

GSoC-2019 PROJECT PROPOSAL

ORGANISATION: PYTHON SOFTWARE FOUNDATION

SUB ORGANISATION: TARDIS-SN

PROJECT TITLE : Expanding the Integration-Testing Framework and Automation

GSoC Application Tag: integration-testing (TEP001)

Debajyoti Dasgupta

**Potential mentors: Wolfgang Kerzendorf Vytautas Jancauskas,
Ulrich Noebauer**

Basic Information

Contact Information :

Name: Debajyoti Dasgupta

Email: debajyotidasgupta6@gmail.com

University: Indian Institute of Technology Kharagpur

Degree: B.Tech+M.Tech(Dual Degree)

Department: Metallurgy and Materials Engineering

Phone Number: +91 8327406555/9082194157

Github/Gitter chat room: **debajyotidasgupta**

Whatsapp Number: +91 9969101602

Meeting and Discussion with the mentors

1. Easily reachable through Gitter, Email and Github.
2. Available all days from 6:00pm(IST)-1:00am(IST).

CODING SKILLS

PROGRAMMING LANGUAGES AND SKILLS

1. Fluent in Python , C/C++ and moderate knowledge of Cython
2. Well versed with Object-Oriented programming.
3. Good knowledge of Azure and azure pipelines
4. Web development framework:Django and flask
5. Well versed with html, css, jekyll
6. Good knowledge of data structure and algorithms

7. Good knowledge of data science.

Development Environment

1. Ubuntu 18.04 LTS , Python 3.7.1+Anaconda
2. PyCharm Community IDE for python development.
3. Also supports atom/sublime text editors.
4. Familiar with conda and virtual environment.

Version Control System

1. GIT/Github (Preferred) strong concepts
2. Also comfortable with gitlab and mercurial.

Other Skills(relevant)

1. Good background in physics ,chemistry,maths and astronomy.
2. National level olympiad student in all above.

PRE GSoC INVOLVEMENTS

PRE-REQUISITES ACCOMPLISHED

1. Set up the TARDIS-SN environment
2. Created a conda package
<https://gist.github.com/debajyotidasgupta/b288facae47afef26477d980800a3d30>
3. Ran all the example datasets
4. Created azure pipelines and implemented some integration test locally
5. Made few benchmarks with asv module.

6. Made few changes with cFFI to remove cython dependance.

PULL REQUESTS

1. <https://github.com/tardis-sn/tardis/pull/899> : this was the first pull request for the errors in the github pages of the documentation.
2. <https://github.com/tardis-sn/tardis/pull/900> (**merged**):the error in the github pages were corrected and and merged in the gh-pages branch
3. <https://github.com/tardis-sn/tardis/pull/901> (**merged**):proper editing of the issues with the github pages and storing the required files in docs folder
4. <https://github.com/tardis-sn/tardis/pull/903> (**open**):conda package developed for tardis
5. <https://github.com/tardis-sn/tardis/pull/904> (**open**):documentations corrected and docstrings corrected according to numpydoc.
6. <https://github.com/tardis-sn/tardis/pull/907> :travis ci, the main integration system updated to new python versions

PROJECT PROPOSAL INFORMATION

INTRODUCTION:

TARDIS is a scientific tool (more specifically a Monte Carlo radiative transfer code) whose primary goal is the calculation of theoretical spectra for supernovae.

DESCRIPTION:

Testing a scientific code like TARDIS is very important. It need to ensure that the scientific insights TARDIS gain using the code are not based on bugs. Open collaboration with GitHub is great, but the more people work on the code the more opportunities there are to introduce bugs. Making

sure that the code doesn't change or only changes as expected it, is thus an important part of TARDIS development.

