Organisation de la plate-forme P@L CEA-EDF

Historique

Date	*Version*	*Description*	*Author*
25 février 2003	0.1	Organisation des sources	J. Roy
13 mars 2003	0.2	Organisation de l'environnement de développement	J. Roy
27 mars 2003	0.3	Organisation de l'environnement de développement	J. Roy
16 avril 2003	0.4	Organisation de l'environnement de développement	J. Roy

• *Introduction*

L'objectif de ce document est de définir l'organisation et la gestion de configuration de la plate-forme PAL CEA-EDF. Celle-ci se base essentiellement sur ce qui a été défini pour la plate-forme *Salome* [1], mais doit s'adapter aux nouvelles exigences apportées par le futur développement de la plate-forme *Salome2*.

En effet, si l'organisation de la plate-forme *Salome* autorisait la possibilité de gérer de manière globale l'ensemble des sources, il n'en sera pas de même dans le cadre de *Salome2* dans lequel 21 participants collaboreront à son développement. Il est clair que chacun de ces collaborateurs participera uniquement au développement d'une partie restreinte du projet en fonction de ses compétences et de ses pôles d'intérêt. Les autres modules seront vu uniquement comme des pré-requis à leur développement.

Cette analyse nous a donc conduit à vouloir organiser les sources de la plate-forme PAL sous forme de modules gérés séparément. La gestion de configuration sera faite par CVS comme précédemment, mais nous proposons de définir une base CVS par module : une pour le noyau *Salome*, IAPP et l'étude, et ensuite une pour chacun des autres modules de base : MED, SMESH, GEOM, VISU, SUPERV et DATA.

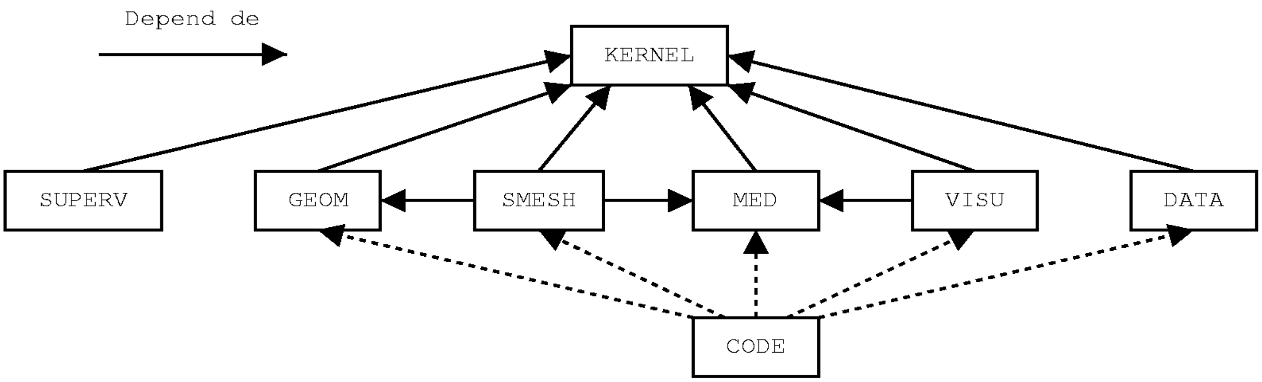
- ***Organisation en modules distincts***

Les différents modules de la plate-forme PAL sont les suivants :

Noyau+Etude+IAPP	KERNEL
Géométrie	GEOM
Format d'échange de données	MED
Maillage	SMESH
Post-traitement	VISU
Supervision	SUPERV
SALOME	SALOME

Le module SALOME est un module global ne contenant pas de code source à proprement parler, mais uniquement les fichiers de configuration servant à l'installation des modules de base. Ces fichiers sont regroupés ici, car ils sont communs à l'ensemble des modules. Cela permet ainsi d'éviter la duplication des scripts de configuration. On prévoit également de conserver la possibilité d'installer l'ensemble de *Salome* par une procédure qui appellera successivement les procédures individuelles.

