VISA Communication Tool Technology Review

Chenliang Wang[1], Lucien Armand Tamno[1] and Wenbo Hou[1]

[1]Oregon State University

November 2016

Abstract

This document provide technology review for the new VISA communication Too. Each group member takes responsibility for one major feature of this software. Each section in this paper contains individual research on finding proper technologies to achieve assigned functionalities.

Contents

1 Overview

After specify the software requirement, the develop team starts to assign jobs to each group member. Since the software has three main components, each team member takes responsibility for one component. Before designing this software, developers ought to skim all features of this software and find corresponding technologies to achieve them. The following three sections show individual researches on from team members.

2 Command Auto-Completion & Documentation — Wenbo Hou

2.1 Introduction

The command completion and the documentation are two important parts of this programmatic software because users need them to learn how to use this software. Furthermore, they can help users to dig deeper to fix bugs on instruments. In the Software Requirement Specification, the develop team summarized specific requirements in this part. First, the software provides the syntax auto-completion feature to users. Second, users query commands with different keywords, which save users' time on searching commands. The last related function is the command reference. The client requires a well-prepared library for commands so that users can use this software effectively.

In this section, the develop team will discuss technologies used to achieve above goals including, the IDE, the program, and support materials, etc. IDE and support material are two primary technologies in this assigned part so that the developer analyze them first.

2.2 Integrated Development Environment(IDE)

An IDE refers to a software suite that contains all necessary tools for developer to write and test a software [1]. A good IDE allows developers to fully control programs so that they can optimize software easily. Based on researches, there are lots of IDEs for Python, the programming language required by the client. The developer chooses three most popular IDEs to analyze and find the most suitable one. The developer will apply the chosen IDE on all three assigned functionalities.

2.2.1 VIM

VIM is the most advanced text editor program for Unix. It is lightweight and provides lots of keyboard shortcuts for operations. As a result, the programmer can get rid of the mouse and save much time [2]. However, VIM does not provide any language support. In other words, developers need to spend lots of time on initial configuration.

Also, no debugger exists in VIM. developers have to access GDB, the Linux built-in debugger, to debug programs. Based on my experience, GDB is not user-friendly. Developers need to manually go through the whole program, which wastes much time.

2.2.2 PyCharm

PyCharm is another IDE built for professional developers[2]. According to the information from JetBrains' website, PyCharm can handle the whole develop routine and provide keyboard-centric approaches to the majority of functionalities[3]. Besides that, the code completion and instant syntax checking improve the coding efficiency significantly.

PyCharm integrates powerful developer tools and scientific tools. For example, the well-known Jupyter Notebook in PyCharm allows developers to combine code execution, rich text, mathematics, plots and rich media [3]. The built-in debugger and test runner simplify the test procedure and then improve the work efficiency. The version control in PyCharm is another bonus.

PyCharm also offered education edition to college students, which saves students lots of money.

2.2.3 Pydev with Eclipse

Pydev is a Python IDE for Eclipse. Like PyCharm [1]. Pydev also provides code completion and error checking service. TODO Tasks is an excellent feature in Pydev [4]. With it, developers can manage their time effectively. The

debugger in Pydev is as powerful as PyCharm's debugger. The code coverage associated with the debugger makes the test plan comprehensive and efficient.

As an open source software, Pydev is free to use and has a strong community support. Developers can get a lot of plugins from other developers. However, Pydev does not have a professional maintenance team. If bugs existed in the software, no one could fix them in time. Also, users cannot work with new features of a programming language without timely updates.

2.2.4 Evaluation

The developer first removes VIM from candidate IDE list. VIM has limited code support and provides little debugging supports. Additionally, Unix is not the operating system with that the developer works. Thus, the developer needs to make a choice between PyCharm and Pydev.

