CVSAnalY

CVSAnalY

A tool to analyze software repositories.
`CVSAnalY` Version 2.0.0
April 2009

Carlos Garcia Campos

This file documents the `CVSAnalY` tool, which extracts information out of source code repository logs and stores it into a database.

This is Edition 2.0.0, last updated 16 April 2009, of *The CVSAnalY Manual*, for `CVSAnalY` version 2.0.0.

Published by LibreSoft

Short Contents

Table of Contents

1 Overview of CVSAnalY

The CVSAnalY tool extracts information out of source code repository logs and stores it into a database.

2 How to install CVSAnalY

CVSAnaly uses the standard Python Distutils. First of all you should install all the dependencies.

- Python MySQLDB: there should be packages for your operating system, so you should be able to install it as any other software. In Debian systems the package is python-mysqldb

```
# apt-get install python-mysqldb
```

- Repository Handler: it's, like CVSAnalY, part of the LibreSoft tools set[1]. You will probably have to install it from sources. Here is an example assuming Repository Hanlder 0.2 is used and the tarball has been already downloaded.

```
$ tar xvjf repositoryhandler-0.2.tar.bz2
$ cd repositoryhandler-0.2
$ ./configure
$ make
# make install
```

- Python SQLiteDB: it's optional, required only to use SQLite instead MySQL as database system. It's also usually available in most of the operating systems.
- Other dependencies (CVS, SVN and Git) are actually optional, although required depending on the type of repository you want to analyze. It's recommended to install of them.

We are now ready to install CVSAnalY.

```
$ tar xvzf cvsanaly2-2.0.0.tar.gz
$ cd cvsanaly2-2.0.0
# python setup.py install
```

You can also use CVSAnalY without installing it, just by running the cvsanaly command from the directory sources.

```
$ cd cvsanaly2-2.0.0
$ ./cvsanaly2 --help
```

[1] https://forge.morfeo-project.org/projects/libresoft-tools/

3 Running CVSAnalY

Once CVSAnalY is installed you can use it just by running the executable cvsanaly2[1]

The syntax to run cvsanaly2 is the following:

```
cvsanaly2 [options] [URI]
```

Analyze the given URI. An URI can be a checked out local path directory, or a remote URL pointing to a repository. If URI is omitted, the current working directory will be used as a checked out directory. The type of the repository will be automatically detected, so the only information you have to provide about the repository is this URI. CVSAnalY doesn't run checkouts, so if the repository doesn't support remote retrieving of the log, a checked out directory must be provided. The repository log will be parsed and stored in a database. CVSAnalY doesn't expect to have all privileges on the database server, so the database should be created before running CVSAnalY or it will fail. This is not relevant if you are using SQLite since there isn't any server.

Global options:

- -h, –help

  Show help information

- -V, –version

  Show the version number of CVSAnalY

- -g, –debug

  Enable debug mode. It shows useful information for debugging like the commands that are being executed, the SQL statements, parsing status and so on.

- -q, –quiet

  Run silently, only important error messages is printed.

- –profile

  Enable profiling mode. It shows information about how long some tasks take to run.

- -f, –config-file

  Use a custom configuration file. See Chapter 4 [The configuration file], page 7

- -l, –repo-logfile

  Use the given log file as the input of the log parser instead of running the log command for the repository.

- -s, –save-logfile

  Save the input log information to the given path.

- -n, –no-parse

  Skip the parsing process. It only makes sense in conjunction with –extensions

- –extensions

  Run the given extensions after the log parsing/storing process. It expects a comma-separated list with the name of the extensions to run. Dependencies among extensions are automatically resolved by CVSAnalY.

[1] It's called cvsanaly2 to avoid conflicts with old (incompatible) cvsanaly 1.x)

Database specific options:

- –db-driver

  Use the given database system. MySQL (actually `mysql`) is the default (and recommended) option.

- -u, –db-user

  The name of the user to connect to the database system. The given user should exist, since `CVSAnalY` will not try to create any user. This option doesn't make sense if you are using SQLite. The default option is `operator`.

- -p, –db-password

  The user's password to connect to the database system. If it's not provided, you will be prompted to enter it.

- -d, –db-database

  The name of the database. I should exist, since `CVSAnalY` will not try to create any database. If you are using SQLite, this option might be a local path instead of just a name. Default option is `cvsanaly`.

- -H, –db-hostname

  The host name where database system is running. This option doesn't make sense when using SQLite. Default option is `localhost`.

