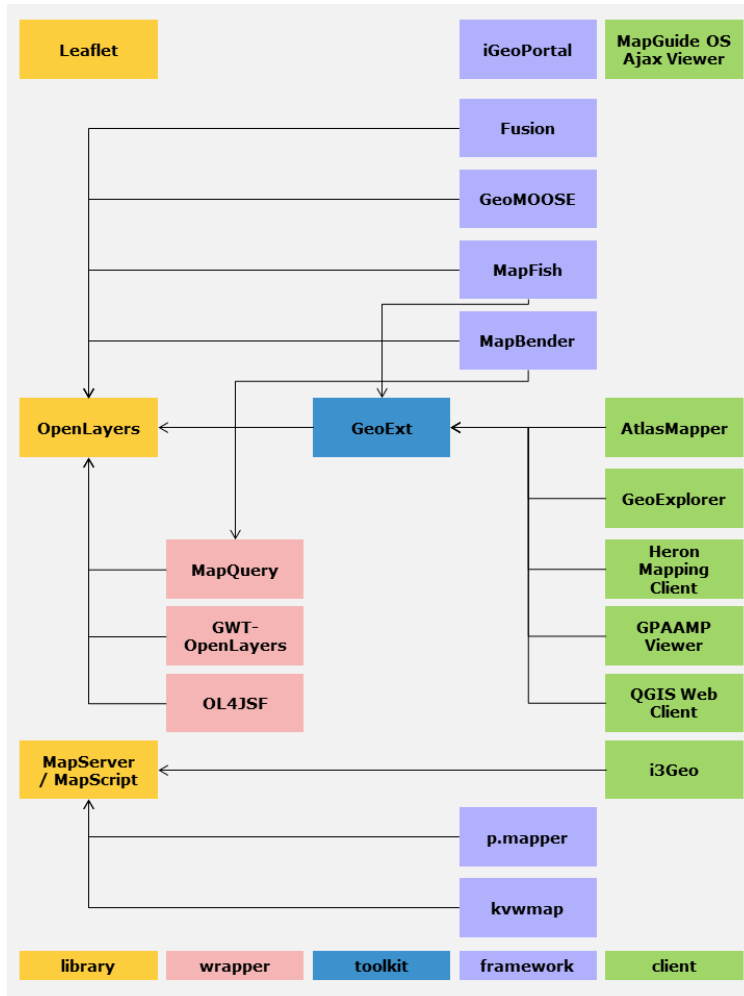
The data was updated (last release, activity, licence type, etc.) to provide the necessary information for the application of the selection *Constraints* on licence type, portability, QSOS maturity criteria (e.g. exclusion of inactive or imature projects) and open standards (e.g. exclusion of *ActionScript* components).

Integration criteria are subsequently applied: e.g. exclusion of components that are essentially plug-ins for specific platforms (e.g. *Plone*) or that depend on components otherwise unrequired (e.g. *PHP MapScript*).

The analysis of alternatives was an update of German Carillo's Web mapping client comparison v.6 [*Carr12*]

The following diagram illustrates the dependencies between the various available alternatives identified in the candidate set.

Dependencies between components



The majority of components are based on the [GeoExt](#) toolkit and the [OpenLayers](#) library which currently the *de facto* standard for web mapping applications.

In the following table a number of components is identified that have EUPL incompatible licences, or development/API language incompatible with the PT Open Standards regulation, or technical documentation not available in English.

Software	Status	Category	OSI License	Source Code Language	API Language	DocLanguage	DocFormats	DocLevel	OSGeo project?
Leaflet	Active	Library	BSD	JavaScript	JavaScript	EN	HTML; Issue Tracker; Twitter	Users; Developers	No
OpenLayers	Active	Library	BSD-style	JavaScript	JavaScript	EN	Blog; HTML; Trac; Wiki	Users; Developers	Yes (Graduated)
uMN MapServer	Active	Library [MapScript]	MIT-style	C/C++	Java; .NET; Perl; PHP; Python; Ruby	EN	HTML; PDF; Trac; Wiki	Users; Developers	Yes (Graduated)
GWT-OpenLayers	Active	Wrapper	Apache License v.2.0	Java; JavaScript	Java	EN	HTML; Javadoc; Trac; Wiki	Users; Developers (incomplete)	No
OL4JSF	Active	Wrapper	Apache License v.2.0	Java	Java; JavaScript	EN	Flash; HTML; Issue Tracker; PDF	Users (incomplete)	No
MapQuery	Active	Wrapper	MIT	JavaScript	JavaScript	EN	HTML; Issue Tracker; Wiki	Users; Developers (incomplete)	No
GeoExt	Active	Toolkit	BSD	JavaScript	JavaScript	EN	Blog; HTML; Trac; Wiki	Users; Developers	No (OSGeo holds the copyright)
ReadyMap Web SDK	Active	Toolkit	GNU GPL v.3	JavaScript	JavaScript	EN	HTML; Issue Tracker; Twitter; Wiki	Users (incomplete)	No
Fusion	Active	Framework	MIT	JavaScript; PHP	JavaScript; PHP	EN	PDF; Trac; Wiki	Users; Developers	No
GeoMapas	Active	Framework	GNU GPL v.3	Java	JavaScript	EN	HTML; Issue tracker; PDF	Users; Developers	Yes (Graduated)
GeoMOOSE	Active	Framework	MIT-style	JavaScript; PHP	JavaScript; PHP	EN	HTML; PDF; Trac; Wiki	Users; Developers	No (Incubation)
iGeoPortal	Active	Framework	GNU LGPL	Java	JavaScript; ASP.NET; JSP	EN	HTML; Issue tracker; PDF; Wiki	Users; Developers	Yes (Graduated) [in deegree]
kvwmap	Active	Framework	GNU GPL v.2	JavaScript; PHP	PHP	DE	HTML; PDF; Wiki	Users; Developers	No
Mapbender	Active	Framework	Simplified BSD License	JavaScript; PHP	PHP	EN	ODP; PDF; Trac; Twitter; Video; Wiki	Users; Developers	Yes (Graduated)
MapFish	Active	Framework	GNU GPL v.3	JavaScript; Python	Java; JavaScript; PHP; Python; Ruby	EN	Blog; HTML; Trac; Twitter; Wiki	Users; Developers	Yes (Graduated)
OpenScale	Active	Framework	GNU LGPL v.3	ActionScript 3	ActionScript 3; JavaScript	EN	HTML; Issue tracker; Twitter; Wiki	Users; Developers	No
p.mapper	Active	Framework	GNU GPL v.2	JavaScript; PHP	JavaScript; PHP	EN	HTML; PDF; Trac; Wiki	Users	No
AtlasMapper	Active	Client	GNU GPL v.3	Java; JavaScript	Java; JavaScript	EN	HTML; Issue Tracker; Wiki	Users (incomplete)	No
GisClient	Active	Client	GNU GPL v.3	JavaScript; PHP	JavaScript; PHP	EN	HTML; PDF; Trac; Video; Wiki	Users	No
Heron Mapping Client	Active	Client	GNU GPL v.3	JavaScript	JavaScript	EN	HTML; Issue Tracker; PDF; Wiki	Users; Developers (incomplete)	No
i3Geo	Active	Client	GNU GPL v.2	JavaScript; PHP	JavaScript; PHP	PT	Blog; HTML; PDF; Trac; Twitter; Video; Wiki	Users; Developers	No
Legato	Active	Client	GNU GPL v.3	JavaScript	JavaScript	EN	HTML; Issue Tracker (Jira); PDF; Video	Users; Developers	No
MapGuide OS Ajax Viewer	Active	Client	GNU LGPL	C++	JavaScript; ASP.NET; JSP; PHP	EN	HTML; PDF; Trac; Video; Wiki	Users; Developers	Yes (Graduated) [in MapGuideOS]
QGIS Web Client	Active	Client	BSD	JavaScript; Python	No published API ?	EN	HTML; Issue Tracker; PDF; TXT; Wiki	Users (incomplete)	Yes (Graduated) [in QGIS]

