# Standard for packaged Simusol Documents; CSSAN Repository and other advances

**Diego Saravia**[1,2] **, Dolores Alía de Saravia**[1]

[1]Facultad de Ciencias Exactas – Universidad Nacional de Salta
Buenos Aires 177 – 4400 Salta, Argentina

[2]Inenco (Instituto U.N.Sa – Conicet)
Buenos Aires 177 – 4400 Salta, Argentina

loli@unsa.edu.ar, dsa@unsa.edu.ar, diego.saravia@gmail.com

***Abstract.*** *We present a package standard for Simusol which allows to contain in a single file, called a "document", diagrams, tables, charts, modules, results, and other elements, making it easier to edit, simulate, install, transport and store these same elements and so enabling collaborative work. Additionally, we developed a reference software to implement it, a repository of packages together with its software; exchange control repositories (with Subversion software); a graphical interface for them, and some changes to Simusol in order for it to be adapted to these new standard.*

Simusol it's been used by various people to work in many different projects. We think it is convenient to create concepts and tools that allow all this work to be used by others in a collaborative framework.

## 1. Background, Issues

In INENCO we have been developing and using Simusol since 2002[Alía et al., 2002], making continuous improvements to date. At this point it is necessary to make its use easier, providing a graphical interface, essential in Windows and useful in GNU/Linux, automating the installation of extensions and the creation and distribution of them so they can be used by third parties, and allowing the emergence of a community of solution developers for Simusol.

Simusol uses a set of files:

- **Diagrams (.dia)** Are XML files. All Simusol diagrams will have the .dia extension. They are displayed in the Dia program window (canvas), as a set of interconnected objects. Objects can be of the following types: boxes, branches, nodes (single and multiple, particularly the reference node), elements, models, and others (which simusol ignores, such as texts or other symbols without an established use). Objects of the element type represent a capacitor, a voltage source, a current source, a resistor or an inductor, and their "description in a sheet" must end respectively in C, E, J, L, R, uppercase or lowercase depending on whether the shape has two connection points or just one. Model type objects represent more complex objects than an element and its description in a sheet must end in T.
  Some Dia objects are defined by Dia itself; others, the most interesting to developers of Simusol, are defined by the shapes that are listed in sheet files. Day and Simusol programs need access to these files to be able to run. In addition to the forms associated with each object, element, and diagram can be associated with any of them.
  There are several types of diagrams:
  - **Main (projects) and auxiliary.** For each simulation, processing or run of Simusol (compile + link + execution), there is a main diagram, usually referred to as "project". There can be auxiliary diagrams, summoned from the main one, containing useful indications for Simusol; that is determined by the contents of the diagram.
  - **Elements.** These diagrams contain definitions of element or node type objects. There may be element or node type objects with no diagram but that is rare, and each diagram may contain many object definitions. Each element type object is associated with a shape file, the description of which is contained in one or more sheets.
  - **Models.** These diagrams give the definition of a single model type object. Each model object has a diagram and a shape file; its description is contained in one or more sheets.
- **Shapes and Sheets.** XML files with .shape and .sheet extensions respectively, essential for the proper functioning of Dia, which are identified by an internal name that does not necessarily correlate with the file name. Dia doesn't accept repeated internal names for