Examples:

- Running `CVSAnalY` with a CVS repository already checked out using MySQL driver.

  ```
  $ cvsanaly2 -u carlos -d gstreamer ~/src/cvs/gstreamer
  ```

- Running `CVSAnalY` with a SVN repository using the remote URI and SQLite as the database driver

  ```
  $ cvsanaly2 --db-driver sqlite -d ~/db/nautilus.db \
  http://svn.gnome.org/svn/nautilus
  ```

4 The CVSAnalY configuration file

Running CVSAnalY might require to provide a lot of command line options. Some of such options such the hostname, database driver, database user name and so on, depend on the system where CVSAnalY is running and not on the repository to be analyzed. Those options have to be always provided, making the CVSAnalY execution command too large in some cases. A configuration file can be used to avoid this situation. Before parsing the command line option provided by the user CVSAnalY reads its configuration file, taking the options found there to replace the default values. There might be two configuration files:

- System-wide configuration file: /etc/cvsanaly2
- User configuration file: ~/.cvsanaly2/config

The system-wide configuration file is read first, then the user configuration file is read overriding the options already provided by the system-wide file, and finally the command line options are parsed overriding any other option already provided. For the options not provided by a configuration file or the command line interface, the default values will be taken.

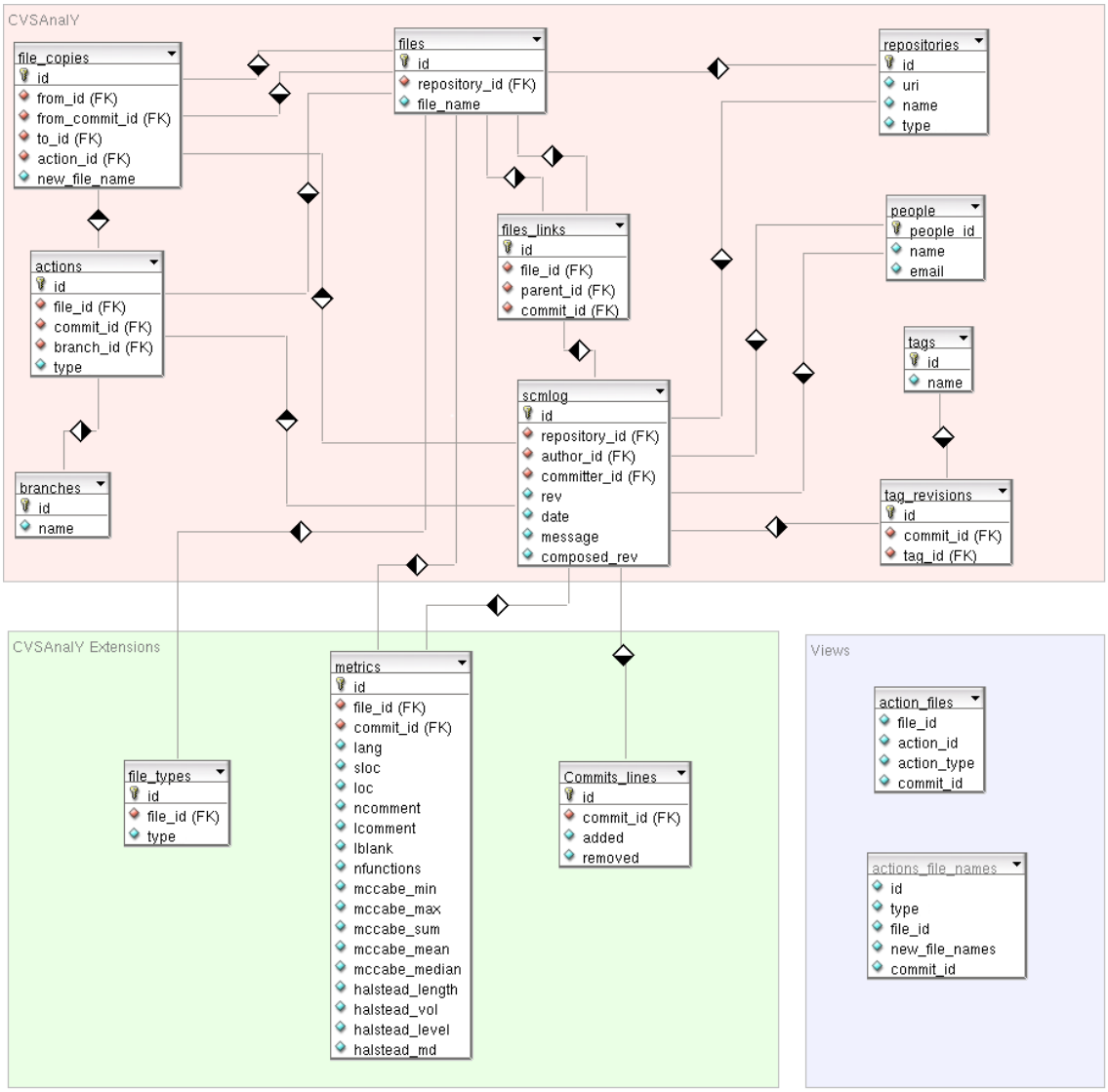
The configuration file is just a file containing key-value pairs.

```
# Run in debug mode
debug = True

# Run quiet
quiet = True

# Enable profiling
profile = True

# Database driver
db_driver = 'mysql'

# Database user
db_user = 'cvsanalyuser'

# Database user password
db_password = 'mysqlpassword'

# Database hostname
db_hostname = 'localhost'

# Run always Metrics and CommitsLOC extensions
extensions = ['Metrics', 'CommitsLOC']
```