Todo: Include updated table with the supported protocols.

Examples

The following demos illustrate the use of [OpenLayers](#) for:

- Geocoding or routing purposes
 - [OpenRouteService](#)
 - Using the [GeoNames service](#) (or the ESRI Locator Service);
 - Using the [OpenStreetMap Nominatim service](#)
 - Using the an [OpenLS service](#)
- Visualisation purposes
 - [Map Compare](#) using different base map service providers (Google, Bing, OpenStreetMap, etc.).
 - [Thematic mapping with OpenLayers](#)
 - [Interactive Heatmap](#) or [Heatmap Overlay](#)

References

1.4.21 Web Browser

Requirements

Web User Interfaces (WUI) **must** tested/accepted in:

- [Google Chrome 28+](#)
- or [Mozilla Firefox 22+](#).

Due to the specific requirements of the infrastructure provider for PT, WUI **should** also be tested in:

- [Internet Explorer 8+](#)

Rationale

In this section, the objective is not to select a web browser, but to establish which web browsers should any Web User Interfaces be tested and accepted in.

Two simple criteria are used in the selection process:

- Browser usage statistics in Europe,
- Feature support and performance

Analysis of alternatives

Usage share

Browser usage statistics produced by [StatCounter](#) allow the identification of the products with the larger usage share in Europe. Other sources of statistics on the [usage share of web browsers](#) are available, but the relative ranking of the products is similar.

[Google Chrome](#) is currently the most used browser in Europe (since June 2012). [Internet Explorer](#) relative usage share is declining as is, to a lesser extent, [Mozilla Firefox](#)'s. However there are marked regional differences, for example, in Europe's two most populous countries:

- In Germany, Mozilla Firefox is the dominant browser (> 45%), and Google Chrome has only recently (April 2013) surpassed Internet Explorer;
- In the United Kingdom, Google Chrome is the dominant browser with a share similar to Internet Explorer. Mozilla Firefox and Apple's [Safari](#) have similar usage quotas, around 15%.

Alternative version: [Dynamic charts on web browser usage statistics](#))

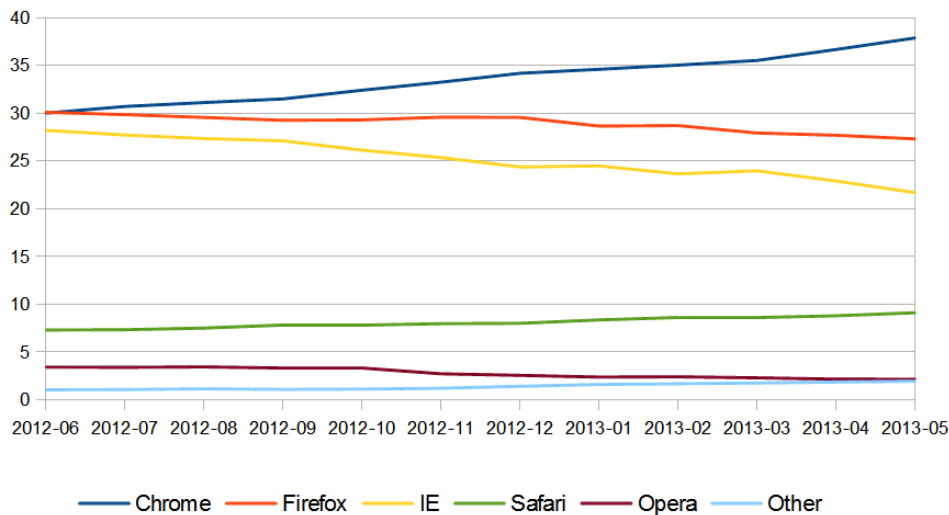


Fig. 1: **Legend:** Top 5 browsers in Europe, June 2012 to May 2013, monthly average

Relative benchmarking

The current versions of the 3 most common browsers ([Chrome](#), [Firefox](#) and [IE](#)) can be ranked using the average of their min-max normalised scores on the following ‘feature support’/‘performance benchmark’ tests:

- [HTML5 test](#)
- [SunSpider](#) (Version 1)
- [Octane](#) (Version 1)
- [Kraken](#) (Version 1.1)
- [Peacekeeper](#)
- [RoboHornet](#) (alpha version)

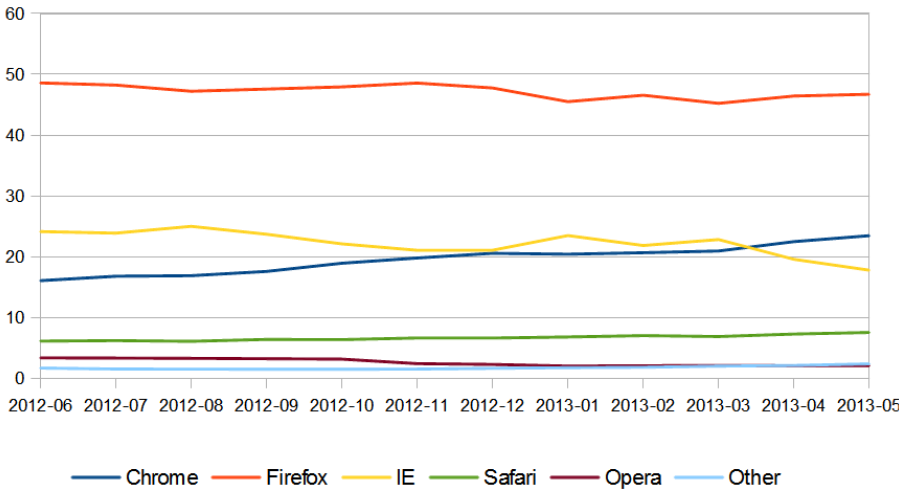


Fig. 2: **Legend:** Top 5 browsers in Germany, June 2012 to May 2013, monthly average