Supported by Consejo de Investigación, U.N.Sa

shapes or sheets. It notifies about this problem with respect to shapes.
Each shape is associated with a file containing an icon for its graphic display in the Dia menu. The sheets are a list mentioning shapes or objects that appears as a sub-menu in Dia. The sheets act as sub-menus offering shapes. Dia recognizes all shapes it finds, but it offers to incorporate to a diagram, through a menu, only those shapes it has found in a sheet. While using Dia, its menu offers, at all times, two sets of objects: the first one is always the same, the second is associated with a sheet. The user can change what sheet is being offered without affecting the diagram in preparation. There are no limitations for the objects or shapes which can appear in a sheet. It's just a matter of practicality related to the diagram being prepared, so that the need for changes (and its frequency) of sheets is reduced.
The sheets and shapes used by Dia are the ones available at the time of loading. If a user installs new sheets or shapes (elements or models), he must reload Dia. When using Dia, icons representing objects or shapes are displayed in the menu, which are provided by the chosen sheet.
Dia has a directory where the user can save his definitions of sheets, and a directory from which he can save his shape and icon files. It also has system directories which can be accessed by all users. Shapes included in a sheet are displayed on the menu with an icon that can have descriptions made by the sheet in one or more languages. The description, in the user language of Dia, is displayed on mouse-over the menu icon.
**Sheets for Simusol.** Some of the sheets and shapes used by Dia are relevant for Simusol. Their internal name is of the form Simusol–Something. Some current examples of "something": thermal, electrical, generic, models, hydraulic (in review). A sheet with the name Simusol–foo can refer to any object or shape, but shapes that we can call sheet's own are Something–foo.
The box object, or "UML - Object" of Dia, is widely used and therefore should be present in all Simusol sheets. Simusol also uses branches to connect shapes; branches are always available for Dia.
Simusol's own shapes with the name Something–foo are used to represent simple class shapes: node, capacitor, voltage source, current source, resistor or inductor, or more complex shapes we call models.
**Shapes for Simusol.** Simusol requires that Simusol's own shapes meet certain conditions:

- **Identifying Text:** All must admit one, except reference nodes.
- **Connection points:** Reference node representatives have a connection point. Representatives of other nodes must have at least one, and may have a "main" one; other simple shapes may have one or two connection points, but no "main" one; models may have any amount of connection points, but no "main" one.
  Simple shapes with a single connection point have always an implicit connection to a reference node.
- **Descriptions in the sheets that offer them:** Shapes for Simusol should have them. Except when they serve to represent nodes. Those descriptions have a termination at the end, after a semicolon; the same termination for all Dia used languages.
- **Termination:** If used to represent an element: a capacitor, voltage source, current source, resistor or inductor, they must end with, respectively: C, E, J, L, R, uppercase or lowercase depending on whether the shape has two connection points or only one.
- **Functions.** Written in FORTRAN, c, or other languages, they must be able to be linked (linkeadas) with Sceptre by the gcc linker.
- **Data.** Of various kinds, usually in the form of tables, which Simusol translates for use with Sceptre, or data for comparing results, etc.

The concept of package that is described in this paper has been created and implemented for organizing and facilitating the use of these files, as well as for sharing them and installing them in the right places.

## 2. Methodology and tools

We have used: Perl[Wall, 2012], as main programming language and Wxwidgets[Ollivier, 2012] to develop the GUI; Make[FSF, 2010] to develop the installer and the package manager; Subversion[Apache Software Foundation, 2010], Perl and Apache[Apache Software Foundation, 2012] to develop the package repository. CMSimple[CMSimple.org, 2012], to create the website, a system that creates a single "html" file on the site and that does not require databases. This file is backed with a version control system. Simusol uses as usual: Dia, Sceptre, gnuplot, gfortran.

All developments are made in version controlled repository here described. One for each project. We had a problem with the repository we used before and therefore started from the latest preexisting version to create the first version of these repositories.

## 3. Results

In order to meet the main objectives foreseen, several specific actions were carried out:

1) definition of a package standard for sharing work with Simusol.
2) creating a software (Simpack) that allows to create, install, upload and download packages from the repository.
3) changes to Simusol in order to work with new standard.
4) creating a package-centered GUI that runs on different operating systems.
5) creating a repository website that enables package management "https://cssan.simusol.org.ar" and the software that makes it work (SimpackSrv). This website offers:
   - user registration and a space to their packages, validate and publish them.
   - subversion repositories for participants to work on their packages,
   - a mailing list for developers,
   - verification that no package uses words reserved for its parts, remembering which words were reserved by each previously uploaded package.
   - the possibility that a participant endorses the work of another.

Simusol documents (s.tgz files or packages) are source type packages. In the future we will create, using summapack[Saravia, 2012b], tools for creating deb, rpm, ebuilds, and other types of packages from source packages.

### 3.1. The concept of a "Simusol document" and its standard

A "Simusol document" or package is a container for the various files used and generated when working with it. It allows for easy use, reuse, storage, transportation and installation of models, elements or projects that can be used or visualized by third parties.

The standard is a set of rules that Simusol packages must comply. A package is a directory tree packaged with tar and compressed with gzip. within a single file with a ".s.tgz". extension; it can be thought as a "Simusol document" and, therefore, it is used with various interchangeable denominations: package, document, s.tgz file.