5 The Database design

5.1 Database schema overview

The database is divided into two main parts. The first one consists on the set of tables that represents the history of the project based on the information from the repository log. These tables are filled by `CVSAnalY` during the parsing process exclusively with the information provided by the repository log. This is the main goal of `CVSAnalY` and, therefore, these tables will always be present in the schema independently of how `CVSAnalY` was executed or even what project has been analyzed. The second part is composed by tables created and filled by `CVSAnalY` extensions (See Section 5.3 [Extensions schema], page 13). The information provided by these tables depends on every `CVSAnalY` Extension, however the main goal is common: to complete the `CVSAnalY` schema with additional information directly related to the existing tables but not provided by the repository log.



5.1.1 General conventions

The database has been designed according to the following rules:

- Internal identifier: every table has an internal identifier called `id`. It's an auto-incremental integer and it's always the primary key of the table.
- Foreign keys: all fields with the suffix `_id` are foreign keys so that it's easy to identify them quickly.
- Character encoding: `CVSAnalY` uses always utf-8 so all the fields in the database that contain strings are utf-8 encoded.

5.2 `CVSAnalY` database schema details

The scmlog table

The main table is scmlog. Every commit in the repository is represented by a record in the scmlog table.

- id: Identifier in the database.
- rev: It's the revision identifier in the repository. It's always unique in every repository.
- committer_id: Committer identifier, that is, the identifier in the database of the person who did the commit.
- author_id: Author identifier. Some source control management systems, differentiate the person who did the commit from the person who actually made the changes. When not supported by the repository, this field will be `NULL`.
- date: The date when the commit was done.
- message: The commit message.
- composed_rev: It's a boolean to indicate whether the rev field is composed or not. This is needed because the rev field must be unique in every repository which is not possible in CVS since it uses revision numbers per file. The combination of a file path and its revision is what make a commit unique in a system like CVS. For this particular case the rev field is represented by the concatenation of the revision number, the pipe character ('`|`') and the file path. Here is an example for a CVS repository:

  ```
  1.1.2.1|/poppler/glib/demo/render.c
  ```

- repository_id: Identifier in the database of the repository where the commit was done.

The actions table

This table describes the different actions performed in every commit. In systems like CVS, where the commit is limited to a single file, there will be only one record in the actions table for every commit. However, most of the version control systems support atomic commits, where several actions are carried out on several files[1].

- id: Identifier in the database.
- type: It's a character representing the type of the action. Currently supported actions are:
  - `'A'`: The file has been added to the repository.

[1] With `file` we actually refer to both file and directory

  - `'M'`: The file has been modified. It's the most common action.
  - `'D'`: The files has been deleted.
  - `'V'`: The file has been renamed. This might be a simple rename or that the file has been moved to another path keeping its name or not. In any case, the file identifier will never change after a `'V'` action.
  - `'C'`: The file has been copied. It's similar to an `'A'` action, since a new file is added to a repository, but in this case it was copied from another existing file.
  - `'R'`: The file has been replaced. This means that an existing file is removed, and a new one with the same name is added. In addition, another existing file might be used as a base for the replacement, that is, the new file is a copy of such an existing file.

  Not all of the action types are always supported, for example, for CVS repositories only `'A'`, `'M'` and `'D'` actions are supported.
- commit_id: It's the identifier of the commit where the action was performed. It's a foreign key that references the `id` field of `scmlog` table.
- branch_id: It's the identifier of the branch where the action was performed. It's a foreign key that references the `id` field of `branches` table.