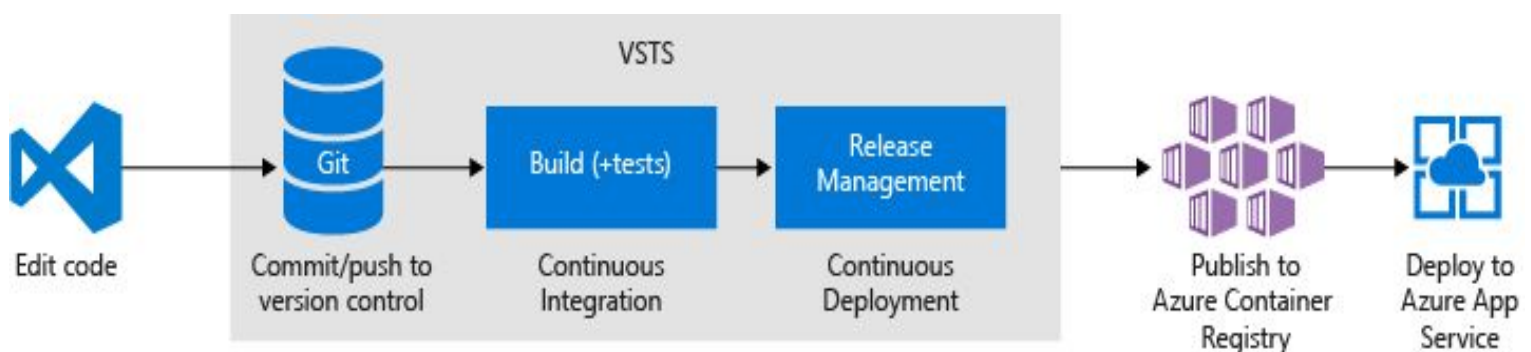
TARDIS has scientific code. While many people working together on various functionalities - there are possibilities for anything to break and go undetected for a long time . It might misbehave at times , detecting it would get very difficult. Testing a scientific code like TARDIS is crucial - the test suite needs to be up-to-date and should have a good code coverage and these tests be AUTOMATED is absolutely necessary.

CURRENT SCENARIO:

1. TARDIS uses pytest as the primary testing framework . Unit tests are maintained inside the tests directories and have tests for individual 'units' (methods).
2. To test the full TARDIS code, there is a single class in tardis/test_tardis_full.py with a simple configuration. The current unit tests are fast enough and are executed within around three to four minutes on Travis-CI.
3. Current code coverage is approximately 54-55%, there are possibilities that some part of code is untested and developers may not be able to comment about its health.
4. The current test case for an integration test of full TARDIS code does not exhaustively probe every functionality of TARDIS and hence does not ensure rigorous testing of the whole code.
5. Complex setups - Stratified W7 setup and AbnTom Setup are also in use.

REQUIRED IMPROVEMENTS AND PRIMARY GOALS:

- Setup the integration tests to run once a week on Azure pipelines
 - **TARDIS** currently has two types of tests: **unit tests** that verify small portions of the code and **full-scale integration tests**. Both of these test types are implemented with a framework. But the integration tests are difficult to use.
 - First task is to implement continuous running of the large **integration tests** in the background weekly once so that the code base remains bug free and does this automatically .
 - This will not only reduce the headache of running big integration tasks on external server, which are very time and space consuming and do all this automatically on azure's user friendly cloud platform
 - This will not only save time and space but also if the user forgets the integration test , azure will still be doing it weekly in the background
 - This will be implemented as per the following flowchart.