The user interface works conceptually with these documents, abstracting the user from their internal structure: opening them, modifying them and saving them. Each Simusol document can contain one or more main diagrams, or none when containing only text documents, models, elements or other information. The interface can process all these diagrams at once or one at a time. Similarly, it acts with contained LaTeX documents. A processed Simusol document will allow displaying the results without reprocessing, (even when Simusol or Sceptre are not installed), using graphic files of stored results.

Simusol documents have a name ir order to be able to identify them, and a version to mark their evolution over time. Their name is of the form: package-name_package-version[mod—ele—pro—other].s.tgz

The version of the shape (example: 24.12), consists of two numbers separated by a point; the second number will be less than 1,000,000. Also, there exist in the repository name-last.s.tgz packages with the version "last", meaning the last.

A package can depend on others to be processed by Simusol. For example, a package containing a model used by another package. The package using it requires the first package to be installed in order to be processed. That is why the idea of installing a Simusol document in a system is a useful one, as it allows other documents to use the models on which it depends, or to display the results (even when Simusol itslef is not fully installed).

Each package:

- Has a directory structure to store any kind of content linked to Simusol and Sceptre: projects: contains main diagrams. For each processable diagram there will be a subdirectory with its name containing several files produced by its processing, in subdirectories named: tests, results, tmps. Diagrams can be in the root directory of the package, but it is not advised; data: any kind of table-like files readable by Simusol and added to Sceptre if applicable or used for other purposes; shapes: icons and their shapes; soft: any kind of program, script, etc. not included in the above, each program in a subdirectory with its name, in a src subdirectory with its source; sheets: sheets that refer to their shapes and perhaps to others contained in the packages on which it depends; elements: diagrams of the included element; shapes; models: diagrams of the included shapes; docs: documentation, their sources and at least 3 versions. txt. pdf and. html, each document in a subdirectory (named), sources in the src subdirectory. They will also be placed in a subdirectory man the .man, in another info the .info, and in a pod the .pod; functions: functions in C, Fortran or another language that produces objects linkable with scepter objects (gcc, gfortran), each in a subdirectory (with its name); objects: objects of Dia; graphics: graphics of interest, for example, .png files of the diagrams, or others used in LaTeX or HTML documents that one wants to put in there; auxs: for diagrams or auxiliary .tex files, ie reconvened from a main, Pm: for perl modules; other: other material.

Not all packages have all directories with content.

- Contains in its root the following files: INSTALL: indicating any special instructions for installaing the package; README, indicating, among other features of the package, its concepts, what is contained in it and how is it used; COPYING * licenses, the repository only accepts GPL, AGPL and FDL, each according to the content type. The authors retain the rights, a copyright transfer is not requested; AUTHORS: list of authors; Makefile for installation and process; the generic file contained in "skeleton" should work (not recommended to use another) NEWS: news; THANKS : people who collaborated; TODO: tasks to do; nombre_del_paqute. summa: package description.
- The nombre_del_paquete.summa file is a file that contains package data in an form automatically processable by bash with the variables: DEPENDS: dependency list; IDEPENDS: dependants list. (Do not write it. On uploading summa the repo puts it automatically); BASEVERSION: same pair of numbers in the package name; SIMUSOLTYPE: other, mod, ele, project; DESCRIPTION: package description; SUMMARY: summary description; KEYWORDS : keywords separated by spaces, the first should be the package name; AUTHOR: "name ¡mail¿, name ¡mail¿"; MAINTAINER: also an author; VENDOR: institution as the author; PRIMARYSITE: https://cssan.simusol.org.ar; SRC_URI: = $PRIMARYSITE/pkgs/name; SRC_PKG_TYPE: = simusol; CATEGORY:= [drying, water heater, etc.]; SUMMAVERSION: summa file version in the same package version, since each package version changes the BASEVERSION variable.

To create the package it is recommended to do it from the empty structure of package "skeleton". With simpack-k command a skeleton directory is created with this structure. This directory should be renamed with the name of the package one wishes to create and work on adding content.

"Ele", "mod" and "pro" types will be used only for packages containing material relating respectively to an element, a model or project (at least one main diagram, without models or elements usable by another package). Any package that is not exclusively of these types or that has mixed types will be of the "other" type. We suggest splitting the packages, one for each type. If a function is used by many, it's best to put it into a package and designate it as a dependency of those who use it.

