VISA Communication Tool Software Design Document

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

[1]Oregon State University

December 1, 2016

Abstract

The content of this document describes the organization for the software design description pertaining to VISA communication tool (VCT) as can be shared, used to convey designed information to its stakeholders. This software design description (SDD) gives a clear picture on how VCT will be implemented by presenting designs from particular viewpoints.

Contents

1 Introduction **3**

1.1 Scope . . . 3

1.2 Purpose . . . 3

1.3 Context . . . 3

1.4 Summary . . . 3

2 References **4**

3 Glossary **5**

4 Stakeholders and Design Concern **6**

5 Design views of requirements assigned to Wenbo Hou **6**

5.1 Creating the Documentation——Information Viewpoint . . . 6

5.1.1 Design Description . . . 6

5.1.2 Key Elements . . . 6

5.1.3 Design Diagram . . . 7

5.2 Querying Documentation with a keyword——Interaction Viewpoint . . . 8

5.2.1 Design Description . . . 8

5.2.2 Key Elements . . . 8

5.2.3 Design Diagram . . . 8

5.3 Code Auto-completion——Context Viewpoint . . . 8

5.3.1 Design Description . . . 8

5.3.2 Key Elements . . . 9

5.3.3 Design Diagram . . . 10

5.4 Device Type Identification——Interaction Viewpoint . . . 11

5.4.1 Design Description . . . 11

5.4.2 Key Elements . . . 11

5.4.3 Design Diagram . . . 11

6 Design views of requirements assigned to Chenliang Wang **13**

6.1 VCT GUI Design——Structure Viewpoint . . . 13

6.1.1 Design Description . . . 13

6.1.2 Design Elements . . . 13
6.1.3 Design GUI . . . 13
6.2 VCT UI Framework——Patterns Viewpoint . . . 14
6.2.1 Design Description . . . 14
6.2.2 Design Elements . . . 15
6.2.3 Code example of Pyqt . . . 15
6.3 Function Testing——Structure Viewpoint . . . 16
6.3.1 Design Description . . . 16
6.3.2 Design Elements . . . 16
6.3.3 Code example of unit testing . . . 17

7 Design views of Requirements Assigned to Lucien Tamno 18

7.1 **Communication Interface-VISA API INVOCATION————-Logical Design Viewpoint** . . . 18
7.1.1 Viewpoint Description . . . 18
7.1.2 Key Components and Utilization . . . 18
7.1.3 Design diagram Associate . . . 19
7.1.4 Testing Techniques . . . 19
7.2 **Communication Interface-Data Management————-Dependency Design Viewpoint** . . . 19
7.2.1 Viewpoint Description . . . 19
7.2.2 Interconnection Data Management . . . 19
7.2.3 Design diagram Associate . . . 19
7.3 **Communication Interface-Connection with Hardware————-Interaction Design Viewpoint** . . . 19
7.3.1 Viewpoint Description . . . 19
7.3.2 Key Components and testing techniques . . . 19
7.3.3 Design diagram Associate . . . 19

1 Introduction

1.1 Scope

After getting started on the technology review which is the introductory part of the current designed document, the bulk of work performed in this document by the develop team is to elaborate on each viewpoint of the VCT software. The process will consist of designing each part of the VCT after the guidelines required by the summary design viewpoints table on page 13 of the IEEE Std 1016^{TM}-2009 document.In this current document, each entry from the IEEE's table will be matching each section of this design document.

1.2 Purpose

This VCT design document aims to provide stakeholders enough information on what VCT is like, how the software is implemented, the framework of its implementation, resources, as well as momentum among the various parts of the system.

1.3 Context

This document contains all necessary information related to the design of VCT. The developing team will identify stakeholders and their concerns in the beginning of this document. Then, developers will show their designs for specific requirements from a proper viewpoint. Stakeholders' concerns will be addressed in each design view.

1.4 Summary

In this document, all three team members show designs for requirements they chose in the technology review. The analysis of stakeholders and technologies review for chose requirements help developers to create a basic design for the software. Since developers follow the guideline from IEEE 1016-2009 to put all necessary information into this document, it could be a good starting point for the following development cycle.

2 References

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

[2] PyMOTW, "csv – Comma-separated value files," in *pymotw.com*, 2016. [Online]. Available: `https://pymotw.com/2/csv/`. Accessed: Nov. 30, 2016.

[3] M. Bruch, "Eclipse Code Recommenders," in *code-recommenders.blogspot.com*, 2010. [Online]. Available: `http://code-recomm` Accessed: Nov. 12, 2016.

[4] PyVISA, "PyVISA: Control your instruments with Python," in *://pyvisa.readthedocs.io/* [Online]. Available: `http://pyvisa.readthedocs.io/en/stable/`. Accessed: Nov. 30, 2016.

[5] UMLet, "UMLet: Effective free tool for design UML diagrams," *UMLet* [Online-link]. Available: `http://www.predictiveanal` Accessed: Dec. , 2016: 5:25AM

[6] "Introduction to Balsamiq Mockups 3 - Balsamiq Documentation", Docs.balsamiq.com, 2016. [Online]. Available: https://docs.balsamiq.com/desktop/intro/. [Accessed: 02- Dec- 2016].

[7] "Balsamiq Mockups 3 Application Overview", YouTube, 2016. [Online]. Available: https://www.youtube.com/watch?v=6BxL5 [Accessed: 02- Dec- 2016].

