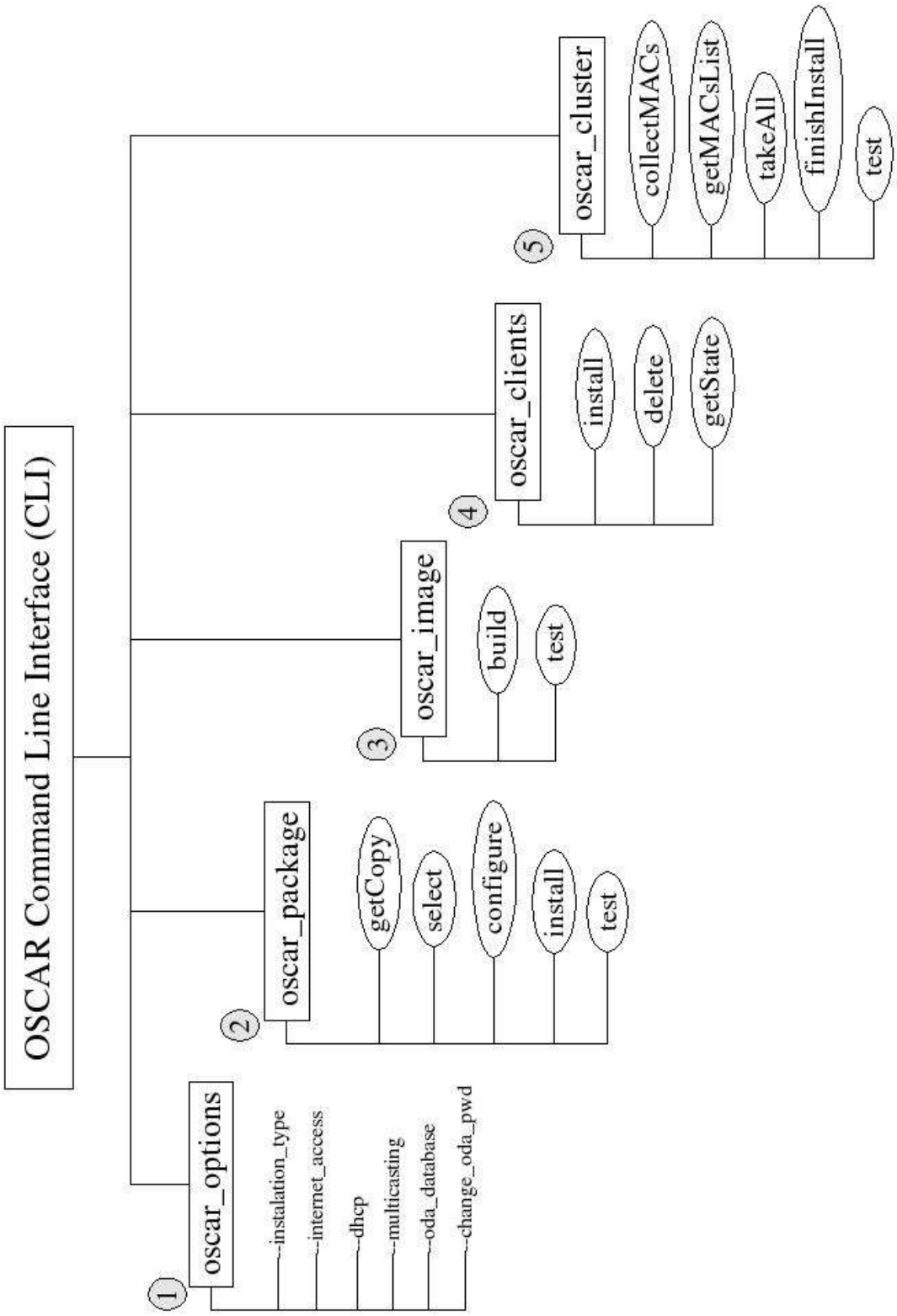
(making the modifications but mantaining part of the additional things...)

