# VRE template Tool

2020-2022, Barcelona Supercomputing Center (BSC)

Aug 30, 2022

# CONTENTS

This documentation extends the specifications detailed in the Deliverable 6.1 document: *"Design of computational architecture of software modules"* (http://bit.ly/MuGD6_1) to provide a simple programming template to develop new tools for an open-source Virtual Research Environment (GitHub VRE). Tools are implemented as thin wrappers over existing software to facilitate their integration into the VRE.

CHAPTER

ONE

# REQUIREMENTS AND INSTALLATION

## 1.1 Requirements

### 1.1.1 Software

- Python 3.6+
- Git

### 1.1.2 Python Modules

- openvre-tool-api

Tool API. This API provides a standard way for all tools to be wrapped to allow for a common interface layer.

## 1.2 Installation

Directly from GitHub:

```
git clone https://github.com/inab/vre_template_tool.git
cd vre_template_tool
```

Create Python environment:

```
python3 -m venv venv
source venv/bin/activate
pip install --upgrade wheel
pip install -r requirements.txt
```

## 1.3 Documentation

To build the documentation:

```
cd docs
pip install -r requirements.txt
make html
```

Documentation will be generated (in HTML format) inside the `_build/html` directory.

CHAPTER

TWO

# TUTORIAL

In this tutorial, we'll assume that `vre_template_tool` is already installed on your system. If that's not the case, see *Installation Section*.

The following is a walk through of developing a tool and a wrapper to include a new functionality within the VRE. Some stages are covering the tool development, using the tool and defining the configuration files required so that the final product can be integrated into de VRE.

This tutorial will walk and help you through the following tasks for creating a tool ready to be integrated into de VRE.

## 2.1 Creating a tool

Before you start creating your tool, the basic entity *Tool* of the *basic_modules* from the openvre-tool-api library is imported.

```
from basic_modules.tool import Tool
```

The *Tool* is the core element to running an application within the VRE. It defines the procedures that need to happen to prepare the data along with the application to run over chunks of data provided. An application can be either a piece of code that is written in Python or R that is run with given chunks of data or defined parameters. The results are then returned to the calling method *toolExecution()*.

All tools contain at least a *run()* method which takes the input files, defined output files, and relevant metadata. Returned by the run method is the output files and their metadata.

All the tools should be placed within the *tools* directory within the package.

### 2.1.1 Your first *myTool*

Use the `tool/VRE_Tool.py` script as a template to create you new tool.

The script contains *myTool* a class that you can define to run your application. You must define the location of your application to run, e.g. `/example/hello.py`.

You can see the first lines of *myTool* class saved in a file named `VRE_tool.py` under the `tool/` directory from the project's top level folder. In this class, we define the application we want to run using the wrapper. The application can be implemented in different programming languages, such as Python or R, taking into account how to execute it. To see how to execute your application see *Command line tool* and *Adding software dependencies* sections.

```
class myTool(Tool):
    """
    This class define <myTool> Tool.
```

(continues on next page)

(continued from previous page)

```
    """
    DEFAULT_KEYS = ['execution', 'project', 'description']
    """config.json default keys"""
    PYTHON_SCRIPT_PATH = "/example/hello.py"
    """<myApplication>"""

... (omitted for brevity)
```

*myTool* class defines two attributes and some methods. Specifically:

- *DEFAULT_KEYS*: identifies default arguments from openVRE configuration file (`config.json`).
- *PYTHON_SCRIPT_PATH*: location of your application that you wanna run with the wrapper. You can use a relative or absolute path.
- *run()*: *TO BE DOCUMENTED ...*
- *toolExecution()*: *TO BE DOCUMENTED ...*

### 2.1.2 Adding input files

If your application expects one or more input files, you must add them as follows in the method *toolExecution()*:

```
input_file_1 = input_files.get('hello_file')
if not os.path.isabs(input_file_1):
    input_file_1 = os.path.normpath(os.path.join(self.parent_dir, input_file_1))
```

For each input file, you must add a new variable. The VRE works with absolute paths and as you can see, it is preferable to check if the directory is absolute or not; which in case it is not, becomes absolute.

### 2.1.3 Adding arguments

If your application expects one or more arguments, you must add them as follows in the method *toolExecution()*:

```
argument_1 = self.arguments.get('username')
if argument_1 is None:
    errstr = "argument_1 must be defined."
    logger.fatal(errstr)
    raise Exception(errstr)
```

For each argument, you must add a new variable and validate their importation from openVRE configuration file (`config.json`) to ensure the execution of the application.