Les modules sont liés entre eux par leurs dépendances intrinsèques :



– <u>Conséquences pour les utilisateurs finaux de *Salome* :</u>

Pour les utilisateurs d'un module, l'extraction des sources du module depuis l'archive *src.tar.gz* et leur compilation se font généralement dans le même répertoire (notons qu'il est également possible de récupérer des archives binaires *bin.tar.gz* contenant un ou plusieurs modules déjà compilés), puis l'installation se fait par défaut dans le répertoire **/usr/local/**, ou au choix de l'installateur via l'option *--prefix* du script *configure*. Les commandes associées sont :

./configure
make
make install

La procédure d'installation (plus précisément le *./configure*) d'un module particulier est chargée de vérifier la présence des autres modules dont il dépend. Le premier module installé est donc toujours KERNEL.

Il est aussi possible de réaliser une procédure d'installation complète d'une partie des modules ou de tous les modules, qui permette d'installer par les trois mêmes commandes que ci-dessus l'ensemble des modules dans le même répertoire (les sources seront dans ce cas précis regroupées dans un répertoire **SALOME_SRC/**.

– Conséquences pour les développeurs de *Salome* :

Pour les développeurs, les choses sont différentes. Un développeur intervenant sur un sous-ensemble de modules donné récupère les sources de ces modules dans leurs bases CVS respectives, les compile et les modifie localement éventuellement, de façon itérative. Chacun des autres modules de *Salome* extérieurs à ce sous-ensemble rentre dans l'une des catégories suivantes :

- Soit il n'est pas indispensable à l'utilisation des modules sur lesquels intervient le développeur, auquel cas il n'est pas forcément compilé et installé. Les dépendances entre modules sont données par la figure ci-dessus.
- Soit il est indispensable et doit être installé. Cette installation peut se faire par ces deux manières :
    - soit le module est vu comme un pré-requis, et est installé à la manière “utilisateur” à partir de l'archive *tar.gz* comme décrit ci-dessus.
    - soit le module est extrait de sa base CVS, compilé et installé.

Il est possible de créer des scripts *build_configure*, *configure* et *Makefile* dans le module SALOME qui permettent d'automatiser la procédure si le développeur souhaite compiler et/ou installer plusieurs modules en même temps à partir des sources extraites des bases CVS.

Chacun des modules possède un répertoire **<MODULE>_SRC/** archivé dans sa base CVS. Si le développeur décide de récupérer les sources d'un module, celles-ci sont rangées dans un répertoire **<MODULE>_SRC/** local. Il appelle ensuite *build_configure*, *configure* (depuis le répertoire de compilation) et *make* pour compiler ces sources dans **/mon/repertoire/de/compilation/pour/MON_module/**. Il appelle enfin *make install* (facultatif) pour installer le module dans **/mon/repertoire/d/installation/pour/MON_module/**. Trois répertoires sont donc nécessaires pour chaque module installé séparément (par exemple pour KERNEL **$HOME/KERNEL_SRC/**, **/.../KERNEL_BUILD/** et **/.../Products/install/Salome-kernel-1.0/**). Les différents répertoires sont décrits de manière plus détaillée ci-après.

Chaque module utilise les outils *autoconf* et *libtool*, comme c'est déjà le cas dans l'organisation actuelle de *Salome*. Le changement d'organisation peut éventuellement être l'occasion d'introduire en outre l'outil *automake*.

• *Organisation des sources*

– Organisation des bases CVS :

Chacun des modules de la plate-forme PAL possède sa propre base CVS. Le module CVSROOT (module de configuration de CVS) étant spécifique à chaque base, cela permet de gérer de manière différente les accès en lecture et en écriture aux différents modules PAL.