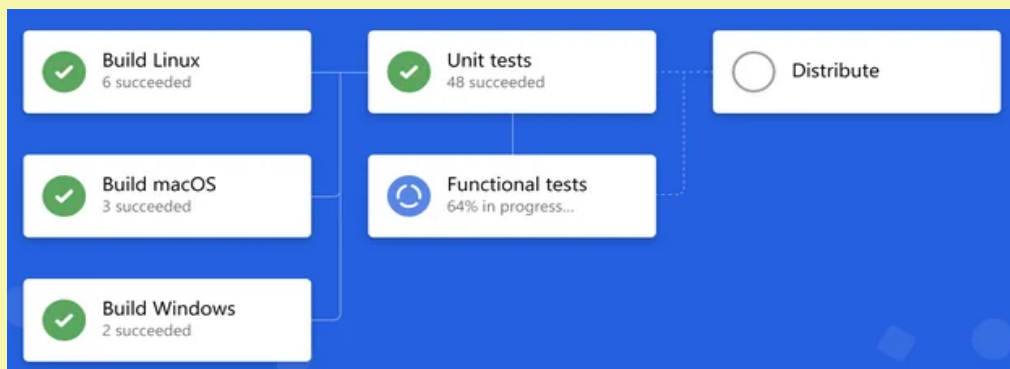


Entire of this task will be performed using yaml files which run both on LINUX as well as MAC with the help of miniconda installation.

```
strategy:
  matrix:
    linux:
      vm_Image: 'Ubuntu-16.04'
      conda: '/usr/share/miniconda'
    mac:
      vm_Image: 'macOS-10.13'
      miniconda.url:
        'http://repo.continuum.io/miniconda/Miniconda2-latest-mac-x86_64.sh'
      maxParallel: 4
```

- Sample tests run when TARDIS is installed on any machine to check integrity.

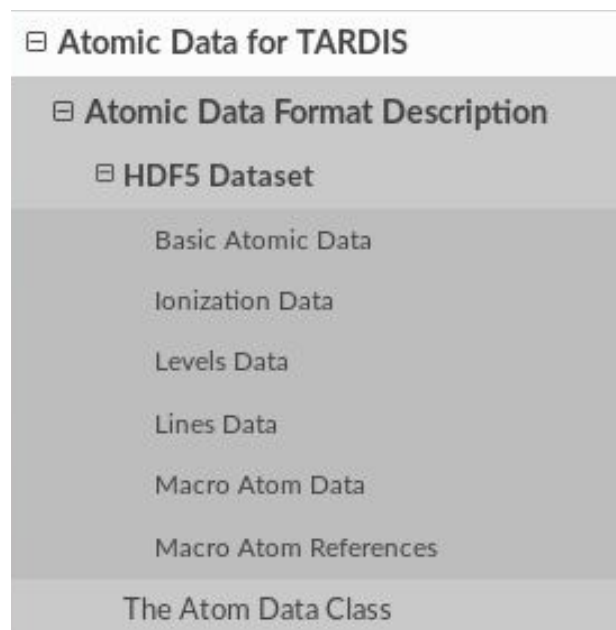
```
bash: |
    source activate tardis
    conda install -y pytest-cov
    pip install
git+https://github.com/tonybaloney/pytest-azurepipelines.git
    pytest tardis --tardis-refdata=$(ref.data.home)
--test-run-title="TARDIS test $(vm_Image)" --cov=tardis
--cov-report=xml --cov-report=html
    displayName: 'TARDIS test'
```



This flow diagram shows the testing process of **azure-pipeline**.

● Expanding the checks

- The above checks will be expanded and check also different properties of our model against reference data (e.g. electron densities, ionization fractions, dilution factors)
- These datasets are present in the folder **hdf5 files** form an integral part of TARDIS and failing there tests implies a big flaw in the program
- Thus the testing system must be expanded to implement the checks these datasets also.



- These are the datasets available on tardis and the code will be tested against these datasets, failing which would mean code with high error
- All these tests will be finally automated using **AZURE PIPELINES**.

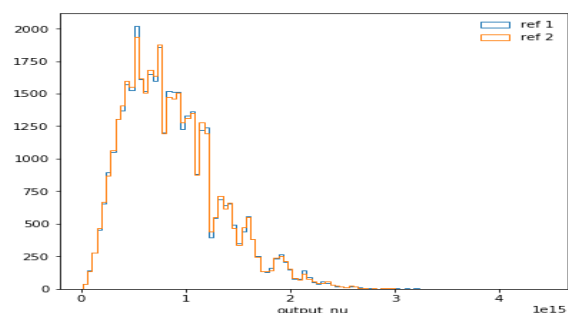
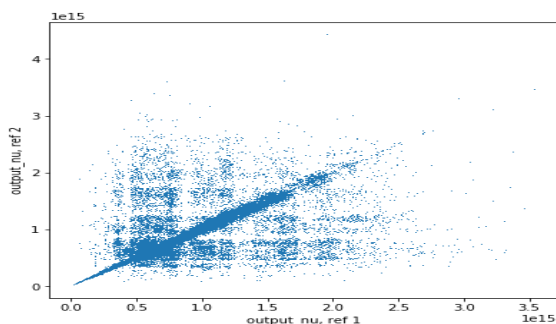
● Improving the Reporting Process

