OPOPEN CASCADE is pleased to announce SALOME version 4.1.4sp2. It is a public release that contains corrections of the several bugs in Post-Pro module against SALOME version 4.1.4 released in December of 2008.

Table of Contents

- GENERAL INFORMATION ............................................................................................................. 1
- NEW FEATURES, IMPROVEMENTS AND GENERAL CHANGES ........................................................... 2
  Prerequisites changes ................................................................................................................. 2
  New installation procedure ........................................................................................................ 3
  Various developments implemented by CEA/EDF/OCC partners ............................................. 4
  Porting on Windows platform .................................................................................................... 5
  Filter module ............................................................................................................................... 6
  Implementation of MED_SPLITTER .......................................................................................... 7
  New MULTIPR module .................................................................................................................. 7
  New mesher plugins .................................................................................................................. 9
  YACS module (new SUPERVISOR) ........................................................................................ 10
  Slider and Cache System .......................................................................................................... 12
  KERNEL: Replacement of the OCAF data model with a non OCC data model ................................ 14
  Porting of SALOME KERNEL module on a new XML parser (libxml2) ..................................... 15
  Porting on automake .................................................................................................................. 16
  MED unitary tests ...................................................................................................................... 19
  SALOMEDS unitary tests ............................................................................................................ 21
  Porting on omniORB 4.1 .......................................................................................................... 23
  Porting on VTK 5.0 .................................................................................................................... 23
  GUI Module ............................................................................................................................... 24
  GEOM Module ........................................................................................................................... 24
  SMESH Module ......................................................................................................................... 29
  VISU Module ............................................................................................................................. 34
  Other improvements ................................................................................................................... 36
- SUPPORTED LINUX DISTRIBUTIONS AND PRE-REQUISITES .......................................................... 39
  Supported platforms .................................................................................................................. 39
  Software Requirements ............................................................................................................. 40
- HOW TO INSTALL AND BUILD SALOME .................................................................................... 41
- SALOME SYSTEM REQUIREMENTS .......................................................................................... 41
- HOW TO GET THE VERSION AND PRE-REQUISITES .................................................................. 41
- KNOWN PROBLEMS AND LIMITATIONS ..................................................................................... 42

Copyright © 2001- 2009. All rights reserved.
New features, Improvements and General Changes

Prerequisites changes

Open CASCADE Technology 6.3

Salome 4.1.4sp2 is based on Open CASCADE Technology 6.3 version, the latest OCCT release, which includes fixes of some bugs that came from SALOME. The most important features of the new OCCT version are:

- Support of UTF8 encoding for extended strings, and Unicode symbols in IGES
- Next step in thread-safety: protection against concurrent construction / destruction of Handle objects
- Improved compatibility with STL and Windows-specific code
- Multiple new features introduced in visualization module
- New visualization library NIS (New Interactive Service)
- New standard attributes and numerous improvements in OCAF
- Integrated code changes made by OCC users for MacOS X and FreeBSD porting
- Improved support of perspective view in Open CASCADE viewer
- New version of the OCAF binary persistence format
- The functionality of reading/writing VRML2.0 files has been implemented
- The definitions of Standard_CString and Standard_ExtString (typedefs) have been changed to be const: from char * (or short *) to const char * (or const short *)
- New supported platforms: Windows Vista, Mandriva 2006, 2007, 2008, Debian 4.0 (Etch), Red Hat Enterprise 4.0
- New supported compilers: gcc 4.0-4.2, Visual C++ 8.0

Important note: Since NETGEN mesher depends on OCCT, it is necessary to recompile NETGEN mesher with the new Open CASCADE Technology version.

Open CASCADE Technology 6.3 patch

Installation procedure for Salome 4.1.4sp2 contains a patch for Open CASCADE Technology 6.3 available as an independent "Patch for Open Cascade" checkbox item in the "Prerequisites" page. The patch updates binary and/or source files of the previously installed Open CASCADE Technology 6.3, fixing a number of important bugs. By default this feature is unchecked.

Important note: In case of "Install Binaries" installation type, the patch applies only to Open CASCADE Technology 6.3, while SALOME binaries are installed as built with Open CASCADE Technology 6.3 without the patch applied. In case of "Install sources and make compilation" installation type this feature allows building SALOME modules with patched Open CASCADE Technology 6.3.
Other prerequisites changes

The SALOME version 4.1.4sp2 has been also migrated to the new versions of most of pre-requisite products comparing with version 3.2.6. Moreover, the SALOME Installation procedure now provides the same versions of products for all the supported Linux platforms.

**Changed versions of products:**

- Python 2.4.4
- Open CASCADE Technology 6.3
- OmniORB 4.1.0
- Med 2.3.5
- Vtk 5.0
- Qt 3.3.8
- Sip 4.7.3
- PyQt 3.17.4
- Boost 1.34.1
- Swig 1.3.31
- Tcl/Tk 8.4.14
- Numeric 24.2
- Graphviz 2.16.1
- Doxygen 1.5.6
- docutils 0.3.9

**New prerequisite products:**

-metis 4.0
-scotch 4.0
-libxml2 2.6.27
-elementtree 1.2.6
-cElementTree 1.0.5

**New installation procedure**

SALOME 4.1.4sp2 installation can be performed with the renewed installation procedure developed in accordance with the specification prepared by joint efforts of CEA and Open CASCADE companies.

SALOME 4.1.4sp2 Installation Wizard supports Linux Mandriva 2006 (32bit and 64bit), Linux Mandriva 2008 (32bit and 64bit), Linux Debian 3.1 Sarge and Linux Debian 4.0 Etch (32bit and 64bit). For other platforms, you can use universal binaries or products and SALOME modules built from source files. The universal binaries are a set of system and SALOME binaries prepared with Pbuilder tool. This set is fully self-sufficient and allows starting Salome on any Linux platform after the installation.
The SALOME Install Wizard can be launched in GUI or console mode. In both modes the wizard automatically detects current platform and parses the corresponding XML configuration file. If the platform is not supported, the installation procedure prompts user to choose a desirable operation system.

A new Installation Wizard provide 3 types of the installation of SALOME platform:

- Install binaries – unpack binaries of the modules and prerequisites.
- Install sources – only unpack sources of the modules and prerequisites.
- Install sources and make a compilation – unpack sources of the modules and prerequisites and run build process.

The mechanism of prerequisites automatic selection simplifies a process of products choosing to be installed.

In addition user has an option to install SALOME modules without GUI libraries – “Installation with GUI” check box can be used for this mode. Another advanced option is a possibility to install all the SALOME modules to a single directory.

Various developments implemented by CEA/EDF/OCC partners

- Implementation of MED_SPLITTER functionality from CEA
- New SMESH plug-ins from CEA
  - GHS3DPRL (parallel)
  - HEXOTIC
  - BLSURF
New FILTER module from CEA and the corresponding MED module modifications

Porting to DSC from EDF

New MULTIPR module from EDF

New YACS module

Slider and cache functionality in VISU module

Porting on Windows platform

SAŁOME version 4.1.4sp2 is now Win32-compatible but with some limitation, listed below. In general, Win32 compatibility means that SAŁOME can be compiled and run on Windows. No compilation procedure is provided for the Windows platform. The same concerns pre-requisite products required for building and running of SAŁOME platform. The binaries of pre-requisite products for Windows are not distributed, since some of them require commercial license for Windows.

To run SAŁOME on Windows, use the launchSalome.py Python script from the KERNEL module distribution. The launchSalome.py script is a cross-platform version of the runSalome shell script.

The following modules have been migrated to Windows platform:

- KERNEL module with the following exceptions
  - Packages implementing the parallel calculations technology are not ported to Windows: MPIContainer, ParallelContainer, TestMPIContainer.
  - Batch package is not fully ported because of the problem with SGE, LSF and PBS pre-requisite products. The other problem of Batch module is using bash shell scripts to run remote execution.
  - DSC packages are not ported to Windows.
- GUI module (see remark below about the PyQt product)
- MED module
  - Packages MEDSPLITTER and MULTIPR are not ported because there are no Windows versions of Metis and Scotch products.
- GEOM module
- SMESH module
- VISU module
- NETGENPLUGIN module

Other limitations:

- There is a problem with PyQt3 (Riverbank Computing Ltd) product which requires a commercial license on Windows. Thus, SAŁOME_PYQT package of GUI module can not be build on Windows if the user does not have a commercial license of PyQt3. Hence, the corresponding functionality – Python wrappings of GUI module will be not available in SAŁOME.
Filter module

The necessity of increasingly finer results of simulation on complex geometries requires the visualization of the obtained results. Indeed, the meshes used to carry out the numerical simulation cannot be entirely visualized. The purpose of the FILTER module is to decimate the grid according to the solutions, in order to allow their visualization or, more generally, their post-processing. In the same time, it will have to maximally preserve the geometry of the field and the precision of the solution.

**Important note: restriction**

FILTER module needs a previously installed *filttoo* tool by DISTENE Company:

Distene S.A.S.  
Pôle Teratec - BARD-1  
Domaine du Grand Rué  
91680 Bruyères-le-Chatel  
FRANCE

The picture below shows a sample of work of the FILTER module:

![Sample of work of the FILTER module](image)

**Fig 1. Sample of work of the FILTER module**

For more detailed information, please, refer to the SALOME documentation (*filter_html_doc.tgz*) delivered with the installation procedure.
Implementation of MED_SPLITTER

MED_SPLITTER is a new SALOME tool to create distributed MED files (in the MED v 2.3 format) by partitioning the cell graph of meshes with SCOTCH or METIS libraries. The fields contained in the original MED file can be projected to the new files. The tool can also be used to redistribute a distributed MED v 2.3 file to a new number of subdomains.

The functionalities can be accessed either as a standalone program (medsplitter) or through a C/C++ API.

The splitter creates independent meshes and fields for each subdomain and adds global numbering and subdomain boundary connectivity information.

In the current version, meshes containing polyhedral and polygons cannot be read by the splitter.

New MULTIPR module

MULTIPR functionality is intended to simplify the processing of big meshes (over 1 000 000 mesh elements) with a large number of timestamps (over 1 000). The main idea of the simplification is to distribute data of a large MED file to smaller ones that will represent only a part of the initial one with a desired level of data resolution. It suggests the following algorithm to perform this task.

- Partition of grains - splitting the initial mesh according to predefined groups of mesh elements. (This feature is strongly necessary if the initial data represents a heterogeneous model.)
- Partition based on specific algorithms - splitting the initial mesh according to some initial restrictions and conditions (for example, desired number of partitions).
- Decimation based on specific algorithms – reducing the amount of input data by passing only the most significant ones to the output.

Currently MULTIPR works in terms of files. It takes a MED file as the input and produces a set of MED files as the output according to the user request. To keep consistent with the produced result MULTIPR generates an ASCII file ("maitre") which describe mapping of the initial file to the set of output MED files according to the given input parameters.

MULTIPR functionality is split between two SALOME modules – MED and MULTIPR. SALOME MED module contains the core functionality that can be used in command line mode. SALOME MULTIPR module provides CORBA and GUI interface for the core MULTIPR functionality.
Fig 2. MULTIPR module

The core MULTIPR functionality is also used in VISU to have a possibility to visualize the distributed mesh data that is produced by SALOME MULTIPR module.

Fig 3. Result of processing of med file in MULTIPR module
To build the core MULTIPR functionality it is necessary to have the following prerequisites installed.

- MED library version of 2.3.1 or above.
- METIS 4.0 or SCOTCH 4.0 partition algorithms libraries.

**New mesher plugins**

SALOME version 4.1.4sp2 includes new HEXOTIC, GHS3DParallel and BLSURFPLUGI mesher plugins.

*Important note*: BLSURFPLUGI mesing plugin requires a previously installed Distene/BLSurf product for the compilation.

BLSURD and GHS3D plugins

**GHS3D**

New GUI for GHS3D plugin allows defining basic and advanced mesh parameters and working conditions of the application.

At the basic level it is enough to choose the degree of optimization, trading quality for time and to decide whether the cavities inside the shape should be meshed.

At the advanced level the user can additionally define: maximum and initial amount of RAM occupied by the plugin; the directory for its input and output files; the level of detail of output report, the possibility to create additional nodes during 3D meshing; the usage of boundary recovery mechanism and command line input.

Error codes reported by GHS3D are translated into human readable problem descriptions.

Object browser icons are provided for the algorithm and its hypothesis.

The problems with incorrect association between tetrahedrons and shapes and failed meshing of internal solids of a shape resulting from partitioning have been solved.
The functionalities of BLSURF plugin have been fully implemented into Salome GUI.

All parameters are split into basic and advanced. The basic parameters allow defining minimum and maximum physical and geometrical size; the ratio between the lengths of two adjacent edges; mesh element deflection from curves and surfaces; the possibility to create quadrilateral elements; the possibility to mesh all geometrical faces are meshed as one hyper-face and the level of detail of output report. The user can also define 32 advanced options.

Other new features:
Object browser icons are provided for the algorithm and its hypothesis.
Uniqueness of default names of hypotheses has been guaranteed.
The problem of locking BLSURF license at abnormal termination of BLSURF work has been fixed.
Redundant commands setting algorithm parameters have been eliminated from python dump script.

YACS module (new SUPERVISOR)
A new light SALOME module YACSGUI included in SALOME release is intended to replace the old SALOME Supervisor module. Its main purpose is creation of distributed calculation schema and management of its execution.

Among YACSGUI functional characteristics the following ones are most important:

- Extended set of schema nodes is supported (composed nodes of several kinds, new loop types and inline node subtypes were added)
- Data structures were designed to be clear and scalable.
- YACS engine libraries can be used either directly as ordinary C++ libraries or via CORBA interface that makes remote access to YACS services from another processes or workstations possible, and on the other hand interaction between GUI and engine is much more efficient than in the old Supervisor.
- Built-in calculation resources manager will choose the most suitable workstations for launching each node.
- Import and export of the new format of XML files containing schema description and import of old Supervisor XML files.

For the new module, advanced GUI was developed. It has the following key features:

- User-friendly, highly informative presentations of schema nodes including tooltips showing more details about an object (node or port) and progress bars displayed in a node when it is running.
- Manual arrangement of schema nodes and links between nodes.
- Automatic link building algorithm that builds orthogonal links for a whole schema.
- Interactive highlighting of nodes, ports and links when the mouse cursor is over an object.

Fig 7. Small graph presentation including inline, service, bloc and ForEachLoop nodes during graph execution

Fig 8. Highlighted port and a tooltip for this port

Fig 9. Large graph during execution

SALOME 4.1.4sp2 includes the first version of YACSGUI that is actually a prototype. In the future, functionality and GUI of this module will be extended, in parallel with continuous improvement of robustness of the module:
- Schema modification and validation will be provided.
- New schema view modes will be provided (control and table views).
- Performance of schema visualization will be improved.
- Automatic schema layout algorithm will be available that will allow the user to arrange graph nodes neatly with a single mouse click.
- Qt 4 advanced 2D drawing capabilities will be used (high-performance drawing tools, alpha blending)

**Slider and Cache System**

The new “slider” functionality implemented in VISU deals with 3D presentations that visualize data assigned to the mesh (for example VISU ScalarMap, DeformedShape and e.t.c). From now on users can investigate the input data in two basic ways – “one presentation per one timestamp” or “animation” ways.