When Simusol is called by the Makefile and therefore by the graphical interface, it uses a directory with the name of the diagram as output directory, placing the output files in the directories: results, tests and tmps.

One or more models may be contained in a package, as well as a main diagram. To use the main diagram the package that contains it must be installed (or created) and also the packages or models on which it depends. Everytime someone installs a model for use with Simusol, all these files must be located in a suitable place for all programs that require them: Dia, Simusol, and Sceptre.

If the installer (using Simpack) has access to the "root" or "administrator" account, and he so desires it, the files can be installed in the system so that all users can access them, or only for a single user. In case they are installed for the entire system it can be done as a system package (.deb or rpm or tgz) in the/usr directory or in /usr/local as a "local" program. These files can also be installed in the same directory as the diagram that works with the diagram that will convene them, or, if the diagram is contained in an "unpacked" package, in the expected directories. The programs that we use will seek a particular file by choosing the first one they find in the following order: 1) current directory 2) package type structure in the current directory 3) directory where the main diagram is 4) Specific Directories of the "unpacked" package containing the main diagram, 5) directories where the user installs packages ( /.dia, /.simusol, /share, /bin, etc.) 6) system directories (/usr/local/ share / day, / usr / local / share / simusol, / usr / local / lib / simusol) 7) system directories for system packages (/usr/share/dia/usr/share/simusol,/usr/lib/simusol). The "share" ones are used for computer architecture-independent files (scripts) and the "lib" ones for libraries that are dependant (executables). If in some project or in other file a path from the root is specified, one should look there, if not, in the locations already listed. This search system can be structured as an internal or environmental variable, type PATH: SIMUSOLPATH.

The GUI has an internal workspace where it will keep packages open and unpacked and where low-level Makefile commands run (executed by the interface) and also the lower-level commands called by Makefile or by the user directly from the console. A user can not reprocess an "installed" project on the system, but all users can see the results, if any; furthermore, the installed projects can be copied, and other elements too, for use and/or reprocessing in other's own projects.

### 3.2. Simpack

It is the software that handles Simusol packages. This software packs and unpacks the files, it uploads and downloads them to and from the repository, it installs them in the system and realizes other required functions. Given the order of commands, it is summoned as:

```
simpack ORDER PACKAGE
```

where the ORDER may be: c: test and make the package SHARE/simusol/pkg, from now PKG; p: copy the package into the execution directory; is: open, test and install the indicated package (according to a root or a common user) from the execution or PKG directory, without dependencies, a: open the package into the execution directory and test it, ni: open and test the package (not installed); nc: test the directory (not created); i: downloads and installs the package and its dependencies (no test); dni: downloads the package and its dependencies but does not install it (no test); ib: install the package and its dependencies without downloading them, previously downloaded; u: upload packages to the server; k: creates package skeleton in execution directory; r: cleans cache with open package tree (PKG) DISTRO: rpm, deb, ebuild, slack: packets generated in those formats; SVN: import, co: creates a repo in the simusol site and installs the local version with svn; all others as up and ci; h: help.

Testing the package involves verifying that it meets the standard. The software also verifies that no other packages use the same names, nor contain files with the same name as other packages in the repository. These names: of package, shapes, sheets, models, elements, etc. are reserved for the first package that contains them. A name already used can not be uploaded.

### 3.3. The CSSAN package repository

The repository allows sharing Simusol documents and, in particular, those which contain models or elements. Documents can be easily uploaded and downloaded from the repository.

The same scheme is used as that of other free software communities in the world, creating a space for sharing contributions, a storage or repository of useful material to work with Simusol. Hence the name CSSAN "SimuSol Comprehensive Archive Network".

It is important to ensure the quality of the packages and generate information for the participants; that is why a peer review system is implemented to evaluate and certify the packages.

In the CSSAN repository each subscribed author can upload his packages. All packages can be downloaded to be uses. Registration as an author is free. There is an example package called "skeleton", empty, to facilitate participants to incorporate their contributions.