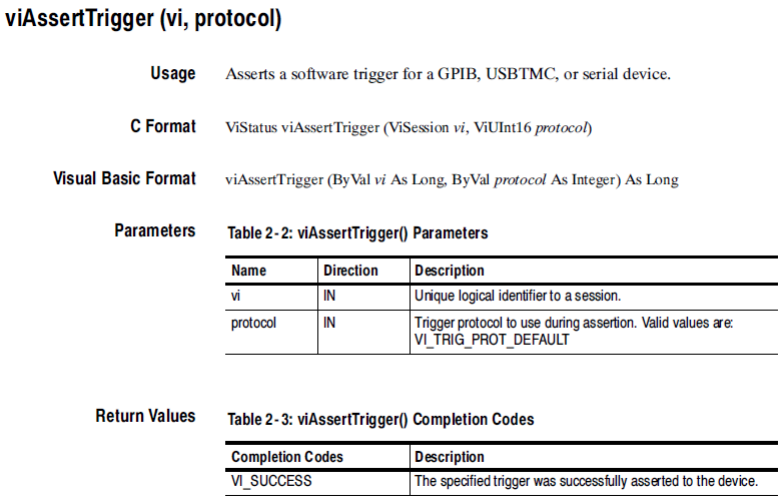
Recalling the Gantt Chart, the developing period lasts three months and tons of tests. Consequently, the version control and the test management are essential features of chosen IDE. Therefore, PyCharm becomes the first choice. Engineers from Tektronix also recommended PyCharm to the developer because of PyCharm's Integrated tools and suitable configuration. Consequently, the developer applies PyCharm for his functionalities development.

2.3 Support Material

A well-organized library is necessary for the command documentation and the syntax reminder. In last video conference, the developer discussed this issue with engineers from Tektronix. Fortunately, Tektronix provides Connectivity Software Programmer Manual to the public. They suggested the developer took advantages of this manual. Accordingly, three options exist here.

2.3.1 Provided Material from the client

The programmer manual from Tektronix is well-organized. This guild introduces the inner structure of the existing software and descriptions of built-in commands. The detailed description including parameter types, usage of the command and returned values. By reading this, users can operate the software efficiently. However, the page layout is not standardized. In other words, the program cannot read and display it accurately.



viAssertTrigger (vi, protocol)

Usage Asserts a software trigger for a GPIB, USBTMC, or serial device.

C Format ViStatus viAssertTrigger (ViSession *vi*, ViUInt16 *protocol*)

Visual Basic Format viAssertTrigger (ByVal *vi* As Long, ByVal *protocol* As Integer) As Long

Parameters

Table 2-2: viAssertTrigger() Parameters

Name	Direction	Description
vi	IN	Unique logical identifier to a session.
protocol	IN	Trigger protocol to use during assertion. Valid values are: VI_TRIG_PROT_DEFAULT

Return Values

Table 2-3: viAssertTrigger() Completion Codes

Completion Codes	Description
VI_SUCCESS	The specified trigger was successfully asserted to the device.

Figure 1: Sample from TexVISA programmer Manual [5]

2.3.2 Written Material from programmers

Since the develop team needs to implement VISA API, the developer can collect API written by other teammates to generate the command library. However, it takes a long time to create the reference material for all commands. Furthermore, the developer may not fully understand generated commands because the developer just invokes API from the built VISA library. The reference material created by the develop team is not clear enough for users to understand well.

2.3.3 Improved material from the client

The developer can acquire information from the programmer manual and reorganize it based on programs requirements. Therefore, written programs can read information accurately. However, this process takes may efforts. The developer needs to retrieve and proofread information from the manual. The last challenge is the copyright. The developer needs to get permission from Tektronix.

2.3.4 Evaluation

After discussing the pros and cons of each option, the developer decides to choose the third one, modifying provided manual from Tektronix. A proper documentation should provide a correct and detailed guide to users. So, the developer generated manual should not exist. If the developer implements the current handbook from Tektronix, he should adjust his read-file program based on different page layouts. This fact requires tons of symbol recognition works, which is hard to achieve. The third one collects all pros from previous solutions. A standardized library containing professional programmer guides is the most suitable one for this project. The developers can generated a regular file reading program for all three assigned functionalities, which reduces the code redundancy and improve the display accuracy.

2.4 Specific Technology Related to Command Completion

One of the requirements from the client is autocompleting commands being typed. Currently, Almost all IDE provides this feature to speed up the process of coding applications. Through exploring several IDE, the developer found that only one technology could achieve this feature that was Intelligent code completion, as known as IntelliSense [6]. This technology works with an automatically generated library with variable names and functions defined before. When users are typing, this technology will suggest matches functions or variables based on the input in a pop-up window. IntelliSense also provides a short description for a suggested function in the pop-up window. The developer will explore similar projects to get a rough design for python based IntelliSense.