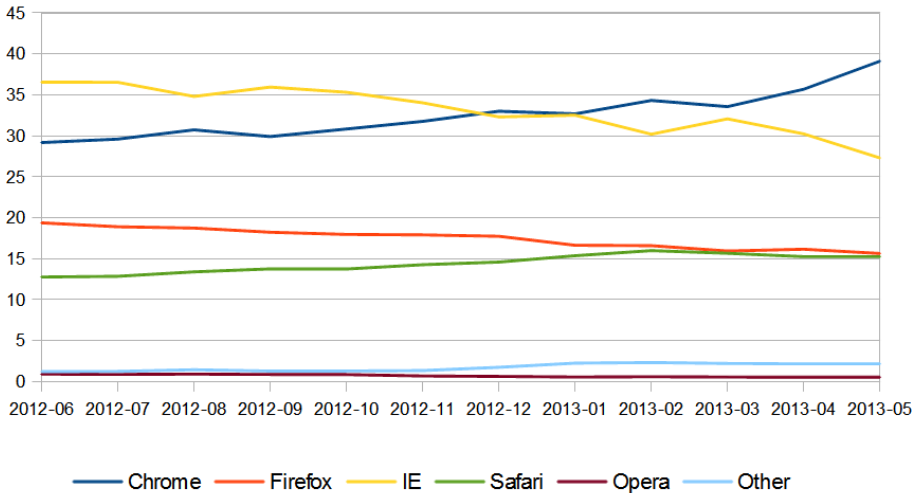


Fig. 3: **Legend:** Top 5 browsers in the United Kingdom, June 2012 to May 2013, monthly average

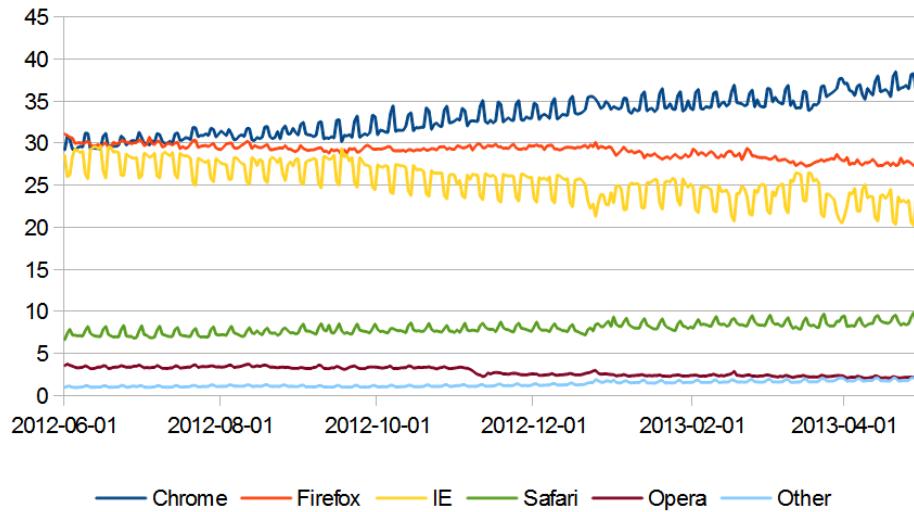


Fig. 4: **Legend:** Top 5 browsers in Europe, June 2012 to May 2013, daily values

The tests focus on partially overlapping aspects of [HTML5](#) and [Javascript](#) support, using different approaches (feature support, laboratory tests directed towards specific features, or performance tests that try to simulate real-world use situations).

The average test score provides a more resilient indicator of each browser's capabilities, regardless of the relative technical merits or applicability of a particular benchmark.¹

Test ↓ Product →	CHROME 28	FIREFOX 23b	IE10
HTML5 Test	476	428	326
Sun Spider (Version 1)	272.5	255.0	205.9
Octane (Version 1)	11189.0	10123.0	4034.0
Kraken (Version 1.1)	2200.2	2330.2	7218.4
Peacekeeper	3574.0	1006.0	1867.0
RoboHornet (alpha version)	122.1	93.2	98.9
Min-Max normalized scores			
HTML5 Test	100.0	68.0	0.0
Sun Spider (Version 1)	0.0	26.3	100.0
Octane (Version 1)	100.0	85.1	0.0
Kraken (Version 1.1)	100.0	97.4	0.0
Peacekeeper	100.0	0.0	33.5
RoboHornet (alpha version)	100.0	0.0	19.8
Relative average score	83.3	46.1	25.6

Conclusions

Usage share statistics and test scores provide a consistent ranking:

1. Google Chrome

¹ Please note that the average relative score reflects the comparison between the 3 products, not among all available browsers. Absolute test results are dependent on hardware and configuration: all tests were made on a Windows 7 (64-bit) machine, running an Intel Core i5-2410M processor, 6GB RAM, and an NVIDIA GeForce 540M graphics card. Firefox 23 beta version was tested instead of the current production version due a (fixed) bug that affected Robohornet's descendant selector test in Firefox 22.

2. Mozilla Firefox
3. Internet Explorer

Based on this ranking, the Google Chrome / [Chromium](#) is considered the adequate browser for WUI development and testing.

An alternative option is to use Mozilla Firefox: if the WUI works correctly in this browser, it will most likely work correctly on Google Chrome.

These two browsers also comply with the defined [Constraints](#) and are supported in the selected [Operating Systems](#).

Searchable miscellanea...

2.1 Notebook

2.1.1 Introduction

Just my chaotic notes about tools...

2.1.2 References

Installing Python on Windows

Note: This page was modified from [The Hitchhiker's Guide to Python](#).

Download the [latest version](#) of Python 2.7 from the official Website. If you want to be sure you are installing a fully up-to-date version then use the “Windows Installer” link from the home page of the [Python.org web site](#).

The Windows version is provided as an MSI package. To install it manually, just double-click the file. The MSI package format allows Windows administrators to automate installation with their standard tools.

By design, Python installs to a directory with the version number embedded, e.g. Python version 2.7 will install at `C:\Python27\`, so that you can have multiple versions of Python on the same system without conflicts. Of course, only one interpreter can be the default application for Python file types. It also does not automatically modify the `PATH` environment variable, so that you always have control over which copy of Python is run.

