**PIP**

**By Augustine, Arif, Sanjog, Masinde**

**Åbo Akademi University**

**Abstract**

Pip is the your friend when doing any python project. It is availble in all platfroms, i.e windows, linux and mac. It is used in installing different python packages. In this report we will give an overview of this open source project. We will view different stakeholders, the models of the project and we will conclude the way we see it from our findings.

**Introduction**

pip is a recursive acronym that stands for Pip installs Packages. It is a package management system for installing and managing Python software packages. It is an open source software by the Python Packaging Authority (PyPA) and was initially released on 4th April, 2011. The most recent stable release was on 6th November, 2016.

pip is the defacto package manager for software packages developed with Python and ships with most Python distributions. As such its user base is almost as large as the entire Python community of developers. It is notable for the ease of its command-line interface, which simplifies the installation and uninstallation of Python software packages.

**Stakeholders**

Stakeholders are people who have direct impact with the product/project of the organization.

**Over view**

The following bullets explain in brief different stakeholders who are involved in pip.

* **Acquirers**: The Python Packaging Authority is the acquirer of pip.
* **Assessors:** The Python Packaging Authority is responsible for assessing and approving contributions from pip contributors to the source code.
* **Communicators**: The communicators maintain communication with other stakeholders. They analyze and make documentation of the system. The Python Packaging Authority (PyPA) is the communicators of pip.
* **Developers**: PyPA developers and other contributors from Python Developers community.
* **Maintainers**: pip is based on core developers from PyPA and other contributors from Python Developers community. When the system is operational the PyPA takes care of it.
* **Production Engineers:** Production engineers are the core developers.
* **Suppliers:** PIP works on different operating system such as Unix/Linux, macOS, and Windows. These stakeholders supply appropriate software and hardware in which the system runs.
* **Support Staff:** The Python Packaging Authority (PyPA) is a working groupthat supports many of the relevant projects in Python packaging. Besides this support staff for pip also consist of github and bitbucket staff and many other volunteers also.
* **System Administrators:** Python organisation is responsible for the system administration.
* **Testers:** The pip development team is responsible for testing the software to ensure that it meets the requirements.
* **Users:** Python developers

**Stakeholders Power/Interest Grid**

<Todo Sanjog add the grid here>

Figure 1 above visualizes the level of interest and power of the different stakeholders in the pip project. From the figure, PyPA at the top right corner of the grid, as the acquirer and represented by Donald Stufft, has the highest interest and power in the project. The success of the project is of utmost importance to it and it has the final say on important decisions concerning the project. At the top left corner of the grid is PyPA again, this time as the accessor. In this role, it has very little or no interest in the project but has significant power regarding decision making in the direction of the project. At the bottom right corner of the grid are the developers and maintainers, which include PyPA core developers, led by Donald Stufft, and other contributors from the Python developers community, and the users, which include Python developers. These do not have much say in the decision making concerning the project but they have relatively high interest in the success of the project. Lastly, at the bottom left corner of the grid are the communicators, testers, support staff, etc. Although these play significant role in the project’s life cycle as stakeholders, they have relatively low power and interest in the project.

**Development**

The open source project pip is maintained by The Python Packaging Authority (PyPA), and is developed by Python core developer and other community hosted in GitHub. Any contributor can contribute in development of pip through GitHub.

In terms of software development Model-View-Controller (MVC) is a design pattern for software development and Python uses MVC architecture. The concept of MVC is, it split the full system into three interconnected segments, and that are a Model, View, and Controller. The Model part represents the domain, and domain is used for data storing and data retrieving. The View represents to the user and the Controller is used to make communication between the Model and The View.

However, to contribute in pip development the contributor needs to submit a Pull Request against the master branch in GitHub. The contributor must provide an explicit description about the tasks what he/she going to do. The task is tested locally first.

**Automated testing:** all kinds of pull requests and push to ‘master’ are tested in Travis, and Travis is based on .travis.yml file. Jenkins CI is used for certain versions of Python on centos and windows.

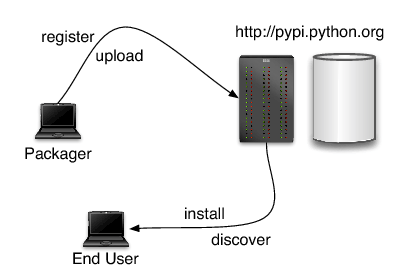
**Tests running:** to run the test an operating system needs to have subversion, git, bazaar, and mercurial. And Python requirements are a virtualenv, tox or pytest, scripttest, and mock.



**Fig. Local testing system**

**Pip usage**

The Python Package Index (PyPI) is a repository of software for the Python. It is the host of different Python packages and their dependencies. At present there are 119685 packages on PyPI. Donald Stufft is a core developer of pip, and PyPI is solely developed and maintained by him.



**Figure: Package management process in PyPI using pip**

Packager can register/upload their own work in PyPI using pip command and other users can browse existing projects. Source can be uploaded to an existing package. Pip uses PyPI as a default host for the packages and their dependencies.

Some important pip commands:

|  |  |
| --- | --- |
| Command | Action |
| Pip install | Install packages |
| Pip uninstall | Uninstall packages |
| Pip freeze | Output installed packages in requirements format |
| Pip list | List installed packages |
| Pip show | Show information about installed packages |
| Pip search | Search PyPI for packages |
| Pip zip | Zip individual packages |
| Pip unzip | Unzip individual packages |
| Pip bundle | Create pybundles |
| Pip help | Show help for commands |

**Architectural structures**

* Modules structures
* Component-and-connector structures
* Allocation structures

**Pip Modules**

The modules shows the units of implementation, or shows what is supposed to be implemented

Decomposition

class

Uses

Layered

**Component-and-connector**

Concurrency

Shared data

Client-Server

Process

**Allocation**

Work Assignment

Deployment

Implementation

**Conclusion**