L'arborescence de chacune des bases (hormis SALOME et KERNEL) suit le modèle suivant :

```
<MODULE>_SRC/
    adm_local/
        unix/
    bin/
    doc/
    examples/
    idl/
    resources/
    src/
    tests/
    README
    build_configure
    configure.in.base
    Makefile.in
```

Il n'y a pas de répertoire **adm/** (répertoire contenant les fichiers d'administration) dans chacun des modules, mais un unique répertoire **salome_adm/** dans le répertoire d'installation de KERNEL, qui est commun à tous les modules. Le répertoire **salome_adm/** est contenu dans **KERNEL_SRC/**, puis recopié dans **/mon/repertoire/d/installation/pour/KERNEL/** lors de l'installation du module KERNEL. Ceci évite la duplication dans les bases CVS et les copies de travail d'un certain nombre de fichiers communs à tous les modules.
Le répertoire **adm_local/** sert aux fichiers d'administration spécifiques à un module particulier. Si un fichier du répertoire **adm_local/** est utilisé par plusieurs modules, alors il sera remonté dans le répertoire **salome_adm/** de la base SALOME.
La configuration d'un module autre que KERNEL a donc besoin de savoir où se trouve le module KERNEL pour accéder au répertoire **salome_adm/**. Il est donc obligatoire de définir une variable d'environnement KERNEL_INSTALL.

L'arborescence de la base du module KERNEL est la suivante :

```
KERNEL_SRC/
    adm_local/
        unix/
    bin/
    doc/
    examples/
    idl/
    resources/
    salome_adm/
        unix/
            make_begin.in
            make_conclude.in
            <other_files>.in
            config_files/
                check*.m4
                <other_files>.m4

    src/
    tests/
    README
    build_configure
    configure.in.base
    Makefile.in
```

L'arborescence de la base du module SALOME est la suivante (le module SALOME ne contient que des fichiers de configuration, compilation et installation globales) :

```
SALOME_SRC/
    build_configure
    configure.in
    Makefile.in
```

– Organisation des copies de travail :

Si le développeur récupère un unique module dans la base CVS, il obtiendra localement un répertoire **<MODULE>_SRC/** contenant les sources identique à celui de la base CVS. Il n'a dans ce cas pas besoin de récupérer le module SALOME.

Si le développeur souhaite récupérer plusieurs modules distincts sur lesquels il interviendra simultanément, il devra d'abord récupérer les sources du module SALOME dans son répertoire de développement **SALOME_SRC/**, puis ensuite récupérer les sources des modules qu'il voudra développer dans ce même répertoire de développement.
L'arborescence de développement est la suivante :

```
SALOME_SRC/
        build_configure
        configure.in
        Makefile.in
        <MODULE>_SRC/
                adm_local/
                        unix/
                bin/
                doc/
                examples/
                idl/
                resources/
                salome_adm/                 (uniquement pour le module KERNEL !)
                        unix/
                src/
                tests/
                README
                build_configure
                configure.in.base
                Makefile.in
```

– Organisation des archives *src.tar.gz* :

Le choix retenu pour l'organisation des copies de travail issues des bases CVS implique une arborescence identique, pour chaque module, à son archive *src.tar.gz*. Les archives ne contiennent pas de répertoire **salome_adm/**, pour lequel elles dépendent du module KERNEL. Elles sont en outre autonomes, ie. une archive donnée contient un et un unique module.
Le script *build_configure* sera fourni avec les archives *src.tar.gz*.
Les deux modes d'acquisition des sources sont donc équivalents, mais il est bien sûr préférable d'utiliser les bases CVS pour les modules sur lesquels le développeur doit intervenir.

- ***Organisation des sources compilées***

Une fois les sources récupérées (à partir d'une base CVS ou par extraction d'une archive *src.tar.gz*), une règle de bon usage pour un développeur consiste à compiler ces sources dans un répertoire distinct (non forcément utile pour un simple utilisateur).
Nous allons distinguer le cas de la compilation d'un module seul et celui de la compilation de plusieurs modules en même temps. Dans les deux cas, l'arborescence du répertoire de compilation est sensiblement différente de celle du répertoire des sources.