Typing the full path name for a Python interpreter each time quickly gets tedious, so add the directories for your default Python version to the `PATH`.

Assuming that your Python installation is in `C:\Python27\`, add this to your `PATH`:

```
C:\Python27\;C:\Python27\Scripts\
```

You can do this easily by running the following in powershell:

```
[Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\Python27\;  
↪C:\Python27\Scripts\","User")
```

The second (`Scripts`) directory receives command files when certain packages are installed, so it is a very useful addition.

You do not need to install or configure anything else to use Python. Having said that, I would strongly recommend that you install the tools and libraries described in the next section before you start building Python applications for real-world use. In particular, you should always install `Setuptools`, as it makes it much easier for you to use other third-party Python libraries.

Setuptools + Pip

The most crucial third-party Python software of all is `Setuptools`, which extends the packaging and installation facilities provided by the `distutils` in the standard library. Once you add `Setuptools` to your Python system you can download and install any compliant Python software product with a single command. It also enables you to add this network installation capability to your own Python software with very little work.

To obtain the latest version of `Setuptools` for Windows, run the Python script available here: [ez_setup.py](#)

You'll now have a new command available to you: **easy_install**. It is considered by many to be deprecated, so we will install its replacement: **pip**. `Pip` allows for uninstallation of packages, and is actively maintained, unlike `easy_install`.

To install `pip`, run the Python script available here: [get-pip.py](#)

Virtualenv

After `Setuptools` & `Pip`, the next development tool that you should install is `virtualenv`. Use `pip`

```
> pip install virtualenv
```

The `virtualenv` kit provides the ability to create virtual Python environments that do not interfere with either each other, or the main Python installation. If you install `virtualenv` before you begin coding then you can get into the habit of using it to create completely clean Python environments for each project. This is particularly important for Web development, where each framework and application will have many dependencies.

To set up a new Python environment, change the working directory to wherever you want to store the environment, and run the `virtualenv` utility in your project's directory

```
> virtualenv venv
```

To use an environment, run the `activate.bat` batch file in the `Scripts` subdirectory of that environment. Your command prompt will change to show the active environment. Once you have finished working in the current virtual environment, run the `deactivate.bat` batch file to restore your settings to normal.

Each new environment automatically includes a copy of `pip` in the `Scripts` subdirectory, so that you can setup the third-party libraries and tools that you want to use in that environment. Put your own code within a subdirectory of the environment, however you wish. When you no longer need a particular environment, simply copy your code out of it, and then delete the main directory for the environment.

Eclipse

The Eclipse Platform is a generic foundation for an IDE. That is, the platform is an IDE without any particular programming language in mind. You can create generic projects, edit files in a generic text editor, and share the projects and files with a version control system. The platform is essentially a glorified version of a file-system browser.

[from <http://www.ohloh.net/p/eclipse>]

The current version is Eclipse 4.3 Kepler.

The basic platform is indeed “a glorified version of a file-system browser”. All functionality is provided through plug-ins.

In Eclipse Kepler, the following two plug-ins are already incorporated in the base product (installation is no longer required):

- Mylyn, the task-focused interface for Eclipse;
- Egit, the Eclipse Team provider for the Git version control system.

Numerous other plug-ins are available. In some cases, packages are provided that bundle together a number of useful plug-ins for a specific purpose.

For example, the Eclipse Java EE IDE for Web Developers also includes the Web Tool Platform (that will be required for XSD and XML creation and validation, CSS editing, etc.). It is the selected option.

The StatET <http://www.walware.de/goto/statet>

Other plug-ins may be required, and are described below. . .

Mylyn

Mylyn (<http://www.eclipse.org/mylyn/>).

Install Mylyn Connectors

Mylyn can use a local **task** repository or a remote one.

If the remote task repository is associated with an issue tracking system, a ‘connector’ is required. By default, a Bugzilla connector is included with Mylyn (and Eclipse).

A long list of different connectors is available at <http://wiki.eclipse.org/Mylyn/Extensions>. It includes connectors for Trac and Redmine, GitHub and Bitbucket, etc. . .

Connectors can be installed in different ways:

1. Some (stable) connectors are available through the Mylyn Task List window:
 - Add repository > Install more connectors. . .

For example, the Trac connector is available in this list.

2. Other connectors (alpha versions, etc) can be installed using the standard plug-in install procedure inside Eclipse:
 - Goto Help > Install new software. . .
 - Providing the link to the update site (available in the connector description in the Mylyn/Extensions wiki page).

This is the case for:

- The Bitbucket Mylyn Connector, which is alpha status and is available at <http://www.mylynbitbucketconnector.xpg.com.br/update>
- The GitHub Connector (egit-github), also in alpha status and available at <http://download.eclipse.org/egit/github/updates-nightly>

Specifically for my projects, two connectors are required:

- The Trac connector
- The Bitbucket connector (currently 2013.10.21 not working properly due to an identified but not yet fixed bug)

How to...

A basic tutorial on Mylyn is available at: <http://www.vogella.com/articles/Mylyn/article.html>.

A generic introduction is available at: <http://www.youtube.com/watch?v=bSYVpjom4pU>

Use tags in code comments to generate tasks

Tags in source code comments can be used to generate tasks. The following Window > Preferences can be enabled:

- General > Structured Text Editors > Task Tags
Enable searching for Task Tags
- Java > Compiler > Task Tags
- JavaScript > Validator > Task Tags
- PyDev > Task Tags
- StatET > Task Tags

The Tasks view includes a helpful customization for Java developers. When a Java project is built, the parser automatically scans for Java task tags in your code comments. You can configure the task tag names and their priorities using the Java > Task Tags preferences. Three tags are provided by default (FIXME, TODO, and XXX), and we added a STORY tag to support our agile development process.

Eclipse Distilled, by David Carlson

Workarounds

If the task list disappears...

The workaround is:

1. Goto the drop-down menu in the Task List pane (or right-click to see the contextual menu).
2. Select Restore tasks from history...
3. Select either a zip file (if you've exported the task list before) or an adequate snapshot.

Tasks can be exported from the Task list pane (right-click to see the contextual menu) using the Import and Export... > Export option.

This apparently can occur when updating and restarting Eclipse (see https://bugs.eclipse.org/bugs/show_bug.cgi?id=403467).

Git integration

How to

Import the content of an existing git repository

1. Open the Git Repository Exploring perspective
2. If the repository isn't already listed, then Add a repository (using the drop-down menu)
3. Select the repository in the list, right-click and select Import projects. . .
4. Choose Import as a general project. . . and follow the wizard.

