**SCS V4.0.0**

**USER DOCUMENTATION**

Contents

[1. What is SCS? 2](#_Toc25017013)

[2. Main Concepts 2](#_Toc25017014)

[3. Requirements: 3](#_Toc25017015)

[4. How to install SCS? 3](#_Toc25017016)

[5. Running a model 3](#_Toc25017017)

[6. Configuration Files 5](#_Toc25017018)

[7. Experiment File 6](#_Toc25017019)

# What is SCS?

SCS is a lightweight software framework that allows to define and execute robot experiments (originally simulating rat experiments). The framework was developed with the following goals in mind:

* **Speed**: simulations need to run as fast as possible.
* **Ease of use**: the framework should be simple to learn and use
* **Extensibility**: the framework should be easily extended to be used with any simulators, robots (both real and simulated) and GUIs.
* **Code reutilization**: the code should be highly reusable allowing users to define their experiments with as few lines as possible.

# Main Concepts

The main concepts in the framework are shown in the following diagram. An experiment defines a set of trials to be performed by a set of subjects and provides mechanism to control and visualize the simulation. Each subject represents an individual and controls a robot. The subject defines higher-level logic while the robot implements the lower level routines. The use of robot modules and proxies allow to abstract the robot routines from the particular robot implementation used allowing to reuse as much code as possible when switching robots / simulators.

# Requirements:

SCS requires java 10 or later.

Optional scripts (recommended) require python with numpy and pandas.

# How to install SCS?

SCS doesn’t need to be installed. To use the framework, you only need to download and add the jar to your class path. If using maven (recommended), you only need to add the following lines of code to your pom file (replacing the version number by the desired SCS release version):

<repositories>

<repository>

<id>scs</id>

<url>http://raw.github.com/biorobaw/scs</url>

</repository>

</repositories>

<dependencies>

<dependency>

<groupId>com.github.biorobaw.scs</groupId>

<artifactId>scs</artifactId>

<version>0.1.0</version>

</dependency>

</dependencies>

# Running a model

To run a model, execute the following command:

java -cp **CLASSPATH** com.github.biorobaw.scs.Main **ARGS**

**ARGS = CONFIG\_FILE CONFIG\_NUM LOG\_FOLDER [CREATE\_FOLDERS]**

Where:

**CLASSPATH** defines the classpath, scs should be included in it. See java documentation for more info about the classpath.

**CONFIG\_FILE** is the path to a csv config file (see section “Configuration files”).

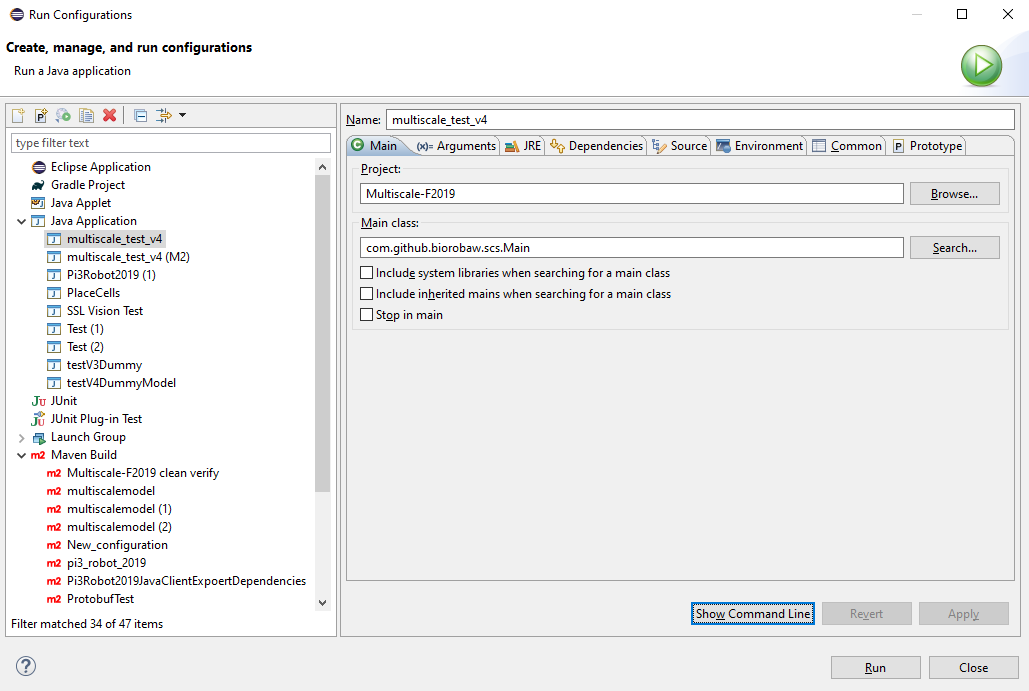
**CONFIG\_NUM** is the number of the configuration to be executed from the config file. The number is zero based (i.e. starts from 0).