### 2.1.4 Adding output files

If your application expects one or more output files, you must add them as follows in the method *toolExecution()*:

```
output_id = output_metadata[0]['name']
output_type = output_metadata[0]['file']['file_type'].lower()
output_file_path = glob(self.execution_path + "/*." + output_type)[0]
if os.path.isfile(output_file_path):
    output_files[output_id] = [(output_file_path, "file")]
```

For each output file, you only need to extract its *identifier*, *file type*, and *file path* from the output_metadata that contains the information from openVRE configuration file (`in_metadata.json`); and save it to *output_files*. As you can see, we recommend to check the existence of each output file.

---

**Note:** If your application generates multiple output files, we recommend to develop a separate method. You can see an example HERE.

---

### 2.1.5 Command line tool

```
cmd = [
    'python',
    self.parent_dir + self.PYTHON_SCRIPT_PATH,  # hello.py
    input_file_1,  # hello.txt
    argument_1,  # username
]
```

## 2.2 Adding software dependencies

If you need some software requirements to run your application, you must add them to the file VRE_RUNNER, in the project's top level directory.

```
DEPENDENCIES=("Rscript", "docker")
```

## 2.3 Integrating the new tool into VRE

The final step is the integration of the tool into VRE. The X section should provide a guide to the initial JSON files. The full JSON specifications is located in this GoogleDoc. the details the requirements for correctly creating the Tool description JSON file and the requirements for parameters needed for an application.

VRE template Tool is written in Python. If you're new to the language you might want to start to learn Python quickly, the Python Tutorial is a good resource.

**What next**

Now that you have and idea of what the `vre_template_tool` package provides, you should investigate the parts of the package most useful for you and see an example of how to use it.

*Reference Section* provides details on VRE template Tool.

*Examples Section* provides an example using VRE template Tool.

CHAPTER

THREE

# REFERENCE

## 3.1 VRE_Tool

**class** tool.VRE_Tool.**myTool**(*configuration=None*)

Bases: Tool

This class define <myTool> Tool.

### Attributes Summary

| | |
|---|---|
| *DEFAULT_KEYS* | config.json default keys |
| *PYTHON_SCRIPT_PATH* | <myApplication> |

### Methods Summary

| | |
|---|---|
| *__init__*([configuration]) | Init function. |
| *run*(input_files, input_metadata, ...) | The main function to run the <myTool> tool. |
| *toolExecution*(input_files) | The main function to run the <myTool> tool. |

### Attributes Documentation

**DEFAULT_KEYS = ['execution', 'project', 'description']**

config.json default keys

**PYTHON_SCRIPT_PATH = '/example/hello.py'**

<myApplication>

### Methods Documentation

**__init__**(*configuration=None*)

Init function.

**Parameters**
**configuration** (*dict*) – A dictionary containing parameters that define how the operation should be carried out, which are specific to <myTool> tool.

**run**(*input_files*, *input_metadata*, *output_files*, *output_metadata*)

The main function to run the <myTool> tool.

**Parameters**

- **input_files** (*dict*) – Dictionary of input files locations.
- **input_metadata** (*dict*) – Dictionary of input files metadata.
- **output_files** (*dict*) – Dictionary of output files locations expected to be generated.
- **output_metadata** (*list*) – List of output files metadata expected to be generated.

**Returns**
Generated output files and their metadata.

**Return type**
dict, dict

**toolExecution**(*input_files*)

The main function to run the <myTool> tool.

**Parameters**
**input_files** (*dict*) – Dictionary of input files locations.

## 3.2 VRE_RUNNER

**class** VRE_RUNNER.**Wrapper**(*configuration=None*)

Bases: `object`

Functions for wrapping the tool set up and execution.

### Methods Summary

| | |
|---|---|
| *__init__*([configuration]) | Initialise the tool with its configuration. |
| *run*(input_files, input_metadata, ...) | Main run function for running <myTool> tool. |

### Methods Documentation

**__init__**(*configuration=None*)

Initialise the tool with its configuration.

**Parameters**
**configuration** (*dict*) – A dictionary containing parameters that define how the operation should be carried out, which are specific to <myTool> tool.

**run**(*input_files*, *input_metadata*, *output_files*, *output_metadata*)

Main run function for running <myTool> tool.

**Parameters**

- **input_files** (*dict*) – Dictionary of input files locations.
- **input_metadata** (*dict*) – Dictionary of input files metadata.
- **output_files** (*dict*) – Dictionary of output files locations expected to be generated.
- **output_metadata** (*list*) – List of output files metadata expected to be generated.