SVN integration

See <http://www.eclipse.org/subversive/>

PlantUML

PlantUML is a component that allows to quickly write :

- sequence diagram,
- use case diagram,
- class diagram,
- activity diagram,
- component diagram,
- state diagram
- object diagram

Diagrams are defined using a simple and intuitive language. The documentation is available here:

```
http://sourceforge.net/projects/plantuml/files/PlantUML%20Language%20Reference
↩%20Guide.pdf/download
```

Images can be generated in PNG or SVG format.

The Eclipse plug-in is described here:

```
http://plantuml.sourceforge.net/eclipse.html
```

Install

It is not clear if the PlantUML plug-in works with the Eclipse Kepler version (4.3).

The update site for Eclipse Juno (4.2) is:

```
http://plantuml.sourceforge.net/updatesitejuno/
```

Let's try it. It's working!

Note that the Graphviz software must be installed.

How to...

1. Goto Window > Show View > Other > PlantUML to open a visualisation tab.
2. Insert the following text into a document (or inside a multiline code comment):

```
@startuml
    user -> (use PlantUML)
    note left of user
        Hello!
    end note
@enduml
```

3. The diagram will be displayed the the PlantUML visualisation pane, where it can be exported to a graphic file.

Papyrus

Papyrus is graphical editing tool for UML2 as defined by OMG.

It can be used as a simple plug-in or as a part of the Eclipse Modelling Tools package. It provides a graphical editor for the Eclipse UML2 project.

UML2 is an EMF-based implementation of the Unified Modeling Language (UML) 2.x OMG metamodel for the Eclipse platform.

The objectives of the UML2 component are: * to provide a usable implementation of the UML metamodel to support the development of modeling tools * a common XMI schema to facilitate interchange of semantic models * test cases as a means of validating the specification * validation rules as a means of defining and enforcing levels of compliance

Although MDT/UML2 provides the metamodel, it does not provide UML modelling tools themselves. One implementation is Papyrus. An older, no longer supported implementation is UML2Tools (<http://wiki.eclipse.org/MDT-UML2Tools>).

[<http://wiki.eclipse.org/MDT-UML2Tools>]

Install

1. Start Eclipse
2. Goto Help > Install New Software
3. Press Add... to add a new resource and specify a name and the URL (the link below is for the Eclipse Kepler version):

```
NAME: Papyrus
URL:  http://download.eclipse.org/modeling/mdt/papyrus/updates/releases/kepler
```

How to

Tutorial on the Eclipse Modelling Framework (not on Papyrus, but it will be useful later on): <http://www.vogella.com/articles/EclipseEMF/article.html>

Python IDE

PyDev is a Python IDE for Eclipse, which may be used in Python, Jython and IronPython development.

Install

Python 2.7 is assumed to be installed.

1. Start Eclipse
2. Goto Help > Install New Software
3. Press Add... to add a new resource and specify a name and the URL:

NAME: PyDevEnv URL: http://pydev.org/updates
--

4. Select PyDev and PyDev Mylyn Integration from the list and press Next.
5. Accept the licence terms when the download ends.
6. Restart Eclipse.

Configure

1. Goto Window > Preferences > PyDev > Editor > Interpreter Python
2. Eclipse can configure the options automatically (press Auto Config) or the location of the Python interpreter can be specified. (in Linux, usually that would be /usr/bin/python)
3. Press OK to finish the configuration.

Start new project

1. Goto File > New > Project and select 'Pydev project'
2. Create a new file (goto File > New > File) HelloWorld.py
3. Add the code
4. Press Run or Ctrl + F11

Python projects will be associated with a "Python perspective", i.e. a customised layout of windows and GUI elements.

References: [1](#) ; [2](#) .

How to...

A generic tutorial covering all the above steps is available at: <http://www.vogella.com/articles/Python/article.html>

ReST Editor

ReST Editor is an Eclipse plug-in providing support to edit reStructuredText files

reStructuredText is a markup language that can be transformed in various output formats with tools like the Sphinx documentation generator, rst2pdf, rst2beamer, ...

More information here : <http://restditor.sourceforge.net/>

Install

This plug-in can be installed through the Eclipse Marketplace (Help > Eclipse Marketplace...) or through the standard plug-in installation:

1. Goto Help > Install new software...
2. Add the project update site: <http://restditor.sourceforge.net/eclipse>
3. Select the ReST Editor plug-in

Configure

The plug-in is configured under Window > Preferences > ReST Editor.

The following options are important:

- The preferred section markers order, that will be used to automatically correct any improper sequence: `#*=^`
- The tab length (3) and the option to insert spaces instead of tabs.
- The spell checking options (see *Hunspell4Eclipse*).

Start a new Sphinx project

The ReST Editor plug-in can be used to create a new Sphinx project:

1. Goto File > New > Project > ReST Editor > Sphinx project
2. Follow the wizard's instructions...

If Sphinx is installed, then ReST documents can be built from within Eclipse (using the make.bat or the makefile).

Hunspell4Eclipse

Hunspell4Eclipse is a plug-in that integrates Hunspell into Eclipse's Spell Checking Service. It is useful if Eclipse is used as for general purpose document editing.

Install

This plug-in can be installed through the Eclipse Marketplace (Help > Eclipse Marketplace...) or through the standard plug-in installation procedure.

Configure

No dictionaries are included. The plug-in uses Hunspell or Myspell dictionaries. These are also used by LibreOffice, and are available in the extensions directory (for example, “C:\Program Files (x86)\LibreOffice 4.0\share\extensions”).

Preferences per workspace can be configured in: 1. Preferences - General - Editors > TextEditors > Spelling

1. Select Hunspell4Eclipse
2. Browse... and select a dictionary(.dic) file

Useful links:

- <https://code.google.com/p/hunspell4eclipse/>
- <https://wiki.mozilla.org/L10n:Dictionaries>

Build a Sphinx project

- Right-click on the make.bat file and goto Run as > Run Configurations...
- Select the option Sphinx (via make file)
- Create a new configuration: * Specify the working directory, for example `${project_loc}/docs` * Specify the type of Sphinx output, for example `html`

The new configuration will be accessible through the button bar (in the Run button drop-down options).

The console pane will show the Sphinx output.

StatET for R

Install

Dependencies (for the stable version StatET 3.3 in Eclipse 4.3):

Java	6 or higher
GNU R	2.13 to 3.0
R package RJ	1.1

For Windows users

The path where R is installed should not contain spaces. For example, install in “C:\ProgramsR” but not in “C:\Programs FilesR”.

Otherwise, strange things are likely to happen, such as packages not getting installed properly. ...