– Cas d'un unique module :

L'arborescence du répertoire de compilation de **<MODULE>_SRC/** est la suivante :

```
/mon/repertoire/de/compilation/pour/MON_module/
        adm_local/
                unix/
        bin/
                salome/
        idl/
                salome/
        include/
                salome/
        lib/
                salome/
        salome_adm/
                unix/
        share/
                salome/
                        doc/
                        resources/
        src/
        Makefile
```

Cette arborescence est générée par le *configure* du module considéré. Le répertoire **salome_adm/** est créé à partir du répertoire **/mon/repertoire/d/installation/pour/KERNEL/**, en respectant la même sous-arborescence (les fichiers *.in gardent le même nom mais sans l'extension '.in' après traitement par le *configure*).
En outre, le *configure* teste la présence des prérequis ainsi que l'installation des modules dont dépend le module considéré.

Organisation du catalogue de modules :
L'organisation du catalogue de modules pourrait vraisemblablement prendre exemple sur celle mise en place pour le projet *Alliances* par la société OpenCascade (Action Y. Fricaud OpenCascade à mener). Nous proposons néanmoins une solution propre au PAL pour l'instant :

le sous-répertoire **share/salome/resources/** contient un fichier « SALOME_<MODULE>_CATA.xml » correspondant à la partie du catalogue de modules relative à <MODULE>. Ces différents fichiers sont concaténés par le script *runSalome*, afin de former le catalogue des modules dans son intégralité. L'ordre de recherche de ces catalogues partiels est le suivant :

1/ le script se réfère tout d'abord à un MODULES_PATH défini par l'utilisateur contenant une liste des répertoires d'installation des divers modules. La liste des modules présents dans une adresse donnée du MODULES_PATH est déduite automatiquement, par recherche d'une ou plusieurs sous-arborescence(s) **share/salome/resources/** non vide(s). La recherche parmi les adresses du MODULES_PATH suit la règle habituelle des PATH, ie. débute à gauche de la liste, et s'arrête dès que l'élément recherché est trouvé. Si plusieurs versions d'installation d'un même module sont trouvées pour une même adresse du MODULES_PATH, c'est alors l'ordre alphabétique qui donne l'ordre de priorité.

2/ en second lieu, c'est la variable <MODULE>_APPLI_INSTALL indiquant le chemin d'installation de <MODULE> (pour une application basée sur *Salome*) qui est prise en compte si elle a été définie par l'utilisateur.

3/ en dernier lieu, le chemin d'installation contenu dans la variable <MODULE>_INSTALL est pris en compte pour trouver le fichier « SALOME_<MODULE>_CATA.xml ».

– Cas de plusieurs modules :

Si nous compilons plusieurs modules en même temps, les répertoires de compilation respectifs de tous les modules appartiennent au même répertoire :

```
/mon/repertoire/de/compilation/pour/MES_modules/
    Makefile
    <MODULE>_BUILD/
        adm_local/
            unix/
        bin/
            salome/
        idl/
            salome/
        include/
            salome/
        lib/
            salome/
        salome_adm/
            unix/
        share/
            salome/
                doc/
                resources/
        src/
        Makefile
```

Cette arborescence est créée par le *configure* global du répertoire **SALOME_SRC/**, qui appelle successivement tous les *configure* des différents modules, qui eux créent les sous-répertoires de **<MODULE>_BUILD/**.

Elle reprend celle du répertoire d'installation qui est décrite plus loin, ce qui permet au développeur de ne pas forcément lancer *make install* après la compilation (ceci est également vrai dans le cas de la compilation d'un seul module).
Chacun des *configure* teste la présence des modules requis par les relations de dépendances, hormis les modules présents dans le répertoire **/mon/repertoire/de/compilation/pour/MES_modules/**. En effet, le *configure* d'un module donné ne doit pas tester si un autre module dont il dépend est installé si nous sommes justement en train de le configurer aussi, qu'il soit ou non déjà compilé.