- Hand in hand with expanding the verification process ,the reporting process will also be improved.
- A framework exists but is currently not actively used.
- Improved the reporting process will contain detailed **plots** and **comparison results**.
- Code must look something like

```
class comp(object):
    def __init__(self, ref1_hash=None, ref2_hash=None, comp_path=
'unit_test_data.h5'):
        self.ref1_hash = ref1_hash
        self.ref2_hash = ref2_hash
        self.comp_path = comp_path

    def gen_table(self):
        (hdfs1,hdfs2) = (pd.HDFStore(self.ref1_fname, mode='r'),
pd.HDFStore(self.ref2_fname, mode='r'))
        (keys1,keys2) = (hdfs1.keys(),hdfs2.keys())
        hdfs1.close()
        hdfs2.close()
        (data1,data2) = pd.DataFrame(index=keys1, columns=['exists']),
pd.DataFrame(index=rd2_keys, columns=['exists']))
        data1['exists'] = True
        data2['exists'] = True
        joinedData = data1.join(rd2_df, how='outer', lsuffix='_1',
rsuffix='_2')
        return joinedData
```

- All the automated testing results on **azure-pipelines** will generate the results and return detailed plots , testing reslts and the comparison results and differences.
- After plotting the results will be like



● DOCUMENTATION UPDATED:

- All the sphinx documentation related to testing framework will be updated and new ones will be added
- The github pages will also be updated in accordance.
- All the [numpy documentation guidelines](#) will be followed.

● Wishlist:

This will be implemented strictly after all the primary goals are completed successfully and correctly tested

- Use cFFI to provide an automatic interface for testing the parts in the monte carlo package instead of ctypes.

TIMELINE (PROPOSED):

- I have semester exam from 15th April to 30 th April and willhence remain during this period. However this period is way of the timeline and will thus not affect my schedule.
- I have been working with TARDIS for quite a while now (more than 2 months) and hence am well versed with the working and requirement of the organisation.
- I have all the requirements of Tardis set up as well as running
- As a result I will be setting grounds early to avoid stopgaps and lags(if any) during the actual GSoC period.

- All dates mentioned here are weeks - starting from Monday and ending on Sunday.

Total duration of coding period: 12 weeks: May 27 - August 26.
This will include coding (accomplishment of the primary goal) as well as the documentation keeping track of the work done .

COMMUNITY BONDING PERIODS (6th May - 26th May):

- Familiarize myself with relevant topics from **astronomy** and physics
- Familiarize myself with all the **datasets** available on TARDIS and the extent of the **Integration tests**.
- Discuss and finalize the improvements on the ideas(if any) with the mentors to start working.
- Deploy azure pipelines for my **forked** directory and prepare setups for running integration tests **once a week**.
- Improvement and updation of the integration tests to be successfully running with the newly adopted python 3.6+ .
- At the begging of the coddling period I will be having a basic framework of the integration tests (up and running) and the results returned form it over a 3 week period.

CODING PERIOD:

- **Week 1 : May 27 - June 3:**

Check the test results and working of the integration tests. If everything is working fine all the tests will be automated and the code will be merged into the main project.

MILESTONE ACHIEVED: Primary goal task 1 accomplished

- **Week 2-3 : June 4 - June 19 :**

Expanding the integration test for the datasets and writing the tests to run checks using the datasets. After the tests have been successfully written and checked for the integrity these tests will be run for the code and checked for integrity

- **Week 4 : June 19 - June 28:**

Buffer time to complete the pending work and documentation and submit the documents for **Phase-1 evaluations**.