CLI (new) Structure

```
(1) oscar_options (global options)

      -> installation_type: -disk
                            -diskless (Thin-OSCAR)

      -> internet_access: yes/no

      -> dhcp: -dynamic
               -static

      -> multicasting: yes/no

      -> oda_database: -mysql
                       -postgree

      -> change_oda_pwd

      -> test


(2) oscar_package (get/install OSCAR's packages)

      -> getCopy: -copy           (OPD can do that?)
                  -download
      -> select

      -> configure

      -> install

      -> test (each package)   ----> different versions?


(3) oscar_image         (build client's image)

      -> build

      -> test


(4) oscar_clients (management of the clients)

      -> install

      -> delete

      -> getState (returns the client's state - also a client's test)


(5) oscar_cluster (setup networking )

      -> collectMACs

      -> getMACsList

      -> takeAll

      -> finishInstall

      -> test
```


OSCAR Command Line Interface (CLI)
1
oscar_options
instalation_type
internet_access
dhcp
multicasting
oda_database
change_oda_pwd
2
oscar_package
getCopy
select
configure
install
test
3
oscar_image
build
test
4
oscar_clients
install
delete
getState
5
oscar_cluster
collectMACs
getMACsList
takeAll
finishInstall
test

Installation sequence: (draw a diagram, showing the steps...)

Steps:

1) **Options:**

Before attempt to install OSCAR, there is a set of options that could be configured. This step should be optional, so there must be a 'default' configuration (when it's needed).

command: 'oscar_options [options]'

Options:

--installation_type -> defines which type of OSCAR installation will be used (that is, 'disk' or 'diskless' (Thin-OSCAR)).
Default: 'disk'.
Ex: 'oscar_options --intallation_type diskless'.

--internet_access -> enables or disables the cluster's internet access (that is, if the cluster will have or won't have "contact" with the "outside world"). This issue is particularly important when deciding if the server will be able to 'download' additional packages (from an outside repository).
Default: 'yes'
Ex: 'oscar_options --internet_acces yes'.

--dhcp -> (DHCP uses the MAC adresses to assign IP adresses to the clients)
Choses between a 'dynamic' or a 'static' DHCP. Using a 'dynamic' DHCP the server configuration will be refreshed each time a MAC is assigned to a node. A 'static' DHCP preserves the association between a MAC adress and an IP adress.
Default: 'static'
Ex: 'oscar_options --dhcp dynamic'.

--multicasting -> enables or disables multicasting.
Default: 'no'
Ex: 'oscar_options --multicasting yes'.

--oda_database -> select between MySQL and Postgree SQL.
**********Will be worked later**********

--change_oda_pwd -> modifies the OSCAR DATABASE administrator's password.
**********Will be worked later**********

--test -> performs a series of test scripts that ensures (or not) that the choosen global options are OK.
Ex: 'oscar_options --test'.

2) **Package:**

Performs the copy/download, selection, installation, configuration and testing of the OSCAR (core and additional) packages.

(a) **getCopy** -> the command 'getCopy' gets a copy of additional packages. It totally depends if the Internet is enable or disable. If it's enable, it can, through the 'OSCAR Package Downloader (OPD)', stablish a connection with a repository and download the desirable packages from there. If there isn't an Internet connection, the only way to get additional packages is copying them from a local directory.

command: 'oscar_package getCopy [options]'

Options:

--local -> specifies the local where the additional packages can be found (it can be either a local directory or a URL) and prints a list with the disponible package's info.
Ex: 'oscar_package getCopy --local /tmp/packages'
(PS: this must be the first command to be run because it defines where the additional packages are - and also, save (register) this information)

--pkname -> when the option '--pkname' is the only option (besides the '--local' option), do the copy/download of the package.
Ex: 'oscar_package getCopy --pkname clumon'

When the option '--pkname' is entered alongside other options, print the respective INFO of the package:

--info -> print the package's basic information data;
--provides -> print details;
--conflicts -> print the conflict issues (if there are any);
--requires -> print the package's prerequisites (if there are any);
--packager -> print the name and e-mail of the package's author.
Ex: 'oscar_package getCopy --local /tmp/packages --pkname clumon --info --packager'.