2.5 Technologies Related to Query Doc

Querying commands by keywords is another form of searching. Technologies related to searching are information carriers and searching algorithms.

2.5.1 The Information Carrier

The information carrier refers to the file formation or the data structure to store information. Considering the project requirements, the developer picks three different file formats: **linked list**, **hash table**, and **Comma Separated Values** files.

The **linked list** is a linear collection of data elements, called nodes. Each node has pointers to its neighbors and stored information. In this scenario, each node contains function keywords and the function description.

The **hash table** is a data structure used to implement the associative array. With it, programmers can map keys to values. In this project, the developer can associate function descriptions with different keywords.

Comma Separated Values (CSV) files stores tabular data in plain text [7]. Each row in this file is a data record, as a row in a database table. Programmers can customize fields in each row to store different information.

```
"Title 1","Title 2","Title 3"
1,"a",08/18/07
2,"b",08/19/07
3,"c",08/20/07
4,"d",08/21/07
5,"e",08/22/07
6,"f",08/23/07
7,"g",08/24/07
8,"h",08/25/07
9,"i",08/26/07
```

Figure 2: CSV format [5]

Because users can query with various keywords, a node in the linked list may not contain enough keywords. Because the linked list and Hash table are pointer based, the developer can easily mass up memory space and create address conflicts. Sometimes, inputted keywords are not consistent with predefined keywords, so that the program needs to go through all information of each function. Such a program requires the developer consider various conditions, which is hard to manage. Furthermore, the time cost of this program is extremely high. With CSV files, the developer has no need to take care of pointers, which makes programming easier and more reliable. The developer can quickly search through the entail row with the keyword to find target function.

2.5.2 The searching algorithm

The searching algorithm is another technology in this functionality. Considering potential file format/ data structure, the developer prepare three searching algorithms: the **binary search**, the **hash table search**, and **SQL**.

Based on the ASCII stands, each character associates with a decimal value. In the ASCII sorted data holder, the developer store data based on the decimal value of each character. So that, the developer can apply binary search to check each character in the inputted keyword to match the keyword. The time complexity of this algorithm is high because the developer needs to iterate binary search for all strings. Despite that, the developer has to spend additional time on writing and debugging the new program.

Another searching algorithm is associated with the hash table. Due to the inefficiency of hash table mentioned in the previous section, the developer decides not to use it. It is not necessary to introduce this algorithm in this document.

The last searching methods is **Structure Query Language** (SQL). SQL is a programming language designed to manage a relational database. Fortunately, Python has a built-in SQL library, querycsv,py for querying CSV files. SQL allows programmers to find data record by keywords.

Considering the time cost and accuracy of a self-generated binary searching program, the developer decides to implement the built-in SQL library to search through the CSV file, the chosen file format in this project.

2.6 Technologies Related to Device Identification

When users apply VCT to interact with instruments, it is necessary to know the connected device type. Technologies related to this feature are how to identify the device and how to show the connected device type.

2.6.1 Device Identification

Three options are: **Manually Input**, **Built-in IDN request**, **Driver recognition**.

After users connect a instrument to the computer, the software can pop up a window to ask users for the device configuration including, its type, IP address, etc. Then, the software stores the configuration into a runtime database. When users select a device from the device list, the software can show what type the device is.

VCT can also apply commands from PyVISA, the built-in VISA library, to identify the device. Since each device associates with a unique IP address, the software will set up a trigger function to ask the IP address for the device ID number. Then, the software can link the ID number with corresponding device type and display it on the user interface.

```
>>> import visa
>>> rm = visa.ResourceManager()
>>> rm.list_resources()
('ASRL1::INSTR', 'ASRL2::INSTR', 'GPIB0::12::INSTR')
>>> inst = rm.open_resource('GPIB0::12::INSTR')
>>> print(inst.query("*IDN?"))
```

Figure 3: Identify Device with built-in commands [8]

According to the information from Margaret Rouge, the device driver refers to the control program for a particular type of device attached to the computer. Consequently, the developer can track the active status of related device drivers to identify target device.

