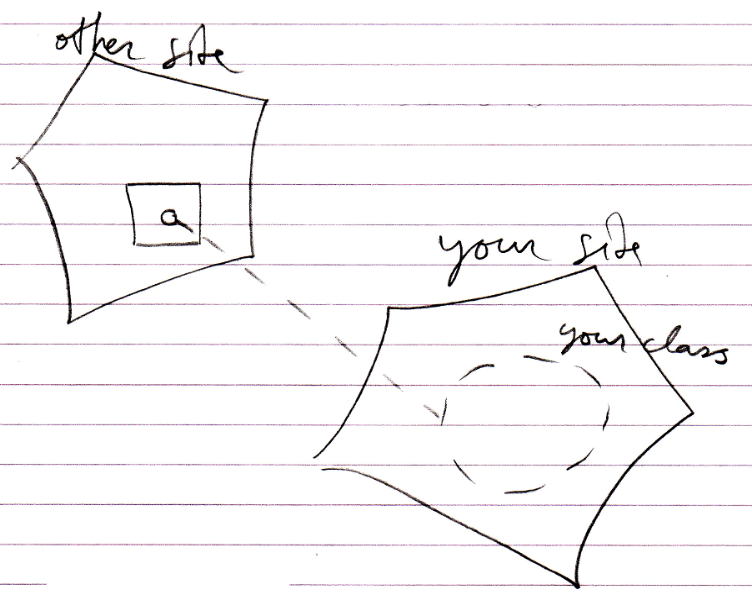
|  |
| --- |
| Circle Language Broader View | Interfaces |

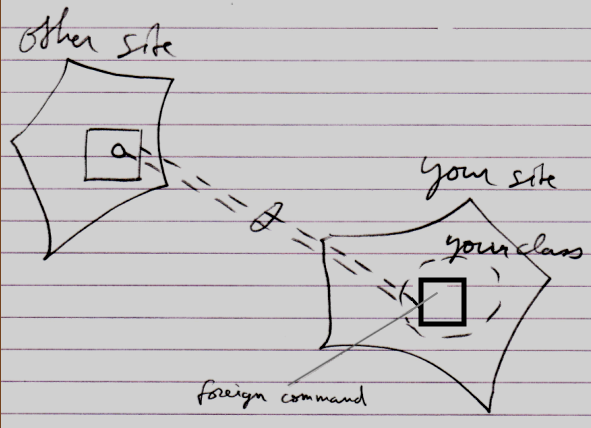
## Group By Source

Your class can automatically be extended with extra commands from other sites, if you allow other sites to form bidirectional relations with your class.

The following forward relation:



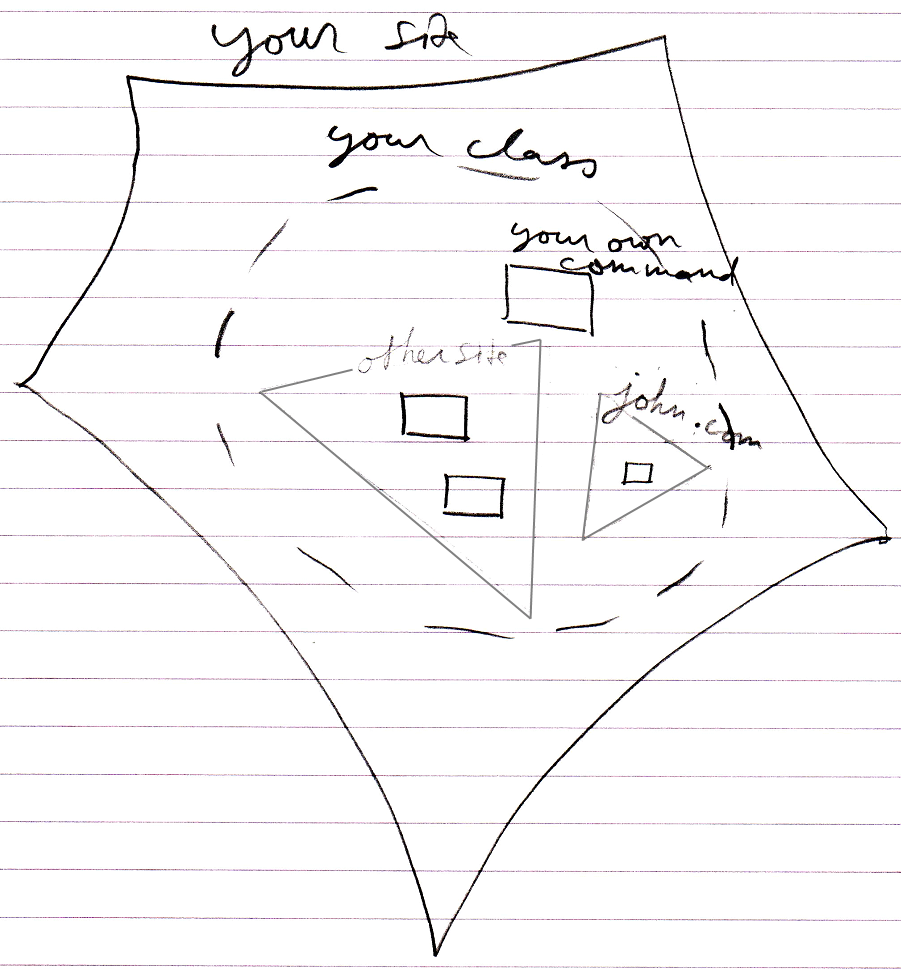
Can add a backward relation:



This creates an extra command inside your class, that is defined on another site.

The danger arises, that your class gets extended with commands that are not from a reliable source. To battle this danger a mechanism of control may need to be imposed.

The group by source concept is such a mechanism. Group by source introduces imaginary interfaces that group related objects by the site.



This means you get to see which site commands are defined on.

If you use a command from some vague site, then the command is not reliable. If you access a command through a class, and that command is hosted elsewhere, you have to know you can trust that other site.

You have to be aware of what site a command is from and estimate if your consider this site reliable before you run a command. You have to do with the whole internet now, so you have to be aware of where stuff comes. This counts for everything: commands, classes, objects, modules. You just need to be aware what site you are accessing and whether you consider it a safe source.

### Other Thoughts

Perhaps the concept of group by source can be inspiration for grouping by other sources than site, like group by module for instance. It does not need to be a site, that is the logical residence of a command; the logical residence of a command theoretically could also be an object. Each *object*, that locally defines commands for a class, could create a separate imaginary interface for that class.

Imaginary interfaces seemingly bring more stability to interfaces. Imaginary interfaces are author-bound, so one clumsy author can not mess up an interface authored by the one that authored the class himself. However, do remember that the real protection to a volatile interface is making backwards relations Private or not automatically establishing backward relations at all.

A command can only be reliable, really, when the people authoring and using the command have a clear agreement about it. That is no different in other systems. Somebody hosting a web service could very well change the interface of the service, and have software, dependent on that service, crash. The only reason that wouldn’t happen, is because people have an agreement with eachother, not to change the interface just like that. And at times it *does* change unannounced, it really means systems will have problems.

Even interfaces of components on your local machine are only stable because of the agreement between you and the producer of the component. If they decided to automatically update the component, the interface or behavior of the component may very well change, and this may cause problems for your software. When it does not do that, than this is because of the agreement, that you have with the other party.

Perhaps this concept can also be inspiration for being able to apply custom imaginary groupings for other analysis purposes.

An alternative method for grouping members by source is be to give things a different color in diagrams and in text code, depending on what site things are coming from.

Also remember that it does not only count for commands. Objects can also be grouped by source.