(b) **select** -> the command 'select' determines which packages will be further installed (by default, all packages directly included in OSCAR (the core packages) are selected and installed). The additional copyed/downloaded packages must be manually selected for installation.

command: 'oscar_package select [options]'.

Options:

--table -> gets the list (table) of the disponible packages for imeddiate selection (including package name, class and location/version).
Ex: 'oscar_package select --table'.

--pkname -> like in the command 'getCopy', if this option is entered "alone", the package is selected for installation.
Ex: 'oscar_package select --pkname clumon'.

When the option '--pkname' is entered with one or more of the following options, print the respective package's INFO (the same information provided by the same command in 'oscar_package getCopy'):

--info ->...
--provides ->...
--conflicts ->...
--requires ->...
--packager ->...
Ex: 'oscar_package select --pkname clumon --requires'.

A complementary, but important option, that helps in the automatization of the process, is:
--all -> select all packages (including the packages previously copied/downloaded)

(PS: there is no need for a a 'default' select option, because all of the core packages of OSCAR are automatically selected for installation)

(c) **configure** -> some packages in OSCAR have configuration options. The command 'configure' possibilits the configuration of those packages that can be configured.

command: 'oscar_package configure [options]'.

Options:

--table -> prints a list of packages that can be configured.

##############Still fuzy. Will be used ? ######################

(d) **install** -> performs the installation of the selected packages in the OSCAR server.

command: 'oscar_package install [option]'.

Where the only option is to choose which server adapter (interface) will be used to connect it with the cluster network:
--interface -> specifies the interface.
Ex: 'oscar_package install --interface eth1'.

(e) **test** -> performs a test with a package to assure that its installation is Ok.

command: 'oscar_package test [option]'.

Options:

--table -> print a list (table) of the installed packages.
Ex: 'oscar_package test --table'.

--pkname -> specifies which package do you want to test.
Ex: 'oscar_package test --pkname clumon'.

3) Image

Builds and tests the future client's image, that is constructed from the OSCAR core and additional (selected) packages, plus the operational system files (like RPMs).

(a) **build** -> builds the image, that will be "network copied" to them (uses the 'mksiimage' command).

command: 'oscar_image build [options]'.

It's possible to either customize the default image, by altering the following options, or simple builds the default image (by omitting the options).

Options:

--seeDefault -> print the default options.
Ex: 'oscar_image build --seeDefault'.

--imageName -> specifies the name of the current image customization, by which you will identify it (you can create several different images customizations);

--packageFile -> specifies the full adress of the text file containing a list of packages that should be installed (usually present in /opt/oscar/oscarsamples);

--packageDirectory -> specifies the directory which contains the packages that will be used to build the image;

--diskPartFile -> specifies a text file that contains the disk partition table for the image;

--ipAssignMet -> specifies the IP assignment method, that can be either 'static', 'dhcp' or 'replicant';

--postInstAct -> specifies which will be the client's post install action, that can be either 'beep', 'reboot' or 'shutdown';

Ex: 'oscar_image build --imagename oscartest1 –packageFile /opt/oscar/oscarsamples/mandrake-9.2-i386.rpmlist --packageDirectory /tftpboot/rpm --diskPartFile /opt/oscar/oscarsamples/sample.disk.ide --ipAssignMet static --postInstAct beep;.

(b) **test** -> test the integrity of a builded image.

command: 'oscar_image test [option]'.

Where the only option is to enter the image identification, that is:

--imageName -> the name of the image, used to identify it.
Ex: 'oscar_image test --imageName oscartest1'.