To install the R Packages of RJ 1.1 (StatET 3.0-3.3), use the following command in a common R Term console:

```
install.packages(c("rj", "rj.gd"), repos="http://download.walware.de/rj-1.1")
```

In Eclipse, use the standard plug-in installation procedure:

1. Goto Help > Install New Software
2. Press Add... to add a new resource and specify a name and the URL:

NAME: StatET URL: http://download.walware.de/eclipse-4.3
--

3. For most users it is recommend to select only StatET (and Add-ons/Utilities, if desired), but no Libraries; the dependencies are resolved automatically.

Tutorials

- [Eclipse and StatET – a working environment for R](#)
- [A guide to Eclipse and the R plug-in StatET](#)

Toad Extension for Eclipse 1.9.0 Community Edition

Eclipse: how to...

Change the encoding to UTF-8

- For a specific project: File > Properties > Text file encoding
- Goto Window > Preferences > General > Content types and change the *Default encoding* for each type.
- Goto Window > Preferences > General > Workspaces > Text file encoding

Show a print margin

- Goto Window > Preferences > General > Editors > Text Editors > Show print margin (80)

Using GIST

1. Open the Task List
2. Goto pane menu and select Show Task Repositories View
3. Goto pane menu and select Add Task Repository...
4. Select the appropriate connector, which is GitHub Gists
5. and follow the wizard.

2.2 Using the issue tracking system

2.2.1 Introduction

This section contains information about [Redmine](#), the project management and issue tracking system adopted for the current project.

2.2.2 References

Redmine instalation

Preamble

Presently, the [BitNami Redmine Stack](#) virtual machine (VM) is being used. The following notes thus apply to the deployment of this VM using [Oracle VirtualBox](#), and not specifically to a [Redmine step-by-step installation](#) <<http://www.redmine.org/projects/redmine/wiki/RedmineInstall>>‘_.

The notes below are applicable to a test environment (e.g. in a personal computer) and are not necessarily adequate or recommended for deployment in a production environment.

Note also that the Bitnami Redmine Stack ships with [MySQL](#) as a database back-end.

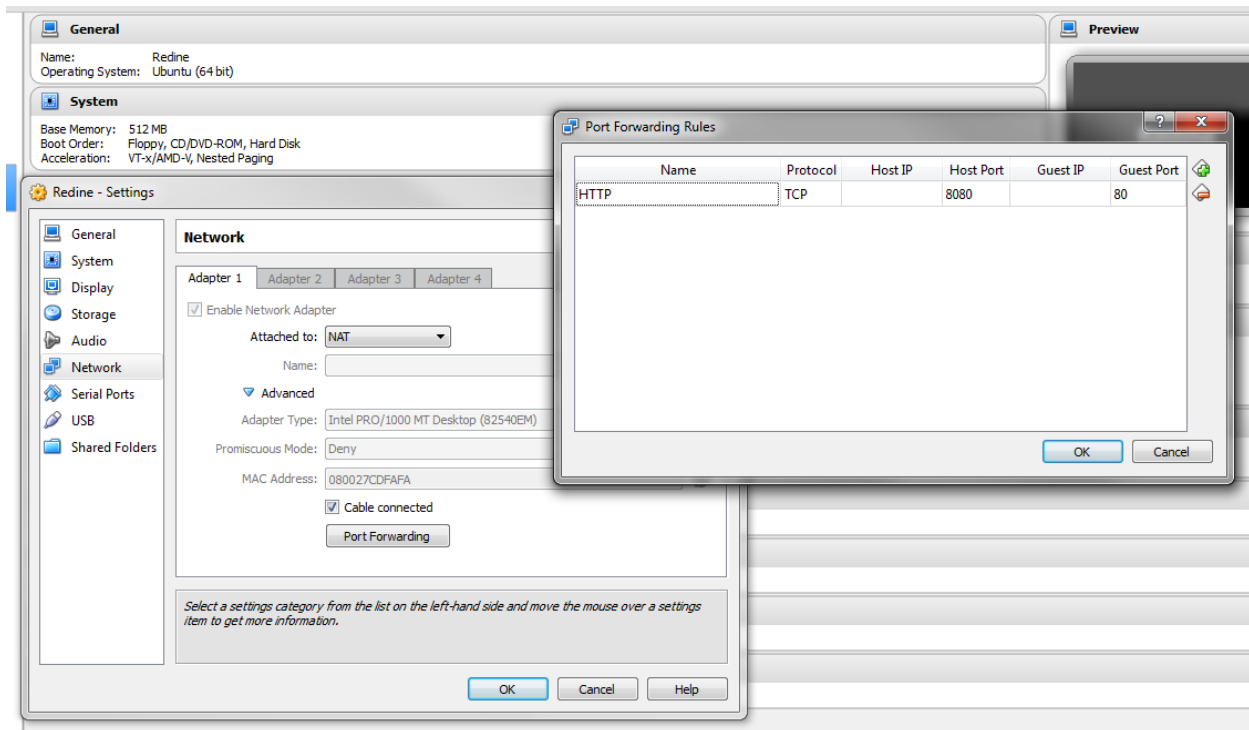
Detailed information can be found at http://wiki.bitnami.com/Applications/BitNami_Redmine.

Deployment

1. Update [Oracle VirtualBox](#) if necessary. Install the most recent [VirtualBox Extension Pack](#).
2. Download the VM from <http://bitnami.com/stack/limesurvey>. Unzip...
3. Create a new VirtualBox VM
 - Guest OS: Ubuntu 64-bit
 - RAM: at least 512 MB RAM
 - Storage: select the VMDK file
 - Network: Enable NAT

After everything is configured, the network adapter will be changed to Bridged Adaptor (had some problems with the Bridged Adaptor over WiFi, hence this choice).

NAT can also be used if the Guest is to be accessed only from the Host. Port forwarding was setup as depicted below:



1. Start the VM. Prepare to install the Virtual Box Guest Additions (Devices → Install Guest Additions)
2. The default Linux login is bitnami/bitnami. In real life, the default users and passwords should be changed.
3. Enable root:

```
$ sudo passwd root
```

In real life, the `root` password should **not** be enabled. (Instead of entering as `root`, use `sudo` instead).

4. Change keyboard layout if required:

```
$ sudo apt-get update
$ sudo apt-get install console-data
$ sudo dpkg-reconfigure keyboard-configuration
$ sudo dpkg-reconfigure console-setup
```

If later reconfiguration is required:

```
$ sudo dpkg-reconfigure console-data
```

5. Install the VirtualBox Guest Additions:

```
$ sudo mount /dev/cdrom /mnt
$ cd /mnt
$ sudo ./VBoxLinuxAdditions.run
$ sudo reboot
```

If the build fails, check the log file:

```
$ nano /var/log/vboxadd-install.log
```

(Note that there is always be a fail message for windows system if no graphic interface is installed in the server):

```
$ sudo apt-get update
$ sudo apt-get install dkms                #if required
$ sudo apt-get install build-essential    #if required
$ sudo apt-get install linux-headers-generic # and/or
$ sudo apt-get install linux-headers-3.2.0-53-virtual # for example, if such is the
↪version required...
```

6. How to access the BitNami Virtual Appliance?

If the network as been set to NAT and port forwarding has been configured as specified above, then the application can be accessed at <http://localhost:8080/redmine> .