The files table

The `files` table contains an identifier for every file or directory found in the repository.

- id: Identifier in the database.
- file_name: The name of the file or directory. Note that this is not a path.
- repository_id: It's the identifier of the repository to which the file belongs. It allows to easily get the list of files of the repository. It's a foreign key that references the `id` field of the `repositories` table.

The file_links table

This table contains relationships between files. The relationship between two files is always parent - child.

- id: Identifier in the database.
- parent_id: the identifier of parent file or -1 if the file is in the root of the repository. It's a foreign key that references the `id` field of the `files` table.
- file_id: the identifier of the file. It's a foreign key that references the `id` field of the `files` table.
- commit_id: the identifier of the commit where the relationship appears for the first time. When a file or directory is moved, a new link is created with the id of the commit where the move action as performed. There will be, therefore, two links for the same file each one with a different parent. In order to know which parent is the right one at a certain point (revision), the commit_id field is used to choose the link that contains the latest commit. It's a foreign key that references the `id` field of the `scmlog` table.

The file_copies table

The `file_copies` table is used to store additional information about actions that involve more than one file. Copies, moves, renames and replacements are actions performed over two or more files. The `file_id` field of the `actions` table refers always to the file that is the object of the action.

- id: Identifier in the database.
- to_id: identifier of the file that is the destination of the action. In a move or copy operation, this field is the same than the `file_id` in the `actions` table. However, when a file is replaced, the `file_id` stored in the `actions` table is the existing file being replaced, and this field contains the `file_id` of the new file that replaces the existing one. It's a foreign key that references the `id` field of the `files` table.
- from_id: identifier of the file that is the source of the action. In a move or copy operation this is the `file_id` of the file from which the move or copy is done. It's a foreign key that references the `id` field of the `files` table.
- from_commit_id: identifier of the commit from which the operation is carried out. The source file contents are taken from the revision associated to this commit identifier. It's a foreign key that references the `id` field of the `scmlog` table
- new_file_name: contains the new name of the file for rename actions or `NULL` for other actions.
- action_id: the identifier of the action. It's a foreign key that references the `id` field of the `actions` table.

The branches table

This table contains the list of branches found in the repository

- id: Identifier in the database.
- name: The name of the branch

The tags and tag_revisions tables

The combination of these two tables represents the list of tags found in the repository. The `tags` table contains the names of the tags while the `tag_revisions` tables contains the list of revisions pointing to every tag

- id: Identifier in the database.
- name: The name of the tag
- tag_id: the identifier of the tag associated to this revision. It's a foreign key that references the `id` field of the `tags` table.
- commit_id: the identifier of the commit representing the revision. It's a foreign key that references the `id` field of the `scmlog` table.

The people table

This table contains the name and email (when available) of the people involved in the repository.

- id: Identifier in the database.

- name: the person's name or nick. Depending on the repository type this field contains the real name (or at least the name provided by the user) or the user name for repositories that have authentication like CVS or SVN.
- email: The email of the person or `NULL` if it's not provided by the repository.

The action_files and actions_file_names views

The database design tries to represent all the logic behind the output given by a repository log. Because of this, the complexity of the schema makes difficult to write queries. In order to help the users of the database, `CVSAnalY` provides these two views.

- action_files: the `file_id` field of the `actions` table might be confusing. Depending on the situation you might want the `file_id` of the `actions` table or the `to_id` field of the `file_copies` table. For example, if you are interested on the new files added, you need the `to_id` field, while if you want to know what files have been deleted, you need the `file_id` field, since a replace operation implies that the replaced file is not available anymore. This view is useful when you are in the first case. The view is just a "clone" of the `actions` table, but using the `to_id` as `file_id` for replace actions.
- actions_file_names: since the name of a file may change during the history, we usually need to get the new file name given to a file as a result of a rename action. The new file name is stored in the `file_copies` tables which means we always need to add an extra join in the queries. This view is also a "clone" of the `actions` table including also the `new_file_name` field of the `file_copies` tables.

5.3 `CVSAnalY` extensions schema details

A `CVSAnalY` Extension adds one or more tables with additional information directly related to the existing tables, but not provided by the repository log.

