

SOFA DESIGN HIERACHY

By Anthony Minessale II

Here is my design hierarchy which is layered:

LAYER 1 (CONCEPT) - *JUST* the core rules.

The laws that govern the system.

LAYER 2 (LIBRARY) - Programming Code that implements the rules.

(interface to the concept).

LAYER 3 (API) - Higher level arrangement of the library indented for developing applications.

(interface to the library).

LAYER 4 (APPLICATION) - Like the name suggests, an application of the API which uses the library to remain faithful to the concept.

LAYER 5 (IMPLEMENTATION) - Configuration, positioning, and execution of the application layer.

In my view, PBX lies at layer 5 (implementation) meaning you have such a fine control over the functionality that you can actually make a PBX purely from the abstraction level of the controller.

It's possible to make an application of your own that is entirely a controller and does all the things asterisk does but it is only passing around jabber id's and speaking commands back and forth.

I think the boundaries of these layers need to be clearly drawn for any success to happen.

Asterisk makes a huge mistake by blurring these lines to the point you can't even see them at all.

CONCEPT: To do everything. that is why it's called '*'

LIBRARY: Rewrite of most of libc and lots of code to mimic the kernel only more poorly and is strewn about the code base.

API: A collection of functions side by side in the same files as the library which, as pointed out, is strewn about as it is.

APPLICATION: a front-end to the pseudo library/api mess and collection of hackish modules with ungoverned cross-dependencies.

(in a few cases the core depends on certain modules to be loaded to even launch) Modules have access to 80+% of the core and have WAY too much power without even asking for it.

IMPLEMENTATION: This is the only one most people see and it's not too bad for what it is.

As promised by the concept you can configure it to do practically everything but the other 3 layers in between are a mess (probably a result of "to do everything" is a lofty goal best left to an operating system).

Since there are no boundaries it makes coding more difficult.

A seasoned programmer has a hard enough time as it is to recognize the boundaries on his own but then to obscure them as asterisk does along with the fact that many contributors are not seasoned programmers and have no chance to find the right spot to put things.

This results in a hapless entanglement of code that may look ok as assembly code but is unintelligible to human eyes which makes it a miracle that the application even executes without exploding.

A module should be implemented in the form of 1 single object that is declared and configured to request and allocate the exact things it needs from the core. It also needs a ruthless api layer (think of it as a firewall keeping stupid people away from your core).

The module should not be able to see any interface other than what is described in a module.h header file.

If you can gain access to what you need then the core is implemented wrong. Go back to layer 1 and see where you messed it up.

Most important I believe that when deciding on layer 1 it should only be up for debate for a small time once it's locked down it should be considered a permanent version and future additions will not be entertained. Asterisk cannot possibly achieve this but we can because we are not in the feature business we are trying to harness pure capability and let a higher level project deal with the features.

If the same group of people is working on that other project, that's fine just be sure to change your hat at the door and have a seat on the SOFA lol.