The default application login information is user/bitnami

Further information is available here: http://bitnami.com/faq/virtual_machines

Backups

Redmine backups should include:

- data (stored in your redmine database)
- attachments (stored in the files directory of your Redmine install)

Here is a simple shell script that can be used for daily backups (assuming you're using a mysql database):

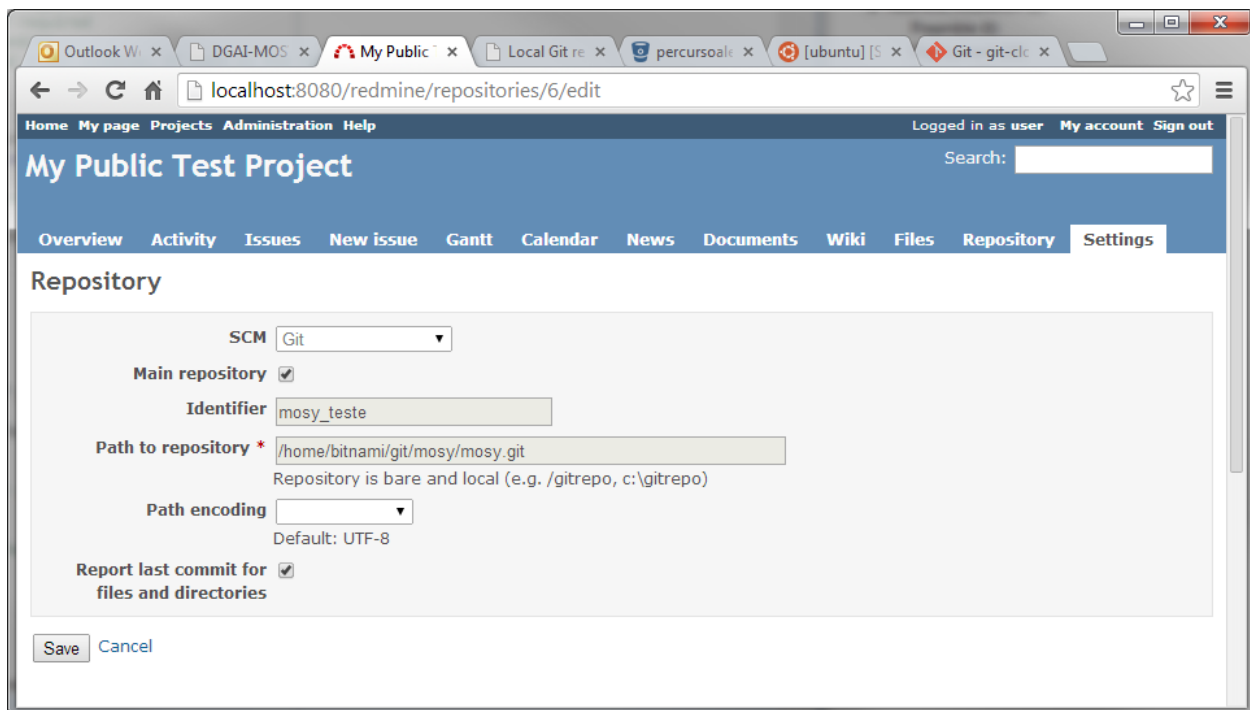
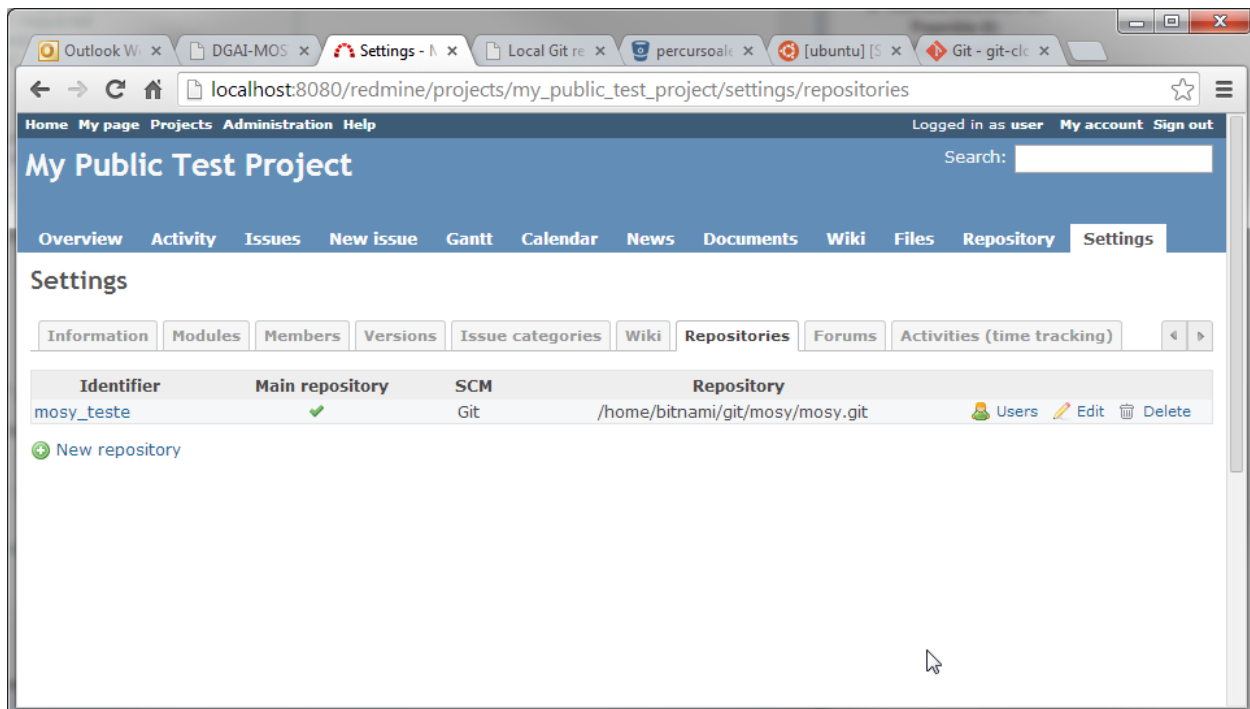
```
# Database
/usr/bin/mysqldump -u <username> -p<password> <redmine_database> | gzip > /path/to/
↪backup/db/redmine_`date +%y_%m_%d`.gz

# Attachments
rsync -a /path/to/redmine/files /path/to/backup/files
```

Repositories

The Bitnami Redmine Stack already includes Git.