The first option requires is not time efficiency because of the involvement of the human-computer interaction. Also, it has a high probability making mistakes. The second option is speedy and accurate but requires a massive matching work. The third option is also quick and precise but requires users to touch the operating system, which is not security. Also, the system permission is another big issue. Considering the time cost and the usability, the developer chooses the second solution for this functionality.

2.6.2 Distinguish Commands for Different Device Types

Three options are : **Color coding**,**customized text cursor**, **Tag information**

Another technology involved is how to show to which category the connected device belongs. Due to the limitation of the interaction method, the developer can only distinguish commands visually.

The simplest way to distinguish commands from different sets is to use different font color on the command window. When users select the target device, the software will change the font color for the following input. As long as users memorize the meaning of different colors, they can easily keep tracking the target device type.

The developer got inspiration from Google doc. Google doc allows people works on the same document at the same time. For each contributor, Google doc will generate a different text cursor with their user names. Similarly, the developer can customize the default text cursor with different device tags.

The last method is showing connected device type in the tag of the window. When a user switches to another instrument, the tag will refresh and show the device type.

In the above three options, the third one is easier to accomplish than the other two. The developer only needs to refresh the user interface to achieve that. However, it is not striking enough, and users can miss it easily. The left two options are equivalent, but the developer decides to choose the color coding because of two reasons. First, the floating bar of a cursor could cover the drop down list of IntelliSense. Second, an oversize floating bar will have

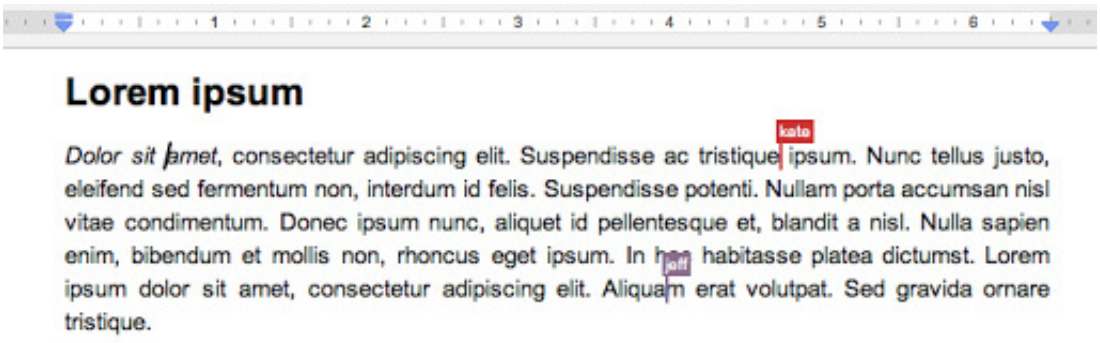


Figure 4: Different cursors in Google doc [9]

adverse effects on the user interface's aesthetic.

3 User Interface Development — Chenliang Wang

3.1 Introduction

User Interface Development is very important for a software. A good user interface will improve the level of the software. Whatever the functions is, it is also a bad software without a friendly user interface. In this section, it includes 3 pieces of project, user interface, keyboard shortcuts, and functional test.

3.2 User Interface

3.2.1 Option

There are 3 option for create UI for this project. Pycharm with pyqt, Visual Studio with IronPython, and Eclipse with PyDev.

3.2.2 Goals for use in design

User Interface for a software is like the face for people. Also, it will affect the function for this software. All in all, the user interface is not only need friendly, but also need practical such as good navigation bar, professional layout, and etc. As this is so important, we need more focus on how to create a user interface. There are three options for create UI with different IDE.

3.2.3 Criteria being evaluated

Pycharm is a IDE specifically for python. It can be used to coding, testing and debugging. This is one of the best IDE for python. In order to create UI, we can add pyqt, and it will be work well with Pycharm. If we use Pycharm, it will be no any problem of compatibility for the python.

Visual Studio is a IDE from Microsoft. It is not focus on python, be it can be used to create UI with IronPython. Visual Studio is not a IDE specifically for python, but it is a great software and it also will work well on python.

Eclipse is also an option to create UI. Eclipse is a IDE with almost people who work on Java will use. It more focus on Java but it also can be great on Python too. To create UI, we can use Eclipse with PyDev.