- **Week 5 -6: June 29 - July 12 :**

The tests of datasets if not completed, will first be completed then this test will be automated using azure pipelines to run once a week.

MILESTONE ACHIEVED : Primary goal task 2 accomplished.

- **Week 7 : July 13 - July 19:**

Write the code for reporting the results in the form of a table using **dokuwiki**, including the improvement of the existing framework and implementing the class compare. The results of the test will be reported and the results analysed.

- **Week 8 : July 20 - July 27:**

Buffer time to complete the pending work and documentation and submit the documents for **Phase-2 evaluations**.

- **Week 9 :July 28- August 4:**

Write the code for plotting the results of the tests of the dataset and example in matplotlib , including the improvement of the existing framework and implementing the class compare. The results of the test will be reported and the results analysed.

- **Week 10 : August 5-August 11:**

Checking the integrity of the code .If any patches they are fixed and then the process will be automated using azure pipelines to run every time a new pull request is made.

MILESTONE ACHIEVED : Primary goal task 3 accomplished.

- **Week 11-12 : August 12-August 26:**

Buffer time to complete the pending work and wrapping up all the tasks and documentation and submit the documents for **Final evaluations**.If time permits implement the wishlist.

MILESTONE ACHIEVED : Primary goal task 3 accomplished.

ABOUT ME:

I am a 19 year old First Year Undergraduate student enrolled in Metallurgy and Materials Engineering (5 year Dual Degree – B.Tech+M.Tech) at IIT Kharagpur,India. I have been largely involved in programming and open source for two years and have build a huge interest in development . I had build my first game called **brainvita** during my class 12th CBSE final project, entirely in C++. Since then I have learned a lot many language and framework,

I have been a huge contributor in open source and have successfully completed many open source contests like [hactoberfest](#) by Digital Ocean and **Kharagpur Winter of Code(KWoC)** organised by KOSS ,IIT Kharagpur.



I have also been an active participant of Kharagpur Open Source Society and have contributed actively.

My major project also includes the one I made for **code.fun.do** hackathon by **Microsoft** for disaster management-

<http://disaster-management.azurewebsites.net/>

WHY ME FOR THE PROJECT:

I have been working for over two months with this organisation and am well versed with the functioning as well as the requirement of the organisation. I have made many contributions, whenever I find some code that can be made better, some documentation that can be improved, some method which needs tests. My contributions can be easily seen at <https://github.com/debajyotidasgupta>.

I am very **punctual** in my work and am very **hardworking** , hence no one will get any chance of complaining that I am lazy to work.I maintain a strict routine which I follow, and **accuracy** and **perfection** are the key terms of my life.Most importantly I love coding a lot , and do all the day ,so I can easily devote **50+ hours** per week for work.I will also maintain blog to update my mentors of the work I am doing.

GSOC

Have you participated previously in GSoC ?Under which project?

No, this is my first participation in GSoC.

Are you also applying for any other project ?

No, I will be entirely committed only to work for TARDIS

Any other commitments?


I have no commitments during the GSoC period as my University's summer holidays start from 1st May to 12th July 2019 and so I will be heartily devoting 50+ hours during this period as I have no pre occupancy.

My classes for the new semester will start from 13th July 2019.Due to veryless academic load during the start of the new semester , i will easily be able to dedicate 37+hours a week during the last week of the coding period.

As a result i will be dedicating 45+ hours to this project and strive hard to complete it successfully.

REFERENCES:

1. [GSoC 2019 IDEAS PAGE](#)

- 
2. [TEP001 - EXPANDING INTEGRATION-TESTING FRAMEWORK](#)
 3. [TARDIS-READ THE DOCS](#)
 4. [TARDIS COVERALL REPORTS](#)
 5. [PYTEST-DOCUMENTATION](#)
 6. [AZURE PIPELINE DOCUMENTATION](#)
 7. [TESTING DOCUMENTATION](#)
 8. [DATASETS](#)

* * * * *