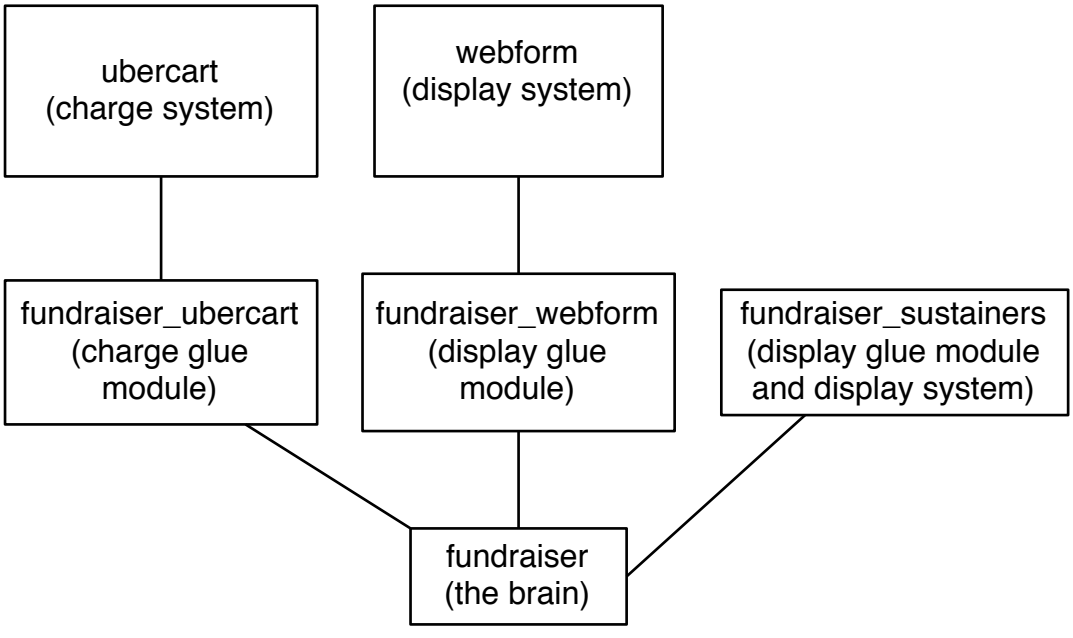

Fundraiser is meant to be agnostic in regards to how it interacts with any modules meant to charge a card (currently ubercart) or any modules meant to display fields to be filled out (currently webform primarily, but fundraiser_sustainers does it too). This allows for multiple systems to be used without changing the internal workings of fundraiser.

This is a complex problem.

The solution is also complex.

There's no getting around that.

But if your code always comes back to fundraiser, then the rest of the system will work with you as well. Always reference back to fundraiser, not to the glue modules, charge modules or display modules unless your system is already dependent on those for other reasons. Try to stay just as agnostic in your own development to minimize entanglement.

Look at the code in fundraiser_webform and fundraiser_sustainers for an idea re: how flexible this approach really can be. These are two modules that share a purpose (field display) and take different approaches to do it. Very handy to model code.

If we add a field (or register it with the fundraiser_field_info api) then both webform and sustainers will automatically pick it up if they need it.

The funderaiser field info api makes this possible. This api is designed to track all of the necessary field information within fundraiser, allowing all glue modules and referencing modules to use that field information as a central reference. It keeps it all in sync.

The field info api also allows modules to register overrides for displaying the field, validating it, and submitting it. This allows specific modules that need to provide information to do so without other modules having to be aware of the exact implementation. The best example of this can be found in fundraiser_ubercart where the country and state defaults and AJAX are defined.