• *Organisation des fichiers d'installation*

– Par *make install* :

Par défaut, l'installation est faite dans le répertoire **/usr/local/**. L'utilisateur et le développeur peuvent choisir un autre répertoire par l'intermédiaire de l'option *--prefix* du script *configure* (*configure* spécifique à un module ou global).
L'installation simultanée de plusieurs modules et l'installation d'un unique module sont similaires.

L'arborescence est la suivante :

```
/mon/repertoire/d/installation/pour/MON_OU_MES_module(s)/
        bin/
                salome/
        idl/
                salome/
        include/
                salome/
        lib/
                salome/
        share/
                salome/
                        doc/
                        resources/
```

Là où KERNEL est installé, nous avons en outre un sous-répertoire **salome_adm/**.

– A partir d'une archive *bin.tar.gz* :

Il est possible d'installer un ou plusieurs modules sans avoir à les compiler, ce qui permet un gain de temps substentiel. Il suffit pour cela de récupérer une archive binaire *bin.tar.gz* contenant les répertoires suivants :

```
bin/
        salome/
idl/
        salome/
include/
        salome/
lib/
        salome/
salome_adm/                     (si l'archive bin.tar.gz contient le module KERNEL)
        unix/
share/
        salome/
                doc/
                resources/
```

Une telle archive contient les fichiers binaires d'un ou plusieurs modules. L'utilisateur extrait cette archive dans le répertoire de son choix.

Avec ce type d'installation, aucun script *configure* n'est executé, donc aucun contrôle de dépendance sur les modules ou les prérequis n'est effectué. C'est à l'utilisateur de s'assurer de la présence de tous les éléments requis.

Remarques générales sur l'installation :

Le script de configuration d'un module recherche dans les répertoires standards du système si les modules dont il a besoin sont installés.

- Si un module est installé dans un répertoire non standard, ou s'il n'est pas installé mais seulement compilé comme c'est souvent le cas en phase de développement, alors le développeur devra positionner une variable d'environnement « <MODULE>_INSTALL » indiquant respectivement le répertoire d'installation ou de compilation. Cette variable est prioritaire sur les répertoires standards dans l'ordre de recherche des fichiers binaires du module.
  Nous définissons en fait trois variables par module indiquant les répertoires contenant les sources, les sources compilées et les fichiers d'installation. Dans un soucis de normalisation, ces variables sont nommées respectivement : <MODULE>_SRC, <MODULE>_BUILD et <MODULE>_INSTALL. Pour les applications basées sur *Salome*, ces variables deviennent <MODULE>_APPLI_SRC, <MODULE>_APPLI_BUILD et <MODULE>_APPLI_INSTALL.
- Si un module est installé dans un répertoire standard, l'existence d'un sous-répertoire **salome/** aux répertoires **bin/**, **include/**, **lib/** implique de rajouter une adresse dans les variables PATH associées. Ceci ne nous semble pas trop gênant du fait du nombre limité de modules (<~10). Ce système présente en contrepartie les avantages de pouvoir installer et désinstaller facilement un module et d'éviter tout conflit de nom.

- ***Conclusion***

Nous sommes actuellement en train de réaliser une étude de faisabilité relative à cette organisation sur la base de *Salome_Pro1.0*. Nous avons réussi à isoler le module KERNEL, c'est-à-dire à configurer, compiler, installer et faire fonctionner correctement le module KERNEL seul. Nous somme actuellement dans la seconde phase de l'étude qui consiste à faire le même travail pour le module MED, afin de faire fonctionner ensemble les modules KERNEL et MED. Une prochaine mise à jour de ce document présentera les résultats obtenus.
Une fois la gestion de configuration mise en place, les différentes bases CVS seront alimentées avec la dernière version de *Salome_Pro*.

- ***Références***

[1] SALOME Organisation et règles de production , P. Goldbronn, P. Rascles, Y. Fricaud du 13 juin 2001.