The New Vdm2Uml Handler

`Vdm2UmlMain` is the name of the new handler for the transformation and it is executed when the file is run on the JAR. The new handler achieves the same functionality as the old handler, without utilising the notion of an eclipse project. In a lot of cases, this is done simply by accessing the methods used by the methods of `Iproject` and its derived classes. This results in the same functionality, but the methods used to do so are accessed at a lower level. The ~~following~~ section ~~will now~~ show examples of how this strategy is applied.

In the implementation of the old handler, the file handling is done implicitly by Eclipse. However as this is no longer a possibility, the file handling is done internally by `Vdm2UmlMain`. This is achieved by the `Vdm2UmlMain` method '`main`', taking a set of arguments which denote the directory containing the files to be transformed and the directory where the resulting UML file should be placed. The arguments are distinguished by using `-folder` and `-output` as flags, before the path arguments.

Similarly, the old handler reads the VDM dialect from a method on the `IVdmProject` class. The new handler is instead passes the dialect along with the command to execute the JAR file.

While the `IVdmModel` class has an attribute informing whether the model is type ~~and parse~~ correct, the methods for performing the actual checks are not dependent on the Eclipse project classes, and are therefore fit for reuse. The class list is also provided by the type checker, `typeCheckPp` or `typeCheckRt`, depending on the dialect of the VDM files. The two methods of the class `TypeCheckerUtil` initialise an instance of the `TypeCheckResult` class, which has an attribute containing the class list needed for the conversion.

For most of the implementation of the new handler, only the handler itself is changed and all the underlying code is reused. The one exception to this is how the UML file is saved to the specified output directory. The `Vdm2Uml` method named `save` is responsible for this. Modifying the code was necessary since the Eclipse classes used are are an external resource in a standalone application and not run on Eclipse IDE like Overture is [20].

The consequence of this is that the link to these resources have to be handled explicitly, where they were handler automatically by Eclipse before [4]. A similar change is made in the UML2VDM implementation when loading a UML resource.

The original code was changed from:

```
Resource resource = new ResourceSetImpl()
    .createResource(uri.appendFileExtension(UMLResource.
    FILE_EXTENSION));
resource.getContents().add(modelWorkingCopy);
```

To: