Function generation developer technical documentation.

Author

Piotr Jessa

Synopsis

The technical view on the Function Generator eclipse plugin.

Hard Dependencies

Those are included in plugin as code or as referenced JAR:

- ecj - in version 20 used to perform symbolic regression problem
- ini4j - used for saving and reading the configuration from the INI files
- tools - bunch of tools for code formating used for displaying the result ofthis program working
- JFreeChart & JCommons - tools for creating 2D chart (visualization)
- javaasist - used for dynamic creation of classes during runtime for registering them in ECJ

Soft Dependencies

Those are included via OSGi require attributes:

- org.apache.commons.logging - used for logging facilities

Structure

TODO: this structure needs to be refactored

- FunctionGenerator - PDE Plugin - main project
- FunctionGeneratorFeature - Eclipse Feature Project (groups plugins)
- FunctionGeneratorUpdateSite - Allows creation of the update site

Tests:

All tests are located in test folder. They are JUnit4 simple test. Most of them work on the integration level (testing INI file saving, CVS file saving). Some work on unit level (quite the isolation).

Not all the code is covered. The tests where used to provide faster means of debbugging application rather than *TDD* approach.

Entry Point

FGRunnable class is responsible for running all of the code. The eclipse entry of the project is CommandHandler.

Core Concepts:

- *Settings*:

  Stores all the settings of the program. Like chosen operations and evolution parameters. Use ISettingsLoader to store settings in some form of persistence. By now it is only INISettingsLoader class.

- *Data*:

  Data are stored as the List and are stored within PointsTableModel class. Can be dumped to file by implementing IDataLoader interface.

- *IOperationProvider*:

  The concept behind this interface is to provide some abstraction over the GPNode.

That allows dynamic creation of the GPNode during program runtime. User specifies the literal value for example: 7.0 and the corresponing GPNode class is created and loaded into ClassPool.

- *IOperationFactory*:

  The list of available operations for user aka List.

- Engine:

  Main runner of the evolution. Made up of two phases:

- init : all code is loaded into ECJ parameters file
- run : evolution goes

For now the engine also manages the creation of the template of class with function, which is not the best solution and is possible future refactoring possibility.