In the first case, the user should create one presentation per one timestamp for the given physical value to obtain the whole picture of the investigated physical process. In this case the user has full control over the created presentations because it is possible to adjust every presentation parameter according to local features of the investigated data. Moreover, this way of data investigation requires minimum machine resources to produce its results. From the other hand, there are several weakness of this way of data investigation.

- It is too tiresome to create the presentation manually especially if the number of timestamps for the investigated field is more than 10.
- There is no way in GUI to synchronize the presentation parameters between sets of presentations that belong to different timestamps.
- It is very confusing to manage the visibility of presentations because they have the same position in space.
- If the user creates and holds presentations for each timestamp there are no advantages in used machine resources.

When the user investigates the input data through “animation” functionality he avoids almost all the problems that appeared using the first way except one – presentations are created for all timestamps. This feature of animation functionality does not allow to use it for large meshes (over 1 000 000 mesh elements) with a huge number of timestamps (over 1 000). Just due to the same reason, modification of parameters of animated physical value leads to regeneration of all presentations that correspond to different timestamps of the physical value.

The introduced “slider” functionality allows solving most of the weakness of the two existing ways of data investigation. From GUI point of view, the “slider” functionality greatly simplifies data investigation in the following topics.

- The “Slider” functionality allows investigating the whole physical value by using a single presentation.
- The same VISU Slider presentation can be used for investigation of different physical values. To do this, it is just necessary to define another physical value as the input of the presentation.
- The “Slider” functionality greatly simplifies visibility management for the presentations in comparison with “one presentation per one timestamp” way of data investigation. In other words, if the user wants to see some timestamp it is enough to move the slider in the corresponding position of time (as in “animation” way of data investigation).
- The “Slider” functionality allows producing investigation of data both in synchronized and in separated modes. In other words, the “Slider” functionality can be applied both for one
presentation and for a set of them and if the “slider” processes a set of presentations, these presentations are synchronized in terms of timestamps.

Fig 10. Slider in the VTK viewer

The “Slider” functionality interacts closely with the “Cache System”. While the “Slider” functionality is responsible for GUI management of presentations, the “Cache System” is responsible for internal memory management of the presentations. This “Cache System” has two main operation modes – “minimal” and “limited”. 
In the “minimal” mode the “Cache System” holds only one presentation per one physical value. This mode allows minimizing memory required for the investigation of a physical value. As a result each visualization of the next timestamp leads to complete recalculating of the presentation, that reduces the performance of the slider functionality.

In the “limited” mode user can define a limited amount of memory that can be used for the visualization. In this mode the “Cache System” tries to find the already cached presentation to visualize next timestamps. This algorithm of presentations management allows playing with slider at the same performance as for animation functionality. This strategy of memory usage allows the user to manually define the most suitable conditions to investigate the input data.

As a result, the “Slider” and “Cache System” functionalities allow to investigate large meshes (over 1 000 000 mesh elements) with a huge number of timestamps (over 1 000) in the most suitable way both from the user interaction and from the performance point of view.

**KERNEL: Replacement of the OCAF data model with a non OCC data model**

The previous implementation of the KERNEL data model was based on the OCC component OCAF. While OCAF-based data model completely met the requirements for KERNEL data model it only partially used the OCAF functionality and at the same time required the whole OCC product as a prerequisite for KERNEL. Thus, whenever the KERNEL was ported to another platform it was necessary to port the OCC product on this platform as well, which was a time-consuming task and could not be done in a feasible time in some environments. If the functionality required for KERNEL data model were implemented without using the OCC OCAF component, there would not have been any need to have the OCC product as a prerequisite for KERNEL module and the KERNEL porting and redistribution would have become significantly simpler.

The OCAF data model has been replaced by a data model that has the same data structure: it consists of a tree of labels, where each label can have one or several attributes.

In KERNEL component, the implementation of the data model has been separated from access methods.

The KERNEL data model consisted of two main parts: package SALOMEDSImpl that contained an OCAF implementation of the data model and package SALOMEDS that contained a CORBA layer of the OCAF implementation and the implementation of SALOMEDSClient abstract interface through which an access to SALOMEDS services was provided. As SALOMEDS package did not contain the implementation of the data model, the previous API was left untouched, the structure of SALOMEDSImpl classes was modified and the OCAF data model was replaced with its counterpart built using various STL containers to simplify the porting procedure in the future. Thus any modifications of KERNEL data model would become transparent to the clients of SALOMEDS services.
The SALOMEDSImpl package has been split in two:

- **SALOMEDSImpl** contains a Salome-specific implementation of the data model: Study, StudyBuilder, StudyBuilder, SObject, SComponent, UseCaseBuilder and UseCaseIterator, AttributeStudyProperties.

- **DF** (data framework) contains a generic data model implementation based on a tree of labels with attributes. The following high-level classes have been added to this package:
  - Label is a counterpart of OCAF TDF_Label class; each label can contain one or several attributes.
  - Document is a container of a tree of Labels
  - Application permits to control the creation, existence and closing of the Documents
  - Attribute – the basic attribute inherited by all attributes.

To reproduce the required OCAF features several low-level classes have been added in DF package.

The existence of a separate DF package allows the developer of the components to reuse this data framework as a non-CORBA data model of a new component. This data model depends only on the standard C++ libraries and STL, so the developer is not forced to use components of other products.

**Important note:** The transaction mechanism in the newly created Data model is not used in the current version of SALOME platform.

### Porting of SALOME KERNEL module on a new XML parser (libxml2)

The purpose of the porting was to exclude all Qt library dependencies from KERNEL module of SALOME. Previously, only XML-parsing features of Qt toolkit were used in KERNEL (Qt XML module). Qt, as a GUI development toolkit, in its turn depends on other libraries (for example, OpenGL).

The usage of libxml2 XML parser instead a QT parser and, consequently, the exclusion of QT from the pre-requisites for SALOME KERNEL module allow decreasing external dependencies and avoid linking KERNEL libraries to the GUI tools which are not used directly in this SALOME module.

Previously (SALOME version 3.2.x) XML files were used in two packages of SALOME KERNEL module: ResourcesManager and ModuleCatalog. Both packages implemented a parser of XML files for reading of the specific information. In addition, ResourcesManager package implemented the possibility to write the resource files in XML format.

The following Qt classes were used:

1. SAX2 (Simple API for XML) classes: QXmlDefaultHandler, QXmlSimpleReader.

QXmlSimpleReader class provided an implementation of a simple XML reader.

QXmlDefaultHandler class provided an interface to report the logical content of XML data and an interface to report errors in XML data.

QDomDocument class allowed converting the document to its textual representation for writing the XML files.

### Using libxml2 instead of Qt XML module

The libxml2 contains an open source XML C parser and toolkit developed for the Gnome project; it is free software and available under the MIT license. The libxml2 works on Linux/Unix, Windows, Mac OS X, ported to a number of other platforms. Most Linux and BSD distributions include libxml2 (for example, Linux Mandrake 10.1).
Reading XML files

The libxml2 toolkit also (as a Qt library) includes a SAX-like interface for reading XML files. The API for processing document-related events in libxml2 is similar to the API provided by Qt library. There is a handler - xmlSAXHandler structure, which is similar to the QXmlDefaultHandler class and also bases on SAX2 standard interface. As Qt, libxml2 library allows creating a reader using a filename and reading information from an XML file by iterating over all nodes in the document.

Writing XML files

The libxml2 toolkit also provides the possibility to build an in-memory DOM representation using xmlDoc structure (QDomDocument class is used for this purpose in Qt) and dump it to the file.

Using C++ API (libxml++)

Libxml++ is a C++ wrapper for the libxml2 XML parser library (http://libxmlplusplus.sourceforge.net). It requires libxml2 library and can be installed from sources on Linux and Windows, using any modern compiler, such as g++ or MSVC++. The libxml++ toolkit is distributed under LGPL version 2 license.

Libxml++ contains SAX XML parser class – xmlpp::SaxParser, which is enough for parsing XML files in KERNEL. This class should be used for reading operations instead of QXmlDefaultHandler and QXmlSimpleReader.

Writing operation was implemented with the help of xmlpp::Document class (it allows writing the document to a file). This class is used instead of the previously used QDomDocument class of Qt library.

Porting on automake

GUI module

Below are the basic features of this update:

- All Makefile.in, make_begin.in, make_conclude.in, make_dependencies.in have been deleted, Makefile.am, make_common_starter.am have been added instead.

- Generated link salome_adm to the KERNEL_SRC folder has been submitted by the generated folder with copied files: this is because automake tries to create some files in this folder and if KERNEL_SRC is read-only, it fails.

- libSalomeIDLGUI.la library has been created by all idl stubs of GUI (only one SalomeApp_Engine.idl in fact) like in KERNEL_SRC.

- All conditional compilation flags have the same meaning. Now AM_CONDITIONAL is used to satisfy them.

- During this work we found some unused files and they were removed from the CVS:
  - SalomeApp_Displayer.cxx
  - SalomeApp_Displayer.h
  - supervisionexample.py (supervisionexample.py.in is used)
  - supervisiongeomexample.py (supervisiongeomexample.py.in is used)
  - testattr.py
  - Session/runSession
  - SUIT/utillies.h
  - VTKViewer_RectPickler.h
SALOME : The Open Source Integration Platform for Numerical Simulation

- VTKViewer_RectPickler.cxx
- VTKViewer_CellRectPickler.h
- VTKViewer_CellRectPickler.cxx
- plot2d_rotate.png

Besides, since the GUI_SRC CORBA packages may use libSalomeIDLKernel.so instead on self idl-stubs generating, KERNEL_SRC/idl/Makefile.am was modified to share generated "SK.hh stubs in KERNEL_INSTALL/include files. For KERNEL_SRC the "BR_GUI_AutoTools" branch has also been created.

Changes in GUI_SRC done during porting to automake:

Procedure of compilation and options are the same like in the previous version:

- build_configure
- configure
- make
- make install

However during the build_configure step Makfile.in files are generated from Makefile.am in the source directory. Thus the SALOME "automake" tool has been added to the list of prerequisites.

Otherwise Makefile.in can be generated on another station and used on the station without "automake".

Since this version developers have to modify Makefile.am files to change compilation/installation/distribution procedure of GUI_SRC. After this the "make" command automatically rebuilds all necessary files.

It is also necessary to have installation directory for the application launching (like for KERNEL_SRC now) because "build" directory does not have right structures of files for this.

The following common variables of Makefile.am are listed below for easy update of the compilation procedure:

- lib_LTLIBRARIES = name of the generated library or a list of names
- salomeinclude_HEADERS = list of exported headers
- dist_<name of library>_SOURCES = list of sources to compile
- MOC_FILES = list of generated moc files (for example: CAM_Study_moc.cxx - it uses CAM_Study.h for generation)
- nodist_<name of library>_SOURCES = $(MOC_FILES) - this string is for adding MOC files to the library
- nodist_salomeres_DATA = list of qm files to be compiled to .po resources
- dist_salomeres_DATA = list of resource files (like .png images etc)
- <name of library>_CPPFLAGS = list of C++ flags for compilation
- <name of library>_LDFLAGS = linking flags
- <name of library>_LIBS = libraries for linking your library

User can easily find other variables in Makefile.am to be changed if necessary.

GEOM Module

Main changes:
All Makefile.in files have been replaced by corresponding Makefile.am

- ./buildconfigure file - all configure files from KERNEL and GUI salome_adm catalogs have been copied (instead of inks)
- ./configure.ac - added instead of configure.in.base
- ./AUTHORS - added for automake tool
- ./ChangeLog - added for automake tool
- ./COPYING - added for automake tool instead of LICENSE file.
- ./NEWS - added for automake tool
- ./README - added for automake tool
- ./adm_local/unix/make_common_starter.am - added as a common header of all Makefile.am files (instead make_commons.in and make Omniorb.in)
- ./resources/Makefile.am - added as a part of configure.in.base file for the automake tool for SMESH resources files.

SMESH Module

Using AutoTools for building procedure.

Main changes:

- All Makefile.in files have been replaced by corresponding Makefile.am
- ./buildconfigure file - all configure files from KERNEL, GUI, MED and GEOM salome_adm catalogs have been copied (instead of inks)
- ./configure.ac - added instead of configure.in.base
- ./AUTHORS - added for automake tool
- ./ChangeLog - added for automake tool
- ./COPYING - added for automake tool instead of LICENSE file.
- ./NEWS - added for automake tool
- ./README - added for automake tool
- ./adm_local/unix/make_common_starter.am - added as a common header of all Makefile.am files (instead make_commons.in and make Omniorb.in)
- ./resources/Makefile.am - added as a part of configure.in.base file for the automake tool for SMESH resources files.

NETGENPLUGIN Module.

Main changes:

- All Makefile.in files have been replaced by corresponding Makefile.am
- ./buildconfigure file - all configure files from KERNEL, GUI, MED, GEOM and SMESH salome_adm catalogs have been copied (instead of inks)
- ./configure.ac - added instead of configure.in.base
- ./AUTHORS - added for automake tool
- ./ChangeLog - added for automake tool
SALOME : The Open Source Integration Platform for Numerical Simulation

- /COPYING - added for automake tool instead of LICENSE file.
- /NEWS - added for automake tool
- /README - added for automake tool
- /adm_local/unix/make_common_starter.am - added as a common header of all Makefile.am files (instead make_commens.in and make_omniorb.in)
- /resources/Makefile.am - added as a part of configure.in.base file for the automake tool for NETGENPLUGIN resources files.

MED and VISU modules

Main changes:
- New text files have been added to the ROOT folder (needed for automake working): AUTHORS, COPYING, ChangeLog, NEWS, README
- The system of configure.base.in files has been replaced by configure.ac file. configure is generated by 'autoconf' (on build_configure step).
- All Makefile.in files are generated from new Makefile.am files by 'automake' (no build_configure step).
- salome_adm folder is no longer a link. All necessary files of this folder have been copied. 'automake' makes some manipulations in this folder, thus it must have permissions for writing.
- adm_local/unix/make_commence.in and adm_local/unix/make_conclude.in have been replaced by make_common_starter.am like in KERNEL and GUI modules.
- Instead of each library including, all files, generated from idl-s now collected in 'idl' folder into libSalomeIDLMED.so and libSalomeIDLVISU.so. Benefits of this solution: much faster compilation and smaller size of libraries.

GHS3DPLUGIN and EXAMPLES modules

Following modules have been ported on automake utilities as well:
- GHS3DPLUGIN
- EXAMPLE modules
  - COMPONENT
  - CALCULATOR
  - PYCALCULATOR
  - HELLO
  - PYHELLO
  - LIGHT
  - SIERPINSKY
  - RANDOMIZER

MED unitary tests

Development of cppunit tests for MEDMEM has been carried out for this version of Salome.

New subfolder “Test” has been added to the MEDMEM package to hold all implemented cppunit tests for MEDMEM.
40 test cases were created during this task implementation. 11 of them are successfully executed till the end; other 29 tests execution is broken by problems, currently existing in MEDMEM. Besides, 13 separately executable tests were created for cases, currently leading to Segmentation Fault.

About 50 problems were detected in MEDMEM during tests implementation. All of them are to be registered in the bugtracker.

**Functionalities covered by MEDMEM cppunit tests**

Implemented tests cover classes and methods, declared in the following headers from MEDMEM package:

```
MEDMEM_Array.hxx
MEDMEM_ArrayConvert.hxx
MEDMEM_AsciiFieldDriver.hxx
MEDMEM_Connectivity.hxx
MEDMEM_Coordinate.hxx
MEDMEM_DriverFactory.hxx
MEDMEM_Exception.hxx
MEDMEM_Family.hxx
MEDMEM_Field.hxx
MEDMEM_FieldConvert.hxx
MEDMEM_Formulae.hxx
MEDMEM_GaussLocalization.hxx
MEDMEM_GibiMeshDriver.hxx
MEDMEM_Grid.hxx
MEDMEM_Group.hxx
MEDMEM_IndexCheckingPolicy.hxx
MEDMEM_Init.hxx
MEDMEM_MedFieldDriver21.hxx
MEDMEM_MedFieldDriver22.hxx
MEDMEM_Med.hxx
MEDMEM_MedMedDriver21.hxx
MEDMEM_MedMedDriver22.hxx
MEDMEM_MedMeshDriver21.hxx
MEDMEM_MedMeshDriver22.hxx
MEDMEM_MedVersion.hxx
MEDMEM_Mesh.hxx
MEDMEM_Meshing.hxx
MEDMEM_ModulusArray.hxx
MEDMEM_nArray.hxx
MEDMEM_PointerOf.hxx
MEDMEM_PolyhedronArray.hxx
MEDMEM_PorflowMeshDriver.hxx
MEDMEM_SkyLineArray.hxx
MEDMEM_STRING.hxx
MEDMEM_Support.hxx
MEDMEM_TopLevel.hxx
MEDMEM_TypeMeshDriver.hxx
MEDMEM_Unit.hxx
MEDMEM_VtkFieldDriver.hxx
MEDMEM_VtkMedDriver.hxx
MEDMEM_VtkMeshDriver.hxx
```

**Tests implementation**

One test case was created per each header file from the above list. Commonly each test case is implemented in a separate c++-source file, except for some short cases, implemented directly in MEDMEMTest.hxx or grouped together with other cases. File MEDMEMTest.hxx contains information about all tests location.

Each test case is supplied with a comment, where all tested methods and methods are listed, defined in the related header, but not tested for some reason, with corresponding marks:
Each found problem in test body is surrounded with

```cpp
#ifdef ENABLE_FORCED_FAILURES
#else
#endif
```

This is generally done to make tests development easier and can be useful for people who will enrich these tests with cases for new functionalities. Also these statements can be used to easily find all detected MEDMEM problems in tests sources.

Tests execution

To run MEDMEM cppunit tests (built on the machine with cppunit installed) in shell with Salome environment:

- cd ${MED_ROOT_DIR}/src/MEDMEM/Test
- make tests (to test all cases, both general and faulting)
- TestMEDMEM (to run only general test cases)
- Look at the result in ${MED_ROOT_DIR}/src/MEDMEM/Test/UnitTestsResult file.
- To check any of 13 faulting cases just run the corresponding executable, for example MEDMEMTest_Support_fault. Results of faulting cases are not registered in UnitTestsResult file.

**SALOMEDS unitary tests**

Development of cppunit tests for SALOMEDS was carried out for this version of Salome.

The tests cover 38 classes in SALOMEDS package. The tests are performed for both: standalone and embedded modes of running SALOMEDS services.

During the tests running 8 problems were found. 3 of them have been located and fixed. The fixed sources have been put in BR_SALOMEDS_Unit_Tests branch. Other 5 problems must be studied to locate their reasons and fixed.

Functionalities covered by SALOMEDS cppunit tests

Implemented tests cover classes and methods, declared in the following headers from SALOMEDS package:

**Note:** * means both: `<class_name>.hxx` and `<class_name>_i.hxx` classes are covered by the tests.

- `SALOMEDS_AttributeComment(*)`
- `SALOMEDS_AttributeDrawable(*)`
- `SALOMEDS_AttributeExpandable(*)`
- `SALOMEDS_AttributeExternalFileDef(*)`
- `SALOMEDS_AttributeFileType(*)`
- `SALOMEDS_AttributeFlags(*)`
- `SALOMEDS_AttributeGraphic(*)`
- `SALOMEDS_AttributeIOR(*)`
Tests implementation

For each header file one test case was created that covers all methods of the given class (Except for UseCaseBuilder and UseCaseIterator for which the common test case UseCase is created). All tests are implemented as separate files with names coinciding with the names of the tested files.

Note: Test cases for SALOMEDS_AttributeTreeNode and SALOMEDS_UseCaseBuilder causes “Segmentation fault” due to the problem with the class implementation. In order to allow the other cases to run the problem placed were surrounded by preprocessor directives:

```cpp
#ifdef SALOMEDS_ALL_TESTS
#endif
```

Which by default is not defined till the fix will be available and the message:

"THE TEST IS NOT COMPLETE !!!" is output.

There are comments placed inside each use case starting with “Checking method” + <method name> before the expression that checks the mentioned method.

Tests execution

To run SALOMEDS cppunit tests (built on machine with cppunit installed) in shell with Salome environment:

- Set up a Salome environment
- Go to the directory for which the user has permissions to write.
- Type TestSALOMEDS (to run test cases for SALOMEDS)
- Look at the result in UnitTestsResult file created in the current directory and at the shell output.
Note 1: The program that launches the tests supposes by default that port 2810 is free. One may choose another first free port starting from 2810 by setting the environment variable SALOMEDS_UNITETETS_PORT.

Note 2: After porting of environment for KERNEL component to “autoconf” the “make tests” is no longer available.

Note 3: In order to check method LoadWith of SALOMEDS_StudyBuilder it is necessary to have the SuperVision component installed and available for loading.

Porting on omniORB 4.1

SALOME was successfully ported on OmniORB version 4.1.0 version

Below is a list of packages in KERNEL module that has a link to RefCountServantBase which was adopted to new omni 4.1 inheritance:

- Communication
- ModuleCatalog
- SALOMEDS
- NamingService/Test
- GenericObj
- Logger
- Container

Other modifications in SALOME which was required when porting to OmniORB 4.1.0:

- KERNEL module, TestContainer package: pd_refCount became _pd_refCount
- KERNEL module, Notification package, NOTIFICATION_Consumer::Receive class: changing in operator >>= of Any type
- SUPERV module, GraphBase, GraphEditor, GraphExecutor packages: changing in operator >>> of Any type

Porting on VTK 5.0

During the migration on the new version pf VTK 5.0 the following documentation was used:

- Roadmap: What changes are being considered for VTK
- Text properties in VTK 4.2

Changes in SALOME after VTK 5.0 porting

- 16 filters were updated to take into account the modifications in VTK pipeline mechanism
- VISU Med converter and SMESH converter was updated to take into account the support of quadratic elements which is realized completely in the VTK version 5.0 and which is absent in version 4.4
- 3DWidget used in the Gauss Viewer was changed in the part concerning the event mechanism, thus the current sources should be adopted to these changes
- With the VTK version 5.0 only VTK Text properties can be used for applying text properties. Thus SALOME sources were adopted to the new API.
GUI Module

New dialog box of exiting from Salome GUI

A new dialog box appears when you exit from the Salome GUI. This dialog box contains a checkbox “Shutdown standalone servers” which allows you to kill completely SALOME. By default this checkbox is active.

Find tool

A new functionality “Find Tool” has been implemented to facilitate the search of objects if in the Object browser for the cases when we have studies with lots of published objects.

GEOM Module

Improved selection mechanism in OCC viewer:

- The right mouse button click & drag activates polyline selection (the left button is used for rectangle selection);
- The left button click during drag adds the current point to the polyline;
- The middle button click or Backspace during drag removes the previously added point from the polyline
- The left button double-click or a single-click on the first point of the polyline closes the polyline and performs the selection
Angle to define Chamfer

New parameters: chamfer size along a face and the angle between the face and the chamfer plane define the chamfer.

