Kayla Carr

11 February, 2016

Will Code For Food

**Metric: Modularity**

The encapsulation ratio:

|  |
| --- |
| Calculates the ratio of encapsulated classes to all classes for each package. A class is said to be encapsulated if it is not used externally to the package. In general, most of the classes in a package should not be used externally to the package, so this ratio should be fairly high. |

|  |  |
| --- | --- |
| **Package** | **Encapsulation Rate** |
| *Total Average* | *53.61%* |
| controller | 0.00% |
| controller.api\_interfaces | 33.33% |
| controller.authentication | 33.33% |
| controller.retrieval | 25.00% |
| model.common | 100.00% |
| model.common.common | 25.00% |
| model.common.common.sorting | 75.00% |
| model.common.common.users | 0.00% |
| model.common.form | 16.67% |
| model.common.messaging | 50.00% |
| model.common.resources | 100.00% |
| model.formTests | 100.00% |
| model.getInvolved | 0.00% |
| model.resources | 16.67% |
| model.ridesharing | 20.00% |
| tasks | 0.00% |
| values | 16.67% |
| view | 100.00% |
| view.common | 76.19% |
| view.dynamic\_form | 100.00% |
| view.events | 20.00% |
| view.getinvolved | 88.89% |

**Analysis**

Our encapsulation rate increased from the midterm value by a few percentage points. This shows that with more experience we’re getting better at organizing our project. Our views are all fairly well encapsulated, with the exception of the events view. This is likely because the events view doubles as both our events and ridesharing views. This is a sign that perhaps we should try to decouple events and ridesharing from each other a bit more so they aren’t as reliant and are more encapsulated.

As expected, our controllers have little or no encapsulation since they are used by our application to load our objects from the database. Our models also have rather low encapsulation, but again this is to be expected.