**Returns**
Generated output files and their metadata.

**Return type**
dict, dict

VRE_RUNNER.**main_wrapper**(*config_path*, *in_metadata_path*, *out_metadata_path*)

Main function.

This function launches the tool using configuration written in two json files: config.json and in_metadata.json.

**Parameters**

- **config_path** (*str*) – Path to a valid VRE JSON file containing information on how the tool should be executed.
- **in_metadata_path** (*str*) – Path to a valid VRE JSON file containing information on tool inputs.
- **out_metadata_path** (*str*) – Path to write the VRE JSON file containing information on tool outputs.

**Returns**
If result is True, execution finished successfully. False, otherwise.

**Return type**
bool

## 3.3 License

vre_template_tool is distributed with Apache License 2.0.

```
                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
```

(continues on next page)

(continued from previous page)

```
1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
```

(continues on next page)

(continued from previous page)

```
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      (except as stated in this section) patent license to make, have made,
      use, offer to sell, sell, import, and otherwise transfer the Work,
      where such license applies only to those patent claims licensable
      by such Contributor that are necessarily infringed by their
      Contribution(s) alone or by combination of their Contribution(s)
      with the Work to which such Contribution(s) was submitted. If You
      institute patent litigation against any entity (including a
      cross-claim or counterclaim in a lawsuit) alleging that the Work
      or a Contribution incorporated within the Work constitutes direct
      or contributory patent infringement, then any patent licenses
      granted to You under this License for that Work shall terminate
      as of the date such litigation is filed.

   4. Redistribution. You may reproduce and distribute copies of the
      Work or Derivative Works thereof in any medium, with or without
      modifications, and in Source or Object form, provided that You
      meet the following conditions:

      (a) You must give any other recipients of the Work or
          Derivative Works a copy of this License; and

      (b) You must cause any modified files to carry prominent notices
          stating that You changed the files; and

      (c) You must retain, in the Source form of any Derivative Works
          that You distribute, all copyright, patent, trademark, and
          attribution notices from the Source form of the Work,
          excluding those notices that do not pertain to any part of
          the Derivative Works; and

      (d) If the Work includes a "NOTICE" text file as part of its
          distribution, then any Derivative Works that You distribute must
          include a readable copy of the attribution notices contained
          within such NOTICE file, excluding those notices that do not
```

(continues on next page)

(continued from previous page)

```
        pertain to any part of the Derivative Works, in at least one
        of the following places: within a NOTICE text file distributed
        as part of the Derivative Works; within the Source form or
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
    any Contribution intentionally submitted for inclusion in the Work
    by You to the Licensor shall be under the terms and conditions of
    this License, without any additional terms or conditions.
    Notwithstanding the above, nothing herein shall supersede or modify
    the terms of any separate license agreement you may have executed
    with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
    names, trademarks, service marks, or product names of the Licensor,
    except as required for reasonable and customary use in describing the
    origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
    agreed to in writing, Licensor provides the Work (and each
    Contributor provides its Contributions) on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
    implied, including, without limitation, any warranties or conditions
    of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
    PARTICULAR PURPOSE. You are solely responsible for determining the
    appropriateness of using or redistributing the Work and assume any
    risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
    whether in tort (including negligence), contract, or otherwise,
    unless required by applicable law (such as deliberate and grossly
    negligent acts) or agreed to in writing, shall any Contributor be
    liable to You for damages, including any direct, indirect, special,
    incidental, or consequential damages of any character arising as a
    result of this License or out of the use or inability to use the
    Work (including but not limited to damages for loss of goodwill,
    work stoppage, computer failure or malfunction, or any and all
```

(continues on next page)

(continued from previous page)

```
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

## 3.4 Credits

`vre_template_tool` was originally written by Laura Rodríguez-Navas and Laia Codó-Tarraubella, and has been developed with the help of many others. Thanks to everyone who has improved VRE template Tool by contributing code, bug reports (and fixes), documentation, and input on design, features, and the future of VRE template Tool.

### 3.4.1 Contributions

If you are a VRE template Tool contributor, please feel free to open an issue or submit a pull request to add your name to the bottom of the list.

- Laura Rodríguez-Navas, GitHub: lrodrin
- Laia Codó-Tarraubella, GitHub: lcodo

CHAPTER

FOUR

# EXAMPLES

The best way to learn is with examples, and `vre_template_tool` is no exception. For this reason, there is an example you can test and it is available at tests/basic/ directory. The example is a test tool that takes an input file, then append an username of characters in that file and finally prints the result to a second file.

## 4.1 Default structure

Before delving into the details, let's first understand the directory of a test. Though it can be modified, all VRE Tool projects have the same file structure by default, similar to this:

```
tests/basic/
     config.json
     in_metadata.json
     README.md
     test_VRE_RUNNER.sh
```

## 4.2 Configuration files

There are 2 configuration JSON files as inputs for the test instance. These describe the input and output files and required arguments that need to get passed to the application. These configuration files are those that would get passed to the tool by the VRE.

The test instance will look for configuration files in:

1. `tests/basic/config.json`
2. `tests/basic/in_metadata.json`

### 4.2.1 config.json

Defines the configurations required for by the application including parameters that need to be passed from the VRE submission form, file and the related metadata as well as the output files that need to be produced by the application.

```
{
    "input_files": [
        {
            "name": "hello_file",
            "value": "unique_file_id_5e14abe0a37012.29503907",
```

(continues on next page)

(continued from previous page)

```
            "required": true,
            "allow_multiple": false
        }
    ],
    "arguments": [
        {
            "name": "execution",
            "value": "tests/basic/run000"
        },
        {
            "name": "project",
            "value": "example"
        },
        {
            "name": "description",
            "value": "test"
        },
        {
            "name": "username",
            "value": "Laura"
        }
    ],
    "output_files": [
        {
            "name": "goodbye",
            "required": true,
            "allow_multiple": false,
            "file": {
                "file_type": "TXT",
                "data_type": "result",
                "meta_data": {
                    "visible": true,
                    "description": "Result of tool application example"
                }
            }
        }
    ]
}
```

In the arguments there 3 sets (execution, project and description) that will always be present and are provided by the VRE at the point of submission of the to the tool. These are the name of the project that has been given in the VRE, the execution path that is the location for where the input files are located and can be used as the working directory for the tool. Other parameters in the arguments list are elements based on what parameters the tool requires from the user at run time.

### 4.2.2 in_metadata.json

List of file locations thst are used as input. The configuration names should match those that are in `config.json` file defined above.

```
[
    {
        "_id": "unique_file_id_5e14abe0a37012.29503907",
        "type": "file",
        "file_path": "example/hello.txt",
        "file_type": "TXT",
        "data_type": "input_file",
        "compressed": 0,
        "sources": [],
        "user_id": "user_id",
        "creation_time": {
            "$date": {
                "$numberLong": 1612777323000
            }
        },
        "meta_data": {
            "size": 0,
            "project": "example",
            "atime": {
                "$date": {
                    "$numberLong": 1612777323000
                }
            },
            "parentDir": "unique_file_id_5e14abe0a37742.64003100",
            "lastAccess": {
                "$date": {
                    "$numberLong": 1612777323000
                }
            }
        }
    }
]
```

## 4.3 Running the tool

To run the tool it needs a config.json file and in_metadata.json file to provide de input. In this case, it uses the example configuration files saved in `tests/basic` directory.

To put the tool to work, go to the project's top level directory and run:

```
./VRE_RUNNER --config tests/basic/config.json --in_metadata tests/basic/in_metadata.json␣
↪--out_metadata out_metadata.json --log_file VRE_RUNNER.log
```

This command runs the tool that we've just created and you will get an output similar to this:

```
2021-08-06 13:08:34 | INFO: 0) Unpack information from JSON
2021-08-06 13:08:34 | INFO: 1) Instantiate and Configure Tool
```

(continues on next page)

(continued from previous page)

```
2021-08-06 13:08:34 | INFO: 2) Run Tool
2021-08-06 13:08:34 | PROGRESS: <myApplication> execution finished successfully -␣
↪FINISHED
2021-08-06 13:08:34 | INFO: 3) Create information
2021-08-06 13:08:34 | INFO: 4) Pack information to JSON
2021-08-06 13:08:34 | PROGRESS: <myTool> tool successfully executed;
```

### 4.3.1 Results

This will create a `run000` directory with the following contents:

```
run000/
    goodbye.txt
    out_metadata.json
    tool.log
```

Now, check the files in the `run000` directory. You should notice that a new file has been created: `output_metadata.json`, with the content for the respective results, as the *run* method instructs.

**Note:** If you are wondering about the `run000` directory, you can see an example in the *Examples Section*.

CHAPTER

FIVE

# INDICES AND TABLES

- genindex
- modindex
- search

# INDEX

## Symbols



## D



## M



## P



## R



## T



## W