### 3.4. SimpackSRV

Its the CSSAN site software. It is called from the web interface as follows:

spserv ORDEN USER PASSWD MAIL PROYECT CODE NUEVAPASSWD

Any irrelevant data is replaced with "-". The "define" type commands store an action internally, which, on being quoted by their code, produce the predefined change. It is called mainly from apache by "cgi".

ORDER:-dnc: defines new user, spserv -dnc USER - MAIL PROYECT CODE NUEVAPASSWD;-dnm: defines new mail, spserv -dnm USER PASSWD MAIL;-drc: defines password recovery, spserv -drc USER - - - - NUEVAPASSWD,-cc: changes password, spserv -cc USER PASSWD - - - NUEVAPASSWD;-np: adds a new project,spserv -np USER PASSWD - PROYECT;-code: executes the action defined by that code, spserv -code - - - - CODE; rp: reviews package uploaded by the user and public, spserv -rp USER PASSWD - - - - - PAQUETE;-av: endorses published package, spserv -av USER PASSWD - - - - - PAQUETE;-pp: tests words, spserv -pp - - - - - - PALABRAS;-au: authentic, spserv -au USER PASSWD;-basedir: informs basedir, spserv-basedir;-avales: lists endorsements, spserv-avales;-paqpub: lists published packages, spserv-paqpub;-paqusr: lists packages uploaded by the author, spserv -paqusr USER;-inicio: runs as root from the makefile all necessary actions to install the system, spserv-inicio;-reuser: runs as root by hand, recreates users, spserv-reuser.

### 3.5. Subversion Repositories for participants

CSSAN provides infrastructure for software development, consisting of an unlimited number of subversion repositories for participants.

These repositories, by default, allow anyone to look at the project and to modify it. There is a command to add more users to each repository and for making it visible only for them. More complex configurations (some watch and others write) are also possible. Sharing the view of all projects with all is encouraged, except when they are used to store personal information, which is not recommended, since it is not a safe place, and is not designed to be so. Actions that can be performed in the repository system are: 1. Create user, 2. Create Project, 3. Import Project 4 Creat working copy, 5. Make changes to project 6 Update working copy, 7. Update repository, 8. Access the repository. The actions that are commonly used are the updating of the working copy from the repository, incorporating to the working machine the changes made by other people or other machines, and updating the repository for "uploading" the improvements made by some user.

### 3.6. Email lists, website and microsites of Simusol and CSSAN and other internet resources

Two mailing lists have been created, one for users[Saravia, 2011h] and one for

developers[Saravia, 2011d]; also a Facebook group (Facebook, 2011).

For the teaching courses and to help in self-training, a series of videos from the last lectured graduate course were uploaded to a Youtube channel. They are not edited yet[Saravia, 2011a].

An online course was created in the Moodle site of the Faculty of Exact Sciences of UNSa[Saravia, 2011c].

The site has been migrated to a system called CMSimple, which allows to save it in a single html file without using databases.

### 3.7. User Interface

The usual way of working with Simusol involves using the command line of the programs in perl that interact with the data files and the executable Sceptre and gnuplot.

A Makefile was created, which besides installing all the necessary programs to work with Simusol, both in GNU/Linux and in Cygwin (Windows), allows control of previous executables and packages at a higher level of abstraction than previously existing programs in Simusol. The Makefile saves the user some complexity, avoiding the need to use the command line directly.

This Makefile is also used by the GUI at a still higher level of abstraction. In order to use the Makefile gnu-make should be installed. Once autoinstalled, the Makefile can run with the command line under the name simusol-make. To start a project, one must create a directory with the name one wants to give to the project, and run simusol-make from that directory. This command can be activated with different commands and variables. For example, the project variable indicating which diagram will be processes. Like this:

simusol-make proyect=battery sample

will show existing results for the main diagram “battery.dia”. Some other commands are: clean, to clear results; [simusol — sceptre — run] to process the diagram with simusol; with simusol and sceptre, and, finally, with simusol, scepter and graphic postprocessing. If a stage was already fulfilled it is not done again; installpack to install the package; repo to put the package into the local repository, upload to upload to the site, help to get help, etc...