3.2.4 Table comparing based on criteria

Table 1: Different UI design tools

-	Cost	Focus	Compatibility
Pycharm with pyqt	Almost same	Python	Very good
Visual Studio with IronPython	Almost same	C, C++	good
Eclipse with PyDev	Almost same	Java	good

3.2.5 Discussion

Compare these three options, both of them are good for python of UI. As Pycharm is focus on python, so it will better for compatibility than other two. Also, it can find more resource on UI create. For the cost, there 3 options

are almost same. They focus on different languages.

3.2.6 Best option based on criteria

For the User Interface, we will choice use Pycharm with pyqy to do this now. It should be the best choice in these 3 options. For other 2 options, these two are not bad option, these are just not better than Pycharm as Pycharm is focus on python. We maybe change our choice if we have some new idea, but the first choice right now is Pycharm.

3.3 Keyboard Shortcuts

3.3.1 Option

For these part, we have 3 options for design. Shortcuts with VCT, custom shortcuts, and both of them.

3.3.2 Goals for use in design

Keyboard shortcuts is very useful for a VCT. It can make users to use this tool easily. From the shortcuts, users can save time to find the function what they want to do in the navigation bar. The shortcuts with VCT means the shortcuts created during the VCT design. It can be use once the VCT development complete. The custom shortcuts are created by users' selves. They can create what they want without system conflict.

3.3.3 Criteria being evaluated

Shortcuts with VCT is almost software's choice. It will the low cost, but low payoff as well. The risk of this option is also low as it is created during the VCT development.

Custom Shortcuts is another option for keyboard shortcuts. This option has low cost but higher payoff than shortcuts with VCT. However, as the shortcuts can be custom by users, it has higher risk that it probably conflicts with system for custom the shortcuts.

The third option is the both option 1 and 2 together. In this way, we can keep the advantage of two options. The cost of this option is a litter bit higher, but the payoff is much high as users can use shortcuts in the VCT or custom by themselves. This way can be suitable for more range of users. The risk of this option have same problem with only custom shortcuts, but the payoff is the highest one in these 3 options.

3.3.4 Table comparing based on criteria

Table 2: Different shortcuts

-	Cost	Payoff	Risk
Shortcuts with VCT	Low	Low	Low
Custom shortcuts	Low	Middle	Middle
Both	Middle	High	Middle

3.3.5 Discussion

For these 3 options, the Shortcuts with VCT and Custom shortcuts are likely same cost as only do one part. The third option is the combine first two options. It will cost more as it has more work to do. For the payoff, Shortcuts with VCT is the lowest one as the shortcuts is static that users can't change it. Custom shortcuts has higher payoff than first one, as users can custom by themselves but it has some risk that the shortcuts will be conflicted. For the combine option have both advantage and disadvantage of two options before, but it has high payoff. This option gives users more choice. Users can choose whether create new shortcuts or not.

3.3.6 Best option based on criteria

For the shortcuts, we will choose third option that combine shortcuts with VCT and custom shortcuts. Even if it has some risk for conflict, but this option still the best one.

3.4 Functional Test

3.4.1 Option

There are 3 option for functional test. These are analytical evaluation, empirical evaluation, and unit test.

3.4.2 Goals for use in design

These three options are for functional test of user interface development. Test is very important, but it will be missed by people sometimes. The test can be make sure the task is good or not, and it can find out some problem then fix it, in order to improve the software.

3.4.3 Criteria being evaluated

The first option for test is analytical evaluation. Analytical evaluation is a process of think by our head. It is test with our selves. We need list some question and task to think about all the functions. It same as a self-test. During the test, we need think about all the possible outcome and all the possible of problems. We will think self as a user to use this software and what will be happen. This option will have few cost and the speed could be fast as it is a self-test. The result of this test will be normal as we can't think all kinds of aspect that maybe lost some problem.

The second option for test is empirical evaluation. This is need test by other people. We also need list some task and find some other people to do the task in order to find the problem. For this option, the cost will be higher than analytical evaluation and the speed will be lower as we need find other people and do interview to get the test result. The result of this will be great, we can find more different people to test more than once and improve this software better.

The third option is unit test. We can make whole software to many kinds of unit and test it. In this option, the cost will be higher and speed will be decrease as it need to coding and test all of them. The result of this is great as we can find almost problems.