4) Clients

Takes care of the management of the clients (include, remove, test).

(a) **install** -> defines what clients will going to be installed. It can be used in a first moment to stablish the main set of nodes and can be used later to add new clients to it.

command: 'oscar_clients install [options]'.

Options:

--seeDefault -> print the default values**** (see: 'numberOfHosts');

--imageName -> specifies which image (previously builded) will be used.

--domainName -> specifies the client's IP domain name;

--baseName -> specifies the first part of the client's name and hostname (this name can not contain an uderscore character);

--numberOfHosts -> specifies the number of hosts. This field **DOES NOT HAVE** a default value (or better, has an invalid one), and must be obrigatory provided. The value **MUST BE** greater than '0'(zero).

--startNumber -> specifies the number of the first node;

--padding -> specifies the number of digits to pad the node name with;

--startIP -> specifies the network subnet mask;

--defaultGateway -> specifies the default route to send all packets;

Ex: 'oscar_clients install --imageName oscartest1 --numberOfHosts 2'.

(b) **delete** -> deletes a client from your running OSCAR cluster.

command: 'oscar_clients delete [options]'.

Options:

--clientID -> specifies the ID of the client that you want to remove from your cluster (the ID is formed by the base name plus the client's index number (that depends of the pad digits)).
Ex: 'oscar_clients delete --clientID oscarnode1'.

--all -> deletes all clients from the cluster.
Ex: 'oscar_clients delete --all'.

(c) **getState** -> get the current state of a determined client (also performs as a test that says if the client is working fine).

command: 'oscar_clients getState [option]'.

Where the only **option** here is to provide the client's ID:
--clientID -> the same (cluster base name + index number).
Ex: 'oscar_clients getState --clientID oscarnode1'.

5) Cluster

Collect the MAC adresses and assign specific IP adresses to the Clients, complete the installation of the cluster and test it.

(a) **collectMACs** -> collect the MAC adresses (you must "network boot" the clients so their MAC adresses can be collected). If the oscar global option 'dhcp' is set to 'static', after collecting the MACs the command 'configure DHCP Server' must be executed..... (should I include it to runs automaically ?)

command: 'oscar_cluster collectMACs'

(b) **getMACsList** -> get the list of the collected MACs.

command: 'oscar_cluster getMACsList [options]'.

The options here are **optional**.
If you just collected the MACs adresses list and wants to see it, you can omit the options. Ex: 'oscar_cluster getMACsList'.

Options:

--exportMACs -> save the list containing the collected MACs adresses in a file.
Ex: 'oscar_cluster getMACsList --exportMACs /tmp/listOfMACs'

--importMACs -> uses a file containing a list of MACs adresses (previously collected).
Ex: 'oscar_cluster getMACsList --importMACs /mnt/floppy/listA'.

(c) **takeAll** -> assign all the collected MACs to all the "free" nodes.

command: 'oscar_cluster takeALL'

? Is it important to keep a command that permits to assign a specific MAC to a specific node **?**

(d) **finishInstall** -> execute scripts that complete the installation of the cluster (must be executed only after all the nodes have completed their installation, reebooted from the hard disk and have their ethernet adaptors UP).

command: 'oscar_cluster finishInstall'

(e) **test** -> test the cluster to check if the OSCAR installation is working.

command: 'oscar_cluster test'.

Quick Install (minimal set of commands)

```
(1)   oscar_package install --interface eth1     (install only the core packages)

(2)   oscar_image build                                  (use the default options)

(3)   oscar_clients install --numberOfHosts 50  (must specify the cluster's size)

(4)   oscar_cluster collectMACs                        (collect the machines MACs)

(5)   oscar_cluster takeAll                  (automatically assign MACs adresses)

(6)   oscar_cluster finishInstall                        (finalizes installation)

(7)   oscar_cluster test                                 (test OSCAR installation)
```