FileTypes extension

This extension adds the `file_types` table containing the file type associated of every file found in the repository. The file type is not the mime type of the file but one of the following categories:

- code: source code files (C/C++, Python, Java, etc.)
- build: files used to build and configure the source code (Makefile, configure, cvsignore, etc.)
- ui: files containing graphical user interface definitions (glade, gtkbuilder, ui files, etc.)
- i18n: translation files (.po, .mo, etc.)
- documentation: documentation files
- devel-doc: documentation for developers (HACKING, ChangeLog, etc.)
- package: package files (.tar.gz, .deb, .rmp, etc.)
- image: icons and files (.png, .jpeg, etc.)
- multimedia: audio and video files (.ogg, .avi, .mp3, etc.)
- unknown: files with an unknown type, generally files that don't have extension

The file type is based on the file extension and it's assumed that a file doesn't change its type during the history.

The `file_types` table contains the following fields:

- id: Identifier in the database.
- file_id: the file identifier. It's a foreign key that references the `id` field of the `files` table.
- type: the name of the type (as described above in this section)

Metrics extension

This extension provides simple source code metrics for every revision of every single file found in the repository. Since this extension is about source code, it uses the FileTypes extension to get only source code files.

- id: Identifier in the database.
- file_id: the identifier of the file. It's a foreign key that references the `id` field of the `files` table.
- commit_id: the identifier of the commit (revision). It's a foreign key that references the `id` field of the `scmlog` table
- lang: the programming language (as given by the sloccount tool)
- sloc: number of source code lines of code
- loc: number of lines of code
- ncomment: number of comments
- lcomment: number of commented lines
- lblank: number of blank lines
- nfunctions: number of functions
- mccabe_*: all fields starting with mccabe correspond to McCabe cyclomatic complexity
- halstead_*: all fields starting with halstead correspond to Halstead software science metrics

CommitsLOC extension

This extension adds a table with the number of lines added and removed for every commit.

- id: Identifier in the database.
- commit_id: the commit identifier. It's a foreign key that references the `id` field of the `scmlog` table
- added: number of lines added in the given commit
- removed: number of lines removed in the given commit

6 Frequently Asked Questions

6.1 Database

6.1.1 Why aren't branches associated to commits instead of actions?

While it's logical to think that a commit is always associated to a single branch, that's not true in SVN repositories. The fact that branches don't really exist in SVN (they are just paths in the repository), makes possible to find commits involving files from different branches for the same revision. It happens, indeed, more often than expected. So, in most of the cases, all actions referencing the same commit will reference the same branch too, but we need to keep the relationship between action and branch in order to support all other cases.

6.1.2 Why doesn't files table contain full paths for files?

`CVSAnalY` stores the whole history of the project in the database. Paths do change quite often during the history of a project as a result of a rename or a move operation. We are interested in files independently of their paths, but we also need to be able to get the full path of a file at any point in the history. Assigning identifiers to the files instead of the paths we can follow the history of any given file even if it's renamed or moved. Additionally, relationships between files are stored in the file_links table. When, for example, a directory is moved to another path, we only need to create a new relationship between two existing files.

6.1.3 Why are there two tables files and file_links instead of a single files table with a pointer to its parent file?

That was the first approach we followed. Since we are trying to represent a tree, sounds reasonable to use a single table where every record is a node of the tree containing a pointer to its parent node. This approach works indeed, but makes quite hard to build paths, since it requires multiple recursive queries for every file path. We use instead a graph schema, where there's a table containing the vertices (files table) and another table containing edges (file_links table). A tree is indeed a graph without cycles. With this approach it's possible to get the adjacency matrix for any given revision with only two queries. Building paths for the files once we have the adjacency matrix is trivial.

6.1.4 Why are there two tables for tags?

Despite it tires to represent the same concept, tags are implemented in a different way in every source code management system. Theoretically, a tag is just a label that points to a snapshot of the repository. In CVS is not possible to take a snapshot of the repository with only one revision, since revisions are per file. In CVS a tag is actually a list of pairs file-revision which is represented by `CVSAnalY` with the `tags` and `tag_revisions` tables. For SVN and Git repositories there will be only on revision for every tag[1]

[1] In SVN doesn't exist tags, they are usually implemented by copying the whole source tree into another directory. In this case the revision stored in the `tag_revisions` table points to the copy operation

6.1.5 Why is CommitsLOC and extension if that information is provided by the cvs log command?

Because it's only available in the log output of CVS repositories. For the other repositories we have to get. In the case of SVN getting such information might take a long time depending on the number of revisions. Since the lines added/removed per commit is not the most important information provided by CVSAnalY and it makes the parsing process quite longer, we decided to move it to an extension, so that it will be optionally executed.

Appendix A GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and

that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called

an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.