Normal to a Face

A new "Normal to a Face" functionality permits to obtain a normal to a face at a given point. If the point is not located on the face, it is projected on it. If the face is planar, the input of the point is optional. If the point is not given for a non-planar face, the normal will be computed in its mass center.
**Fillet with a variable radius.**

It is possible to create a fillet with a variable radius, defined separately for both ends of the filleted edge.

**Free selection of edges for Chamfer**

It is now possible to select the edges for chamfer directly in the viewer.

---

**New features in Face Creation**

The algorithm does not necessarily need a closed wire to build a face and it is possible to authorize creation of a face directly from edges which form a closed contour. A wire of the maximum length is automatically built from all given edges and wires.

If the algorithm finds multiple closed wires, it can build a face with holes or some separate faces, depending on the mutual placement of wires. If some resulting wires remain open, they will be added to the resulting compound as is.

The list of input shapes can include not only wires and edges, but also lines, arcs, circles, ellipse and curves.
Creation of a Circle by three Points

A new algorithm for circle construction permits to build a circle by three points, where the Center Point is the center of the circle, the distance between the Center point and Point 1 is the radius of the circle and Point 2 is used to define the plane where the circle lies.

Get shapes on shape

This new operation produces a group of sub-shapes of the exploded shape, which are located IN, OUT or ON the reference shape. In the images below the box is the exploded shape and the cylinder is the reference shape.

The edge IN the reference cylinder is highlighted in red.

The edges OUT of the reference cylinder are highlighted in red

Set presentation parameters and sub-shapes from arguments

This new feature has been introduced into partition, transformation and Boolean operations. It allows preserving the sub-shapes and colors of parent shapes in the resulting shape of operation.
In the pictures there are two objects: a box with two published faces and a cylinder with published free edges – and the results of their Common (to the left) and Cut (to the right). It can be seen that these objects inherit colors and sub-shapes from their arguments.

Auto color

"Auto color" action has been added for Shape object in GEOM and for Mesh object in SMESH modules (additionally, a new push button for choosing color is provided in Create/Edit Group dialog in SMESH module).
SMESH Module

Point and Line creation by intersection

New algorithms for Point and Line creation have been introduced. A Point can be created by intersection of two lines and a line can be created by intersection of two planes.

Extrusion and revolution in both directions

A new option "Both Directions" in Extrusion and Revolution dialogs which permits to extrude or revolve an object both in the direction of the vector of extrusion or revolution and in the opposite one.
Two new meshers have been implemented into Salome:

2D mesher **BLSURF** and

3D mesher **TetMesh-GHS3D**

This object was meshed with **BLSURF** mesher.

The method implemented in the BLSURF software is suitable for generating a mesh which conforms to given constraints (prescribed sizes of the elements in the vicinity of points of the surfaces) and closely approximates the geometry of the surface. It consists of meshing a 2D parametric domain, and a surface mesh is obtained when this is mapped to the 3D space. By digitization of the interface curves which represent the common boundary of the patches, it has been adapted to surfaces made up of several parametric patches.

The TetMesh-GHS3D is an automatic tetrahedral mesh generator based on the VORONOI-DELAUNAY method. The software is capable of dealing with arbitrary three-dimensional domains and of generating meshes that are adapted to the various problems of scientific computing, e.g. Computational Fluid Dynamics, Computational Structural Mechanics, Computational Electromagnetics, thermal problems, etc.

This object was meshed with BLSURF mesh on the surface and TetMesh-GHS3D in the interior.
Building 3D mesh from the imported 2D skin mesh

3D Hypothesis and Algorithms can be assigned to the imported 2D skin mesh.

The following Hypothesis and Algorithms are available:

1. "Tetrahedron (Netgen)" + "Max. Element Volume"

2. "Tetrahedron (GHS3D)"

After the computation you can see the 3D mesh in “Mesh Infos” dialog box and in the 3D Viewer.

Interpolated coloring of quadratic mesh cells during post-processing.

This new feature allows seeing variations of isovalues inside quadratic mesh cells.

The pictures show the interpolated coloring of a simple 4*4 mesh.
NETGEN meshing algorithm compatible with rectangular mesh cells

NETGEN meshing algorithm is now compatible with rectangular mesh cells due to a new special adaptor, which creates pyramids near the quadrangles in the mesh and creates a list of triangle faces (pyramid faces) for the corresponding quadrangle. This adaptor returns the list of created triangles for each quadrangle.

In the picture you can see a partitioned box with a tetrahedral 3D mesh transforming into hexahedral 3D mesh.

Table 3D Presentation

It has become possible to create a graphic presentation of a table. The relief and coloring of the resulting presentation both correspond to the values contained in the table. The values from the first column of the table define X-axis. The values from the second to the last columns define Z-axis. Y increases with a fixed step starting from Y=0. To successfully produce a Table 3D presentation, the table should not contain empty fields.
"Angle by step" or "Total angle" angle definition

It has become possible to select between "Angle by step" and "Total Angle" angle definition. It is also possible to preview the results of "Revolution around axis" operation.

Angle by Step - the elements are extruded by the specified angle at each step (i.e. for Angle=30 and Number of Steps=2, the elements will be extruded by 30 degrees twice for a total of 30*2=60)

Total Angle - the elements are extruded by the specified angle only once and the number of steps defines the number of iterations (i.e. for Angle=30 and Number of Steps=2, the elements will be extruded by 30/2=15 degrees twice for a total of 30).
VISU Module

New sprite presentation in VISU module

The "sprites" can be seen for every "colored" VISU presentation when the "Points" presentation mode is chosen.

The "sprites" use the same parameters for the rendering as it is defined by default for "Gauss Points".

The magnification parameter of the "sprites" can be changed individually for each presentation of the corresponding VISU object in a VTK view by typing "M" and "m" keys from keyboard, which corresponds to the increasing and decreasing the magnification parameter. It is important, that the corresponding VTK viewer, where the presentations are displayed, should be in input focus.

The "sprites" magnification parameter belongs to VTK representations, not to the corresponding VISU object. As a consequence, if the same VISU object will be displayed into another VTK viewer, the "sprites" magnification parameter will be initialized with some default value.

The "sprites" magnification parameter is the only parameter which can be modified from the whole set of available parameters (see the description in the previous comment). If there will be a necessity to manage other "sprites" parameters, it should be requested by a separate improvement.

Gauss Points presentation on Groups.

Now the Gauss Points presentation can build on Groups.

The interface of the GUI dialog box is now similar to Scalar Map on Groups.

Note:

Accordingly, the presentation of Gauss Points on Groups must be built on groups, which consist of cells.
Scalar Bar Properties dialog has

The newly implemented Scalar Bar Properties dialog allows changing the relative scale of Title, Labels and Color Bar.

It is possible to change the visibility of units in Scalar Bar title and the format of labels.
Other improvements

- Create Group operation in the Mesh module gives the default name "Group on (Name of the geometry)" to the group created on a geometrical object.

- New "Mesh Element Info" functionality allows seeing the information about the selected mesh element, including the element ID, geometrical type (node, edge, face, volume) and gravity center.

- Mesh Translation, Rotation and Symmetry (Mirror) operations create a new object \([\text{meshname}]_{\text{translated}}\) (or _rotated, or _mirrored) as a result of the operation in the object browser. This object can’t be meshed with a new mesh (because the underlying geometry was not modified), but it can be selected and subject to other mesh modification operations.

- Mesh Translation, Rotation and Symmetry (Mirror) operations now allow copying Groups of elements (nodes, edges or wires) to the newly created mesh, resulting from a Modification operation.

- Any edge in the presentation can be selected to be a vector for construction of a new object.

- "Select Sub-Shapes" option in Create group dialog is active by default.

