Silverstripe E-commerce Customisation Chart

Once you have successfully installed your e-commerce application, it is time to start customising. This flow chart shows you what steps to take in what order (from easiest to hardest).

Question	yes → Action	Details
1. Can it be fixed by a setting in the Database?	Update database fields	(1) http://www.mysite.com/admin/shop/ and check settings (2) check page settings (or create required pages)
2. Is it a wording change?	Create your own language file and add your own translation.	1. what is your locale? Also check the locale of the user(s). 2. create new folder "zzz" with _config.php in it as well as a "lang" directory. 3. In the lang directory, add the required translations. Also see: http://doc.silverstripe.org/framework/en/topics/i18n
3. Can it be fixed by changing the config files?	Update the mysite/_config/ecommerce.yaml file or your mysite/_config.php	(1) go to http://www.mysite.com/dev/ecommerce/ecommercecheckconfiguration (2) check ecommerce/_config.php for ideas (3) search code for "static function set_" to see if there are other settings available.
4. Can it be fixed with CSS?	Theme appropriate CSS file and adjust as needed. Use !important in css definition if you have no other choice.	Check Source to see what CSS files are used and try to "Theme" the most appropriate CSS file. For more information on theme-ing, visit: http://doc.silverstripe.org/framework/en/topics/theme-development.
5. Can it be fixed by adjusting the HTML?	Theme appropriate template file (.ss) and change as needed.	Check the source to see what HTML files are used and "Theme" the most appropriate HTML template (.ss) file. For more information on theme-ing, visit: http://doc.silverstripe.org/framework/en/topics/theme-development. Try to theme as few as possible files (e.g. only the include).
6. Is there a sub-module that can add the functionality you are looking for?	Install sub-module.	Visit http://www.silverstripe.org/modules or http://www.ssmods.com/modules to find the module you are looking for.
7. Do you want to change the display / functionality of products?	Extend the Product / ProductGroup class.	The e-commerce sub-modules as well as many other Silverstripe modules contain a wealth of examples on how to do this. Make sure to "override" exactly the methods you need to be different.
8. Do you want to add a new type of product that does not suit Product or Product Variation?	You can create your own "Buyable", which should follow the concepts shown of Product (Variation), but can be a stand-alone DataObject.	Make sure you use the BuyableInterface, which will help you add the right "methods" (functions) to your custom Buyable Class. Make sure to update ecommerce.yaml with the additional Buyable.
9. Do you want to add a discount / charge / tax other crazy thing that is dependent on the content of the order?	Create a class that extends OrderModifier	There are plenty of examples in the e-commerce (sub) modules you can use as a starting point. There is even a complete module dedicated to being an OrderModifier Example. They all extend the OrderModifier Class. Make sure to update ecommerce.yaml with the additional OrderModifer.
10. Do you want to add a general note to your order that is not dependent on the items added to the order?	Create an Order Status Log.	There are plenty of examples in the e-commerce (sub) modules you can use as a starting point. Make sure to update ecommerce.yaml with the additional OrderStatusLog. An OrderStatusLog is often used in conjunction with an OrderModifier.
11. Do you want to change something in the Order Process? E.g. sending an extra e-mail, beaming the order to a third-party app or just shortening the process.	Add / Edit / Remove any of the OrderStep	Removing and editing, in some instances, can be done in the database (see: http://www.mysite.com/admin/shop/), however, in other situations, you may want to create your own OrderStep and add it to the /mysite/_config/ecommerce.yaml file.
12. Do you want a different payment method?	Create your own payment class.	Strictly speaking, the Payment module is outside of the scope of e-commerce, but it is a requirement of course. There are plenty of Payment Class examples out there... Create your own class and add a reference to it in your Payment::set_supported_methods(...); call /mysite/_config.php.
13. Can you extend a class to achieve your result?	Use the standard MyClass extends CoreClass and override any methods and variables you want changed.	See http://doc.silverstripe.org/framework/en/topics/datamodel for more information.
14. Can a class "decorator" / "extension" sort out your worries?	Add a decorator / extension	Within the e-commerce code, we have added a number of "hooks", they often look like this: $this->extend or similar... you can use these "hooks" to change the actions and outputs of core methods. See: http://doc.silverstripe.org/framework/en/topics/datamodel for more information.
15. Can you create a custom class that replaces the core class?	Create custom class to replace a core one.	Create your custom class and add the following to your mysite/_config.php file: Object::useCustomClass('CoreClass','MyClass');
16. Can you hack the core to make it work?	Make a fork on Github or just start hacking away.	A fork on github is the preferred method here. If you are using SVN, you can look at Piston as a solution.
17. Contact the e-commerce developers to discuss options.	Project Home: http://code.google.com/p/silverstripe-ecommerce/ GIT Home: https://github.com/sunnysideup/silverstripe-ecommerce Demo Home: http://www.silverstripe-ecommerce.com	