Access to a local git repository can be configured by the project's manager at the project settings tab.



Note that the repository should be local and bare. For example, to clone from a remote repository, use:

```
git clone --bare git://yourgitserver.org/project.git
```

and add full path to the repository in Projects > Settings > Repositories e.g. /var/repositories/project.git.

Permissions on the repository folder for the redmine user:group should be are 775.

To update the bare repository with changes from its remote working origin do:

```
git fetch -q origin master:master
```

or push the changes into the bare repository:

```
git push --all <url-of-bare-repo>
```

Note: See

<http://www.redmine.org/boards/2/topics/34226?r=35533>
[what-is-a-bare-git-repository/](http://www.redmine.org/boards/2/topics/34226?r=35533)

<http://www.saintsjd.com/2011/01/>

Bibliography

- [Berg05] van den Berg, Karin (2005). Finding Open options. An Open Source software evaluation model with a case study on Course Management Systems. Unpublished Master's thesis, Tilburg University, The Netherlands.
- [EIFv2.0] EIF European Interoperability Framework for pan-European eGovernment Services, Version 2.0. URL: http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf
- [East99] Eastman, J. R. (1999). Multi-criteria evaluation and GIS. *Geographical information systems*. 1 :493-502.
- [HOSG09] Hauge, O.; Osterlie, T.; Sorensen, C. F.; Gereia, M. (2009). An empirical study on selection of Open Source Software-Preliminary results. In *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS'09. ICSE Workshop on* (pp. 42-47). IEEE. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5071359&tag=1
- [INSPIRE] Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). URL: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32007L0002:EN:NOT>
- [MoRE07] Mohamed, A.; Ruhe, G.; Eberlein, A. (2007). COTS selection: past, present, and future. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the* (pp. 103-114). IEEE. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4148924
- [RNID12] Regulamento Nacional de Interoperabilidade Digital URL: <http://dre.pt/pdf1sdip/2012/11/21600/0646006465.pdf>
- [SWLH11] Scott, J.; Wheeler, D.A.; Lucas, M.; Herz, J. C. (2011). Open Technology Development (OTD): Lessons Learned & Best Practices for Military Software. Sponsored by the Assistant Secretary of Defense (Networks & Information Integration) (NII) / DoD Chief Information Officer (CIO) and the Under Secretary of Defense for Acquisition, Technology, and Logistics (AT&L). Version 1.0. URL: <http://dodcio.defense.gov/Portals/0/Documents/FOSS/OTD-lessons-learned-military-signed.pdf>
- [TrCo10] Triaille & Coppens (2010) - JRC Open Source Software Guidelines. Joint Research Center of the European Commission URL: <https://joinup.ec.europa.eu/sites/default/files/OSS-guidelines-v-DIGIT-2.pdf>
- [Atos13] Atos (2013) - Qualification and Selection of Open Source software (QSOS). Version 2.0. 2013-01-19. Appendix A: QSOS Maturity Criteria. URL: http://backend.qsos.org/download/qsos-2.0_en.pdf
- [Gard13] Gardler, Ross (2013) - Software Sustainability Maturity Model (SSMM). OSS Watch. Joint Information Systems Committee. URL: <http://www.oss-watch.ac.uk/resources/ssmm>

- [Foge09] Fogel, Karl (2009) - Producing Open Source Software: How to Run a Successful Free Software Project. URL: <http://producingoss.com/>
- [EUPLv1.1] European Union Public Licence URL: <https://joinup.ec.europa.eu/software/page/eupl/licence-eupl>
- [Conw11] Conway (2011) - PL/R - The Fast Path to Advanced Analytics. URL: <http://bunsen.credativ.com/~jco/2011/plr-PostgresOpen-2011.pdf> <http://bunsen.credativ.com/~jco/2011/plr-PGEast-2011.pdf>
- [Inspire] Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE) URL: <http://inspire.jrc.ec.europa.eu>
- [ISO.IEC_13249-3:2011] "Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial" URL: http://webstore.iec.ch/preview/info_isoiec13249-3%7Bed4.0%7Den.pdf
- [ISO.IEC_9075:2008] "Information technology – Database languages – SQL"
- [ISO_19125-1:2004] "Geographic information – Simple feature access – Part 1: Common architecture." URL: http://portal.opengeospatial.org/files/?artifact_id=25355
- [ISO_19125-2:2004] "Geographic information – Simple feature access – Part 2: SQL option" URL: http://portal.opengeospatial.org/files/?artifact_id=25354
- [ISO_19136:2007] "Geographic information – Geography Markup Language (GML)" URL: <http://www.opengeospatial.org/standards/gml>
- [CMMI10] CMMI Product Team (2010). Capability Maturity Model® Integration for Development, Version 1.3, Improving processes for developing better products and services. no. CMU/SEI-2010-TR-033. Software Engineering Institute. URL: <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>
- [Weic13] Weichand, Jorgen (2013). Entwicklung und Anwendung von Downloaddiensten im Kontext der europäischen Geodateninfrastruktur INSPIRE.
- [Kolo04] Kolodziej, Kris (ed.) (2004). OpenGIS® Web Map Server Cookbook. OGC Document Number: 03-050r1 Version: 1.0.2 URL: http://portal.opengeospatial.org/files/?artifact_id=7769
- [Piep10] Pieper, Gertrude (2010). Existing Open-Source Tools - Possibilities for Cadastral Systems. in Steudler, D; Torhonen, M-P and Pieper, G. (2010) (ed.) - FLOSS in Cadastre and Land Registration. Opportunities and Risks. y Food and Agriculture Organization of the United Nations (FAO) and the International Federation of Surveyors (FIG). :24-32
- [Carr12] Carillo, G. (2012). Web mapping client comparison v.6. 03 January 2012. <http://geotux.tuxfamily.org/index.php/en/geo-blogs/item/291-comparacion-clientes-web-v6>

B

baseline, [53](#)
branch, [53](#)

C

change, [53](#)
changeset, [53](#)
checkout, [53](#)
clone, [53](#)
commit, [54](#)
conflict, [54](#)

E

environment variable
 PATH, [131](#)

F

fork, [54](#)

H

head, [54](#)

M

merge, [54](#)

P

PATH, [131](#)

R

repository, [54](#)
resolve, [54](#)
revision, [54](#)

S

sync, [54](#)

T

trunk, [54](#)

U

update, [54](#)

V

version, [54](#)

W

working copy, [55](#)