3.4.4 Table comparing based on criteria

.

Table 3: Different Test

-	Cost	Speed	Risk
Analytical Evaluation	Low	Fast	Normal
Empirical Evaluation	Middle	Middle	Good
Unit Test	High	Low	Good

3.4.5 Discussion

These three options have 3 different cost, speed, and result. Analytically evaluation and empirical evaluation will more focus on UI design, and the unit test will focus on the code.

3.4.6 Best option based on criteria

Test is very important part for development. As this reason, we choose all of these 3 options. In other words, we will do the functional test through analytical evaluation, empirical evaluation, and unit test. We will try our best to test, find out the problem, and improve our VCT.

4 Communication Interface — Lucien Tamno

4.1 Introduction

The communication interface for the VCT software is the channel through which information should move back and forth between the GUI (graphic user interface) and the connected device. In other words, it is the pipe or hose for communication to take place. Written in python language, this set of subroutines also known as functions or methods in addition to properties at the subsequent modules, is broken down into three main functionalities:

- The communication interface invoking VISA API
- the communication interface managing Data
- The communication interface Testing the availability of the connected device and effective transmission.

4.2 Invocation of the VISA API

4.2.1 Generalities

This functionality has the role of call up several libraries from any operating system (OS) in which the VCT application is installed to run. Meaning that, the communication interface is looking up unto variety of functions able to handle streams of data with responsibility to match them to the appropriate PCI (peripheral component interconnected)/PXI bus of the attached device.

In fact, communication VISA API invocation interface is the dwelling place of all sort of communications between the OS and the set of bus systems (PCI, ISA, AGP BUS and enhanced versions like PCIe/PXIe but also RS323, USB)in that Operating system. These communication methods are likely to trigger inputs /outputs By capturing the incoming streams form the UI that they send down to system circuits. the communication VISA API must preserve the integrity of the streams essentially comply with one of the quality of the VCT software which is the reliability.

4.2.2 Description

Yet, this shared boundary of the VCT software called communication interface in its functionality of invocation of API to handle data, does not operate in isolated environment, for interact not only with certain parts of OS but also internally with the UI (user interface). That said, the interface must consider the exchange with others system components. This can be done by making sure the hosting OS has enough memory to store the calling functions. This API has the obligation to then test the availability, the conformance of the system by calling as precondition, all sort of VISA DLL libraries to the job. Reason why python seems to be suitable to the VISA Communication Tool project. In fact, the VCT software aiming measurement of devices, python represents a high abstraction level [2], which perfectly matches the abstraction level of measurement.

4.3 Data Management

Management data has to keep data unchecked from the GUI to any local bus (16-bits for ISA, [32-bits/64-bits] for the PCI/PXI and extended versions, USB) in the case of close connection but also to (RJ45, RJ11, Wireless) connectors in case of network connection. That said, the data management functionality is just about encapsulation of raw information coming from the GUI to become, electric signal traveling thru buses by using communication protocols.

However, the process of data processing must avoid any mutation of received or sent data. Because the VCT built to seek reliability of its information. And that reliability can be reached if only if the VCT features the implementation of the test functionality.

4.4 Functionality-Test

The programming of measurement instruments can quickly turn into a painful exercise because it requires effective understanding of: instruments being tested, protocols carrying out communications, as well as bus systems embedded on motherboards of each device, all this without putting aside the programming language being used for the coding. It is the inextricable reason why, testing implementation functionality must be done at any stage of the VCT software, specially at the communication interface which i recall is the backbone f the VCT software.

Testing at the communication interface is nothing else than the process of making ensure, not only the right information goes to the right place, but also that layers of communication yield and get data with the most appropriate format they were intended to handle. indeed, the functionality test will also consist of triggering messages back to the user to either alert him/her that the device is correctly connected, or the right configuration is in place, or inversely the device is not reachable,unavailable,or wrongly connected, or ultimately the device is not the right one.

Given the fact that VCT communication interface will implement modules to interact with each driver of the peripheral measurement device, it is important then to test the efficiency of each of the functions embedded in those modules by using the pair (VISA, Python)presents very interesting features in terms of measurement control.