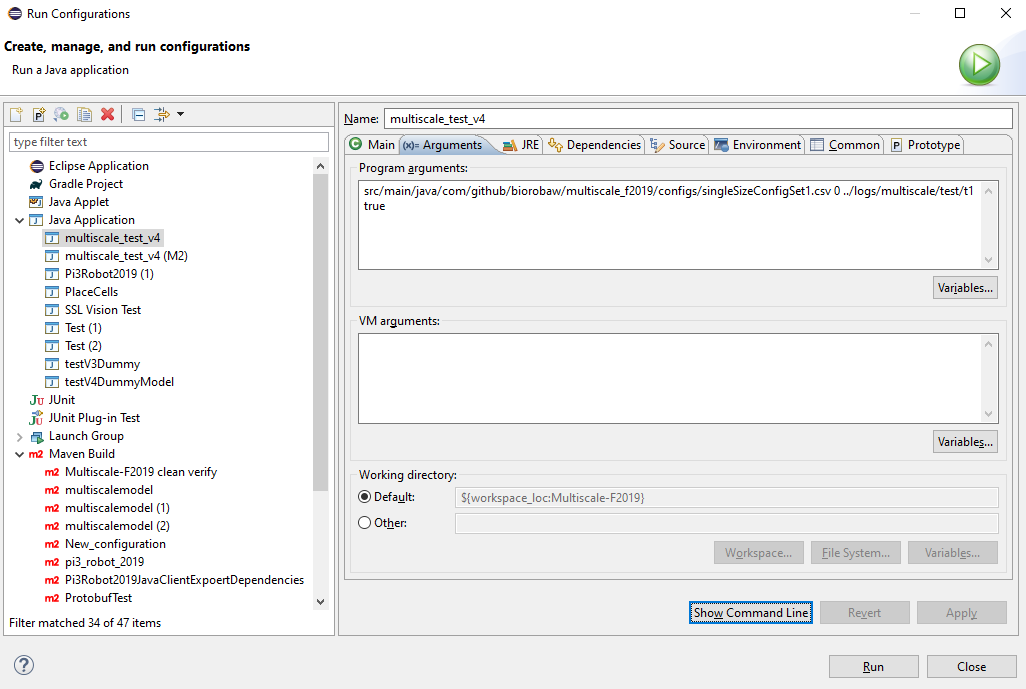
**LOG\_FOLDER** is the path to a csv config file (see section)

**CREATE\_FOLDERS** optional boolean argument specifying whether the log folders should  
be created before performing the experiment. Omitting the argument defaults to false. When running multiple concurrent simulations, you want to manually create the log folders (see “Running on USF’s cluster” for instructions on how to do so) and either omit the argument or set it to false.

As a result of the command, an instance of SCS will start executing the experiment defined by the specified line in the specified csv configuration file.

The following images show a sample run configuration defined in eclipse in a project using SCS. Note the main class and the arguments in the image.





# Configuration Files

Configuration files are csv files that define a set of simulations to be performed. Although these files are meant for instantiating multiple parallel simulations in a cluster, their use was made compulsory to streamline the instantiation method.

A configuration file defines a table of parameters, where the first line defines the column headers and each line thereafter represents a row in the table. Note that columns must be separated by any type of whitespace. It is important not to use commas to separate columns since commas are used to parse list/array arguments. Also, blank rows are not currently supported, and all elements in the table are case sensitive.

To write a configuration file, you need to specify at least the following 4 columns:

* **experiment –** defines the path to an xml file describing the experiment to be performed (see section “Experiment Files” for more details).
* **config –** defines a name for a specific configuration of the experiment
* **group –** defines a group of subjects (from the experiment file) to be executed
* **run\_id –** defines an id to store the results of this simulation. Logging is done to LOG\_FOLDER/config/run\_id

Other columns can be defined as required. The order in which the columns are defined is irrelevant as long all rows are consistent with the headers. Python scripts that simplify the generation of configuration files are available in the “scripts” folder found in scs top folder (see section “scripts” for more information).

At the start of a simulation, a map is generated mapping each column header to the values specified by the line provided as a command argument (see “Running a model” to see the command arguments).

The following is a sample configuration file found used in the example model “dummy\_model” (src/main/java/github/biorobaw/scs/examples/dummy\_model/configs.csv):



# Experiment Files

**How is an experiment**

**Creating a model.**

To create a model, you only need to ex

How to xs?

How to add support for another simulator?