- "Delete" context menu item has been added in the Object Browser. This operation permits to entirely remove the selected object(s) from the study. When this operation is selected, the user receives a confirmation message with a list of objects to be deleted.

- MEDMEM and Gibi driver are now able to convert CASTEM fields with Gauss Point values to the MED format. These fields can be displayed in the Gauss Viewer in VISU module.

- A new field interlace MED_EN::MED_NO_INTERLACE_BY_TYPE allows faster reading "NoInterlace" fields from a MED file and writing them into it. Its specific methods working only with "NoInterlaceByType" fields have "ByType" postfix. There is also a Python interface to create and work with fields with gauss points.

- Proper handling of meshes with families but without groups has been implemented in M EDMEM and MEDSPLITTER. In addition, empty families for parallel codes can be created on a partition that does not own any element from the original family.

- Incorrect hypotheses definition does not prevent mesh computation. Errors in the definition of hypotheses are shown in the same dialogue as computation errors.

- The functionality Main Menu - Repair - Remove extra edges is now available from GUI.

- "Create common groups for initial meshes" checkbox in "Create Compound" allows automatically creating groups of all elements (nodes, edges, faces and volumes) of initial meshes in the resulting mesh compound.

- MEFISTO meshing algorithm has been modified to be able to work without any hypothesis assigned, in the same way as "Quadrangle (Mapping)", "NETGEN*D" and "Hexahedron (i,j,k) algorithms.

- The button "OK" in the interface of many dialogs has been renamed into "Apply & Close".

- "Linear variation of the angles" option has been added to "Extrusion along a path" functionality. It allows defining the angle of gradual rotation for the whole path. At each step the shape will be rotated by angle/number of steps.

- In NETGEN, the upper limit of "Number of segments per edge" and "Number of segments per radius" parameters has been increased to 1e+6.
Submenu "Most recently used" has been added in the menu "File". It shows the list of the last opened and saved studies.

It has become possible to get the curvature radius of a curve in a certain point. If the point is defined by its coordinates, use command CurveCurvatureByPoint(). If the point is defined by a parameter indicating its position on the curve, ranging from 0.0 to 1.0, use command CurveCurvatureByParam().

It has become possible to get minimum and maximum curvature radius of a surface in a certain point. If the point is defined by its coordinates, use commands MaxSurfaceCurvatureByPoint() or MinSurfaceCurvatureByPoint(). If the point is defined by parameters U and V, ranging from 0.0 to 1.0, indicating its position on the surface, use commands MaxSurfaceCurvatureByParam() or MinSurfaceCurvatureByParam().

It has become possible to create a point by a surface and parameters U and V, ranging from 0.0 to 1.0, indicating the position of the point on the surface. For example, (0.5; 0.5) means that the point is located in the middle of the face. This can be done from GUI or using TUI command MakeVertexOnSurface().

Different colors have been implemented for display of already created objects (now shown in Red) and for preview of objects during creation (now shown in violet).

It has become possible to select a compound containing faces as argument in the "Shell Construction" dialog box.

The options --with-ihm and --without-ihm in build configure have been renamed into -with-gui and --without-gui in modules MED, GEOM and SMESH.

Scaling functionality has been implemented in OCCViewer. It allows deforming objects along the coordinate axes.

It has become possible to define the distance at which the object is translated in the direction of a vector. Earlier that distance was the length of the vector.

A new function "Get in Place" has been implemented in Create Group dialog.

It has become possible to select a Compound as the main shape and an object that belongs to the Compound as the Secondary Object. The objects that belong both to the Compound and the Shape will be added to the group result.

QActionGroup* createActionGroup() method has been added to SalomePyQt python interface.

The result of GetInPlace() feature is now used to limit graphical selection during group creation in Geometry module.

Persistence of colors on groups has been implemented. When some groups are saved into MED file in SMESH module, their colors are also saved. So, when the saved MED file is imported the groups have the same color attributes.

Now information that can be obtained with geompy.KindOfShape method is accessible via "Whats Informations" dialog in Geometry GUI.

The possibility to quit Salome with automatic shutdown of standalone servers has been implemented in batch mode (without GUI). For this you can use the following command:

import salome salome.salome_init() salome.lcc.shutdownServers() salome.SALOME_LifeCycleCORBA_kilOmiNames()
The monochromatic representation of level lines on Iso Surfaces has been implemented. To show level lines in one color, uncheck Magnitude coloring checkbox. By default the lines are black, but you can choose any appropriate color by clicking Select Color button. It is also possible to show labels with value on each surface for producing topographic "level lines". This presentation can be superimposed on any other field presentation, for example, Scalar Map.

- **SALOME configuration XML file** `config_appli.xml` is automatically generated in `${KERNEL_ROOT_DIR}/bin/salome` folder during the installation.

- After the end of installation this file contains data about available SALOME modules and their configuration.

- The file names for pictures, saved during animation now begin with a number, which allows sorting pictures by order in the browser.

- The following dialogs show the content of the current directory (the directory of Salome launching) by default:
  - all modules: File - Load script...
  - Geometry module: File - Import... and File - Export...
  - Mesh module: File - Import - ... and File - Export - ...
  - Post-Pro module: File - Import - ...

  Note that these dialogs show the current directory only for the first time. So, if the user selects some a path and clicks "Open" (or "Save" in case of export), for the next time dialogs will show the previously selected path.

- The possibility to translate "Gauss Points", just as other VISU presentations has been implemented.

- It has become possible to see MED file info from VISU and SMESH modules.

- "Current Time Stamp" combo box has been removed from "Scalar Map on Deformed Shape" dialog.

- Documentation for python interfaces `geompy` and `smesh` has been rearranged.
## Supported Linux distributions and pre-requisites

**Supported platforms**

**SALOME 4.1.4sp2** supports Debian 3.1 Sarge, Debian 4.0 Etch 32bit and 64bit, Mandriva 2006 32bit and 64bit, Mandriva 2008 32bit and 64bit.

**SALOME 4.1.4sp2** version has been mainly tested with the below listed pre-requisites on Mandriva 2008 32bit and Debian 4.0 Etch 64bit platforms.

**SALOME 4.1.4sp2** comes with the same prerequisites versions on all supported platforms. The table below lists the versions of the products used by SALOME platform. Other versions of the products can also work but it is not guaranteed.

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
<th>GUI (IAPP)</th>
<th>KERNEL</th>
<th>GEOM</th>
<th>SMESH</th>
<th>VISU</th>
<th>MED</th>
<th>SUPERVISION</th>
<th>Hexotic-PLUGIN</th>
<th>FILTER</th>
<th>GHS3D-PLUGIN</th>
<th>RANDOMIZER</th>
<th>SIERPINSKY</th>
<th>BLsurf-PLUGIN</th>
<th>GHS3D-PLUGIN</th>
<th>PYCALCULATOR</th>
<th>COMPONENT</th>
<th>NETGEN-PLUGIN</th>
<th>LIGHT</th>
<th>MULTIPR</th>
<th>CALCULATOR</th>
<th>HELLO</th>
<th>PYHELLO</th>
<th>YACS</th>
<th>xdata</th>
<th>xSALOME</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc*</td>
<td>3.3.5**</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>automake*</td>
<td>1.9**</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>autoconf*</td>
<td>2.59**</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>libtool*</td>
<td>1.5.6**</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GNU make*</td>
<td>3.80**</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Python</td>
<td>2.4.4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Qt&amp;msg2qm</td>
<td>3.3.8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sip</td>
<td>4.7.3</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PyQt</td>
<td>3.17.4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boost</td>
<td>1.34.1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Swig</td>
<td>1.3.31</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Open CASCADE Technology</td>
<td>6.3***</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Qtqt</td>
<td>4.2.0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OmnIORB</td>
<td>4.1.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>omnINotify</td>
<td>3.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Hdf5</td>
<td>1.6.4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Med</td>
<td>2.3.5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vtk</td>
<td>5.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Numeric</td>
<td>24.2</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphviz</td>
<td>2.16.1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Doxygen</td>
<td>1.5.6</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>NETGEN</td>
<td>4.5</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>docutils</td>
<td>0.3.9</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>metis</td>
<td>4.0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scotch</td>
<td>4.0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>libxml2</td>
<td>2.6.27</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elementtree</td>
<td>1.2.6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cElementTree</td>
<td>1.0.5</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blsurf****</td>
<td>2.7.12</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TetMesh****</td>
<td>4.0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TclTk</td>
<td>8.4.14</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*) Not included into SALOME Installation procedure.

**) Minimal required version.

****) Service pack 1 for OCCT 6.3 fixing some bugs can be also used optionally.

*****) Commercial software, not distributed with the SALOME platform.
**Important note:** For some platforms Salome uses prerequisites with patches like in RPM and defines specific keys. So if you compile products without Install Wizard we strongly recommend you to check compilation keys using shell files located in `config_files` folder of the Installation Procedure.

**Important note:** BLSurf, Hexotic, GHS3D and GHS3DPRL (parallel) mesher plug-ins require commercial license for the corresponding Distene meshers to be used within the SALOME platform. Otherwise, the runtime error messages will be raised at the application execution.

SALOME 4.1.4sp2 depends of a number of products for run time execution, others are necessary only for compilation or generation of development documentation (like doxygen for example). Below there is a list of mandatory and optional products.

## Software Requirements

<table>
<thead>
<tr>
<th>Software</th>
<th>Compilation and Development</th>
<th>Execution</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mandatory</td>
<td>Optional</td>
<td>Mandatory</td>
</tr>
<tr>
<td>gcc</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Automake</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autoconf</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>libtool</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNU make</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tcltk</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Qt</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>msg2qm</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PyQt</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Boost</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Swig</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OpenCASCADE Technology</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Owt</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>OmniORB</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hdf</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Med</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Vtk</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphviz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Doxygen</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NETGEN</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>docutils</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cppunit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mpi</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>openpbs</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Lsf</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>metis</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>scotch</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>libxml2</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>elementtree</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cElementTree</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>blsurf</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TetMesh-GHS3D</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
How to install and build SALOME

Please follow README file from Installation Wizard for processing correctly installation of SALOME and all prerequisites.

If you would like to compile SALOME from scratch, please use `build.csh` or `build.sh` script delivered with the Installation Wizard. Call "`build.csh -h`" to see available options of this script.

### SALOME System Requirements

<table>
<thead>
<tr>
<th></th>
<th>Minimal Configuration</th>
<th>Optimal Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Pentium IV</td>
<td>Dual Core</td>
</tr>
<tr>
<td>RAM</td>
<td>512 Mb</td>
<td>2 Gb</td>
</tr>
<tr>
<td>Hard Drive Space</td>
<td>1.5 Gb</td>
<td>5 Gb</td>
</tr>
<tr>
<td>Video card</td>
<td>64 Mb</td>
<td>128 Mb</td>
</tr>
<tr>
<td>CD/DVD</td>
<td>optional</td>
<td>optional</td>
</tr>
</tbody>
</table>

### How to get the version and pre-requisites

**SALOME 4.1.4sp2** pre-compiled binaries for Linux Debian 3.1 Sarge, Mandriva 2006 32bit and 64bit, Mandriva 2008 32bit and 64bit, Debian 4.0 Etch 32bit and 64bit can be retrieved from the [www.salome-platform.org/download](http://www.salome-platform.org/download) page.

Besides it is possible to install Debian 3.1 Sarge Salome universal binaries on the platforms that are not supported officially.

The SALOME Installation procedure includes SALOME modules sources, and it is possible to build sources from scratch using `build.sh` or `build.sch` script coming with installation procedure.

There are two patches on NETGEN which are placed inside NETGENPLUGIN modules sources. The first patch file is used for all 32 bit platforms, the second patch file is an addition to the first one and should be applied only for 64bit platforms.

During the compilation on NETGEN from sources by SALOME Installation Wizard, the patches are applied automatically to the standard NETGEN distribution. You can download NETGEN 4.5 from its official site using the following link: [http://www.hpfem.jku.at/netgen](http://www.hpfem.jku.at/netgen).

All other pre-requisites can be obtained either from your Linux distribution (please be sure to use a compatible version) or from the distributors of these pre-requisites ([www.trolltech.com](http://www.trolltech.com) for QT for example).
Known problems and limitations

- Create Group operation in the Mesh module gives the default name "Group on (Name of the geometry)" to the group created on a geometrical object.

- The following limitations refer to BLSURF plugin:
  - Mesh contains inverted elements, if it is based on a shape, consisting of more than one face (box, cone, torus...) and if the option "Allow Quadrangles (Test)" has been checked before computation.
  - SIGFPE exception is raised after trying to compute a mesh based on a box with "Patch independent" option checked.
  - It has been found out that BLSURF algorithm can’t be used as a local algorithm (on sub-meshes) and as a provider of low-level mesh for some 3D algorithms because BLSURF mesher (and, consequently, the plugin) does not provide information on node parameters on edges (U) and faces (U, V). For example, the following combinations are impossible:
    - global MEFISTO or Quadrangle(mapping) + local BLSURF;
    - BLSURF + Projection 2D from faces meshed by BLSURF;
    - local BLSURF + Extrusion 3D;

- Sometimes regression test bases give unstable results; in this case the testing should be restarted.

- A native VTK can be used only after manual recompilation with the GL2PS component.

- NETGEN 1D-2D and 1D-2D-3D algorithm do not require definition of 2D and 1D algorithms and hypotheses for both mesh and sub-mesh. 2D and 1D algorithms and hypotheses defined with NETGEN 1D-2D or 1D-2D-3D algorithm will be ignored during calculation.

- SALOME supports reading of documents from earlier versions but the documents created in the new version may not open in earlier ones.

- If SALOME modules are not installed in a single folder, SALOME may not work in the CSH shell since the environment variables are too long by default. In this case, it is suggested to use SH or to install all modules in the same folder.

- During the compilation of OCT 6.x by makefiles on a station with NVIDIA video card you can experience problems because the installation procedure of NVIDIA video driver removes library libGL.so included in package libMesaGL from directory /usr/X11R6/lib and places this library libGL.so in directory /usr/lib. However, libtool expects to find the library in directory /usr/X11R6/lib, which causes compilation failure (See /usr/X11R6/lib/libGLU.la). We suggest making symbolic links in that case: "ln -s /usr/lib/libGL.so /usr/X11R6/lib/libGL.so ln -s /usr/lib/libGLU.la /usr/X11R6/lib/libGLU.la"

- VISU module does not support timestamps defined on the same field but on different meshes.

- Stream lines presentation can not be built on some MED fields due to limitations in VTK.

- In the current implementation of "Save VISU State" operation the parameters of Gauss view Partition mode are not stored. If a window has been partitioned and saved, it will be restored as non-partitioned.

- Step-by-step execution fails on some graphs in SUPERVISOR. This functionality is only a prototype and has not been finished completely.

- MEFISTO algorithm sometimes produces different results on different platforms.

- In some cases the number of triangles generated by MEFISTO may be different at each attempt of building the mesh.

- Results of some TUI non-regression testing are different on slow and fast computers. This is caused by using the functionality of GEOM from Supervisor in parallel nodes in some test cases. Open CASCADE Technology (OCCT) is not thread safe in some geometry operations, so on some hardware configurations the conflict with parallel access to some data may occur and such supervisor graphs may fail. This problem will be fixed in the future. Currently the workaround uses the GEOM nodes subsequently.