4.4.1 Testings

Testing requires multiple combinations of choosing tools, as well as how to effectively use those tools.the expected results are playing an important role in the choice in sense that they provide to the user the processed data to use as criterion of judgment.

1. **Good coverage**

   This criterion of the software is to determine whether or not a software has erors or flaws in it.The ideal percentage of coverage of 100% is set to be a value that defines the status of software free of errors.However, let us mention that 100% is rarely achievable in most of circumstances.

   - Tools:
     - Coverage.py
       the choice of the programming language being Python in this project, one of the best tool monitoring the coverage on the market these days is Coverage.py version 4.2 released on July 26th 2016. And some screenshots give us some idea how to use this new tool.

```
$ coverage run my_program.py arg1 arg2
blah blah ..your program's output.. blah blah
```

Figure 5: command "my.Program" on coverage.py

Use `coverage report` to report on the results:

```
$ coverage report -m
Name                      Stmts   Miss  Cover   Missing
-------------------------------------------------------
my_program.py                20      4    80%   33-35, 39
my_other_module.py           56      6    89%   17-23
-------------------------------------------------------
TOTAL                        76     10    87%
```

Figure 6: command "Report" on coverage.py

- figleaf
  figleaf can be used as tool of replacement of the coverage.py, which supports more features. one thing to notice here is that though both are free code source software, figleaf has some cool features built in like labeling lines rendering the software developer job easier to manage.

2. Random Unit Testing

random testing provides a granular testing on each of the compound of the communication interface, mainly at the function level.The random unit testing is the robust method to check whether or not each function code is well written. to meet the VCT project requirement which is for the software to run on any platform we will use the following testers:

- **Fusil: Taxonomy Tool for Testing**
  effective tool to fuzz(run random testing) with. It uses (limit memory, environment variables, redirect stdout, etc.).
- **Taof: Taxonomy Tool for Testing**
  the Taof manages is suitable to test programs written in python and addtion to that, it is GUI tool and designed to minimize set up time during random test.

References

[1] M. Rouse, "What is integrated development environment (IDE)," in *TechTarget*, 2016. [Online]. Available: http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment. Accessed: Nov. 10, 2016.

[2] S. Arora, "Python IDE: The10 Best IDEs for Python Programmers," in *noeticforce*, 2016. [Online]. Available: http://noeticforce.com/best-python-ide-for-programmers-windows-and-mac. Accessed: Nov. 11, 2016.

[3] Jet Barins, "PyCharm," in *jetbrains*, 2016. [Online]. Available: https://www.jetbrains.com/pycharm/. Accessed: Nov. 11, 2016.

[4] Pydev, "pydev," in *pydev*, 2016. [Online]. Available: http://www.pydev.org/. Accessed: Nov. 12, 2016.

[5] *Programmer Manual*, 1st ed., TekTronix, Inc., Beaverton, OR, 2009, pp. 5-6.

[6] M. Bruch, "Eclipse Code Recommenders," in *code-recommenders.blogspot.com*, 2010. [Online]. Available: http://code-recommenders.blogspot.com/2010/05/its-all-about-intelligent-code.html Accessed: Nov. 12, 2016.

[7] PyMOTW, "csv – Comma-separated value files," in *pymotw.com*, 2016. [Online]. Available: https://pymotw.com/2/csv/. Accessed: Nov. 13, 2016.

[8] PyVISA, "PyVISA: Control your instruments with Python," in *://pyvisa.readthedocs.io/* [Online]. Available: http://pyvisa.readthedocs.io/en/stable/. Accessed: Nov. 14, 2016.

[9] Google Drive team, "What's different about the new Google Docs?," in *Google Drive Blog*, 2010. [Online]. Available: https://drive.googleblog.com/2010/05/whats-different-about-new-google-docs.html. Accessed: Nov. 14, 2016.

[10] Development, "Comparison of Python IDEs for Development | Python Central", Python Central, 2016. [Online]. Available: http://pythoncentral.io/comparison-of-python-ides-development/. [Accessed: 15- Nov- 2016].

[11] "Unit Testing with Python", Dr. Dobb's, 2016. [Online]. Available: http://www.drdobbs.com/testing/unit-testing-with-python/240165163. [Accessed: 15- Nov- 2016].