### 3.8. Installing Simusol with Makefile

The simplest way to install Simusol from the sources is performed using the Simusol projects and packages master Makefile. Download it, place it in any directory (where there is no other Makefile), and run: a) “make install” installation for a user, or b) “sudo make install” installation as supervisor for all users. Ideally, the user has configured sudo for use without a password. The Makefile is installed in the/usr/bin directory or possibly in the /bin(ver/usr/local) as simusol-make. So once it is downloaded into a download directory, running “make install”, depending on whether the user is root or not, will install it on the site. From this point it can be run and it will act based on what directory it is standing on.

Installation in Ubuntu, as it is installed by default, usually takes an hour with an internet connection via ADSL of medium quality, (down to about 30Kb/s). It downloads 60 MB in about 90 packages that take 160 MB of disk space. To these packages must be added Perl modules borrowed from CPAN and the Simusol software. If the installation fails, one must retry.

The projects, unless the author changes it, usually have the same Makefile (or earlier versions) in their directory. In that sense it is a self-installing file, which installs and coordinates the use of Simusol. If you receive a Simusol package and unpack it with tar/gzip, have Internet access and the server is running, running “make install” will also install Simusol.

In some systems, if you are the first user installing Simusol, it will be required to install it as a supervisor, as the Makefile installs software that does not know how to operate without being installed in areas of the computer that the user cannot change. If the installation is successful, an appropiate icon for running the GUI will appear.

Tested on Ubuntu 11.10, Ututo 2012, Debian 6 and Suse 11.4; with Suse 11.3 and earlier versions one must play a little with the Makefile; within it the necesary code, with comments, can be found.

When running “sudo make install” for the first time on the Makefile for installing, there are times when “Wx” installation fails. It is usually solved by running the installation again.

### 3.9. Changes to Simusol

Simusol was adapted to find the files in the locations provided in the packages and for organizing results according to the standard in three directories: results, tests, tmps; options were fitted to the command line.

### 3.10. Windows Installer

Because Windows does not have a centralized package system like most GNU/Linux distributions, it is necessary to create a software that helps the user to install the various programs on which simusol depends for its operation: Dia, emacs, cygwin, gnuplot, etc.

For this purpose the LiberaWin.exe program was created. All firewalls or virus scanners must be disabled before running this program on

Windows as in some cases they interfere with the installation process. Liberawin is downloaded from the Simusol site. The last step of LiberaWin installation, once Cygwin is installed, is to run in it the same Makefile described above. Whereupon Simusol in Windows, using Cygwin, works exactly like in GNU/Linux and such as has been described here.

### 3.11. Graphical user interface

The XUL-based GUI that was presented in a previous work was abandoned, because changes in the new versions of Firefox canceled some features of XUL that we used, and it no longer worked. We have adopted the WxWidgets project for the purpose of having an interface that can be used on various operating systems.

The interface integrates the concept of packages and guides the user to these. So the usual way of working is to open a package (s.tgz file) and modify it, saving it when desired

Normally the interface has multiple open documents (packages) and provides a window to change the current working package.

Each file can have multiple directly processable diagrams and others providing additional information such as auxiliaries, models and elements.

From the interface a window of the diagram editor can be opened for all of these, as well as calculate the simulation under the processables.

In the early versions of the simusol-gui a package or document is called a project (can have many main diagrams); in the following versions, as in this work, a main diagram is called a project.

## 4. Conclusions

Project Simusol was provided with powerful tools and standards to allow collaborative work. Usual tools in all open source projects: mailing lists, version control repositories, a standard for its packages, along with tools for its use. The user interface, self-installing with its command line, and the GUI that manages the new packages, along with the windows installer, make it easier to use and share work.

We invite all those who developed works with Simusol to publish them in the CSSAN repository and to all users to register in the mailing lists, in order to share concerns and progress.

## References


This, Subversion repository of this article sources. https://www.simusol.org.ar/users/dsa/repos/trabajos/cssan.

Alía, D., Saravia, L., and Saravia, D. (2002). Avances introducidos en la capacidad del simulador del sistemas solares térmicos, Simuterm (Simusol). http://www.cricyt.edu.ar/lahv/asades/modulos/averma/trabajos/2002/2002-t008-a006.pdf.

Alía de Saravia, D., Saravia, L., and Saravia, D. (2009). Simusol website. http://www.simusol.org.ar.

Alía de Saravia, D., Saravia, L. R., and Saravia, D. (2012). Simusol: Simulating thermal systems using Sceptre and Dia. *Journal of Free Software and Free Knowledge*, 1(1):30–34. http://www.icfoss.org/ojs/index.php/foss/article/view/14 Also in JAIIO Congress, 2003.

Apache Software Foundation, T. (2010). Apache Subversion. http://subversion.apache.org/.

Apache Software Foundation, T. (2012). Apache HTTP Server. http://projects.apache.org/projects/http_server.html.

CMSimple.org (2012). Sistema de creación de sitios: CMSimple. http://www.cmsimple.org/.

Dia (2012). Some extra shapes for diagrams . http://dia-inxstaller.de/doc/en/custom-shapes-chapter.html#N41016.

FSF (2010). GnuMake. http://www.gnu.org/software/make/.

Murdock, I. (2007). How package management changed everything. http://ianmurdock.com/solaris/how-package-management-changed-everything/.

Nullsoft (2009). Nullsoft Scriptable Install System (NSIS). http://nsis.sourceforge.net/Main_Page.

Ollivier, K. (2012). WxWidgets, Cross Plataform GUI Library. http://www.wxwidgets.org/.

Saravia, D. (2010a). Instalación de Software Libre en Windows: Psicro, Simusol y Sceptre. Interfaz gráfica para Simusol y Sceptre basada en XUL. http://www.cricyt.edu.ar/lahv/asades/modulos/averma/trabajos/2010/2010-t008-a012.pdf.

Saravia, D. (2010b). Sitio de SumaPack. http://www.sumapack.org.

Saravia, D. (2011a). Canal Youtube para Simusol. http://www.youtube.com/user/simusol2011.

Saravia, D. (2011b). Comprehensive SimuSol Archive Network. CSSAN Repositorio de paquetes para SimuSol. https://cssan.simusol.org.ar.

Saravia, D. (2011c). Curso Virtual de Simusol en Moodle.
http://moodlexa.unsa.edu.ar/course/view.php?id=16.

Saravia, D. (2011d). Developers mail list.
http://groups.google.com/group/simusol-develop?hl=es e-mail address: mailto:simusol-develop@googlegroups.com.

Saravia, D. (2011e). Grupo Simusol en Facebook.
https://www.facebook.com/groups/237446849651219/.

Saravia, D. (2011f). Simpack.
https://simusol.org.ar/downloads/simpack-ultimo.tgz.

Saravia, D. (2011g). Simusol-make.
http://simusol.org.ar/downloads/Makefile.

Saravia, D. (2011h). Users mail list.
http://groups.google.com/group/simusol?hl=es su dirección es:
mailto:simusol@googlegroups.com.

Saravia, D. (2012a). LiberaWin: Instalador para Windows de Simusol.
http://www.simusol.org.ar/downloads/LiberaWin.exe.

Saravia, D. (2012b). SUMAPACK: Herramientas para la compilación, empaquetado e instalación de software libre en diferentes arquitecturas, sistemas operativos y distribuciones GNU/Linux.
http://www.icfoss.org/ojs/index.php/foss/article/view/4
También en AVERMA Avances en Energias Renovables y Medio Ambiente. Vol. 14 2010 8.77
http://www.cricyt.edu.ar/lahv/asades/modulos/averma/trabajos/2010/2010-t008-a011.pdf.

Saravia, D. and Alía de Saravia, D. (2010). Mejoras al Sceptre. El paquete Sceptre Inenco.
http://www.cricyt.edu.ar/lahv/asades/modulos/averma/trabajos/2010/2010-t008-a010.pdf.

Wall, L. (2012). The Perl Programming Language.
http://www.perl.org.

Wikipedia (2010). Sistema de gestión de paquetes.
http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_paquetes.

Wikipedia (2012). Open Document.
http://en.wikipedia.org/wiki/OpenDocument.

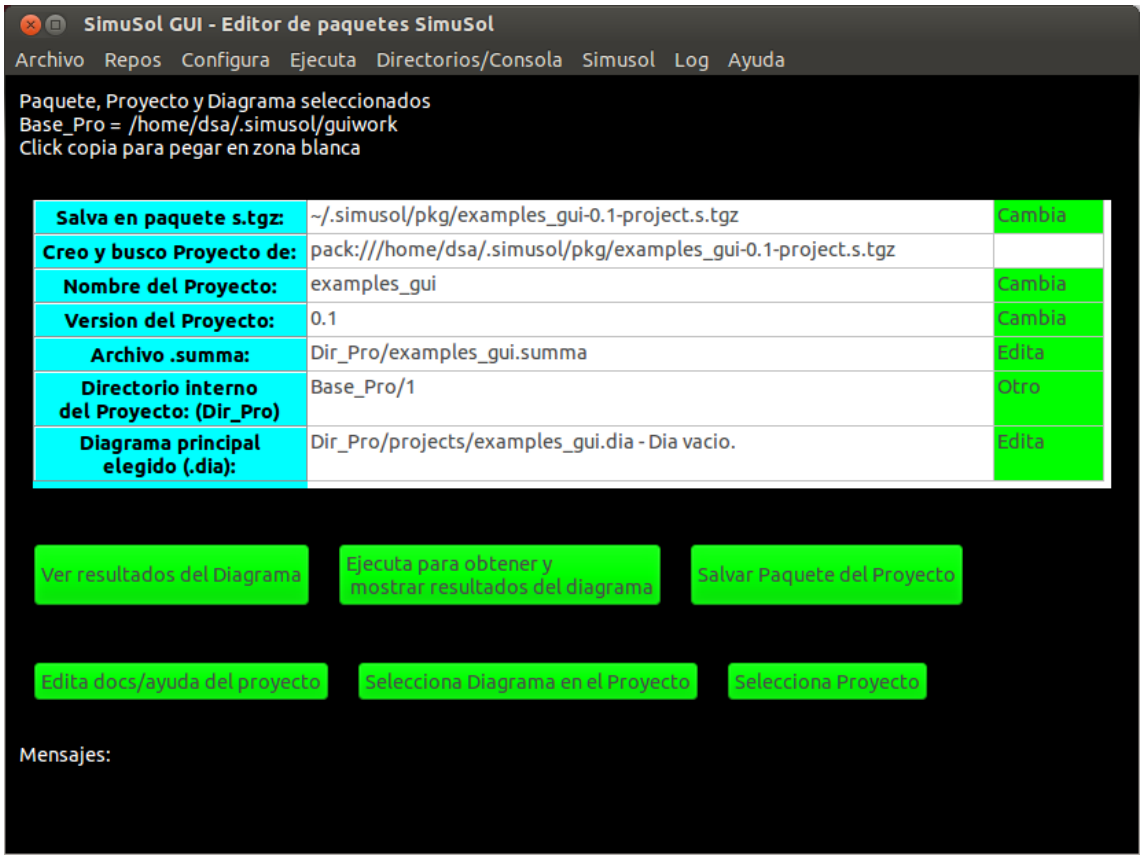



**Fig. 1: Main Screen of simusol-gui**

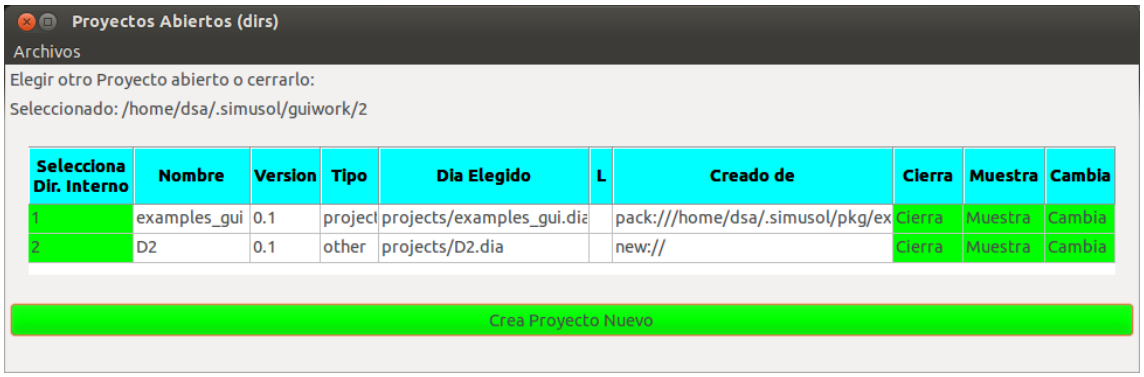



**Fig. 2: Package Selection Screen.**