[8] "Tutorials - Balsamiq Support Portal", Support.balsamiq.com, 2016. [Online]. Available: https://support.balsamiq.com/tutoria [Accessed: 02- Dec- 2016].

[9] "Getting started with PyCharm and Qt 4 Designer", Random Codes - Elementz Tech Blog, 2016. [Online]. Available: https://elementztechblog.wordpress.com/2015/04/14/getting-started-with-pycharm-and-qt-4-designer/. [Accessed: 02- Dec- 2016].

[10] "Intro/basic GUI - PyQt with Python GUI Programming tutorial", YouTube, 2016. [Online]. Available: https://www.youtube.com/watch?v=JBME1ZyHiP8. [Accessed: 02- Dec- 2016].

[11] S. Online, "Create a Basic GUI Using PyQt - Safari Blog", Safaribooksonline.com, 2016. [Online]. Available: https://www.safaribooksonline.com/blog/2014/01/22/create-basic-gui-using-pyqt/. [Accessed: 02- Dec- 2016].

[12] "26.4. unittest — Unit testing framework — Python 3.5.2 documentation", Docs.python.org, 2016. [Online]. Available: https://docs.python.org/3/library/unittest.html. [Accessed: 02- Dec- 2016].

[13] "Eclipse/pydev, Pydev with Eclipse - Best Python IDE Available [online-link] `http://noeticforce.com/best-python-ide-f` Accessed:Dec.1.2016:8:30AM.

3 Glossary

- **Tektronix**: The company that designs and sells test & measurement devices.
- **VISA**: Virtual Instrument Software Architecture,a popular I/O API between devices and PC in the test and measurement industry.
- **Packet**: Data carrier in Internet which contains source and destination network addresses, error detection codes, and sequencing information.
- **CSV file**: A comma-separated values file store tubular data.
- **querycsv.py**: A open source CSV query program.
- **Intellisense**: A open source code auto-completion program from Microsoft.
- **API**: Application Program Interface.
- **PyVISA.py**: A VISA communication API library built with Python.
- **VCT**: VISA Communication Tool.
- **Balsamiq Mockups**: A software for design user interface.
- **PCI**: peripheral component Interface.
- **PCI(e)**: peripheral component Interface Express.
- **PXI**: peripheral component Interface eXtensions.
- **PXI(e)**: peripheral component Interface eXtensions Express.
- **NI**: National Instruments.
- **ISA**: Instruction Set Architecture.
- **USB**:Universal Bus serial.
- **GPIB**: General purpose Interface Bus.
- **COM**: Communication Interface.
- **OSI**: Open Systems Interconnection.
- **ISA**: Instruction Set Architecture.
- **PTVS**: open source and free Best Python IDE for Windows.
- **UML**: Unified Modeling Language.
- **Eclipse**: open source and free Best Python IDE
- **IDE**: integrated development environment.

4 Stakeholders and Design Concern

According to information from previous conferences, stakeholders of this software development are **Clients Tektronix Engineers**, and **Tektronix Customers**.

First and foremost, clients want a fully functional software, which means the develop team should fully test this software to get rid all possible bugs. clients also specify Python as the programming language so that developers must implement technologies in Python. Then, they also ask the develop team to implement the communication layer and IP address related operations. The developer team should also consider maintenance and update issues. In other words, the software should be sustainable.

Based on the requirements checklist and conversions with clients, the develop team realized that the most important design concern is the usability of the software. Clients stated that the command syntax of VISA was too complex to memorize [1]. Tektronix engineers and its customers had to spend extra time on checking the programmer manual. Therefore, clients asked for accurate and quick documentation access services, the code auto-completion and the keyword query program. The responding accuracy and responding time become two design concerns for this software.

Both Tektronix engineers and Tektronix customers use this software to write software/scripts to Tektronix devices. The software should ensure the completeness of information being transferred to devices. Furthermore, the software should give enough permissions to them so that they can access to packet level or other type data to debug.

5 Design views of requirements assigned to Wenbo Hou

5.1 Creating the Documentation——Information Viewpoint

5.1.1 Design Description

The quantity of the documentation directly determines the user experience of the software. A well-constructed documentation can provide accurate and detailed information to users so that they can quickly find needed information. Besides that, a consistent data structure allows maintainers to take advantages of a template to update the library quickly without affecting the software's functionalities. The code auto-completion program and the query program also require a high quality data access schema.

In this project, the designer chose the CSV file to build the command library. The CSV files store the data like database tables do. Each row contains the specific information of one entity [2]. Fields in each row contain all attributes of this object. With built-in Python APIs, the software can quick search through the CSV file with SQL format commands from the Python implementation querycsv.py.

Then, designer will modify the programmer manual from Tektronix and other authorized organizations to fit the CSV data structure. Consequently, the correctness and accuracy of the documentation are ensured.

5.1.2 Key Elements

- **Design Entities**
    1. **Data Item**: The detailed Information of built-in commands, Attributes and Events.
    2. **Data Class**: Operation, Attribute, and Event.
    3. **Data Structure**: CSV file format.
    4. **Access Schema**: querycsv.py.
- **Design Relationships**
    1. Data items will associate with a data class. Information of commands belong to the Operation Class.

2. The information of a command contains the commands' syntax, description, input, and output.
3. The information of a special parameter contains the parameter's name, usage, and developer comments.

- **Design Attributes**

1. The documentation is built inside the software to avoid improper changes or undetectable lost.
2. Modifications of the documentation must be approved by Tektronix engineers to ensure the correctness and the consistency of the documentation.

5.1.3 Design Diagram

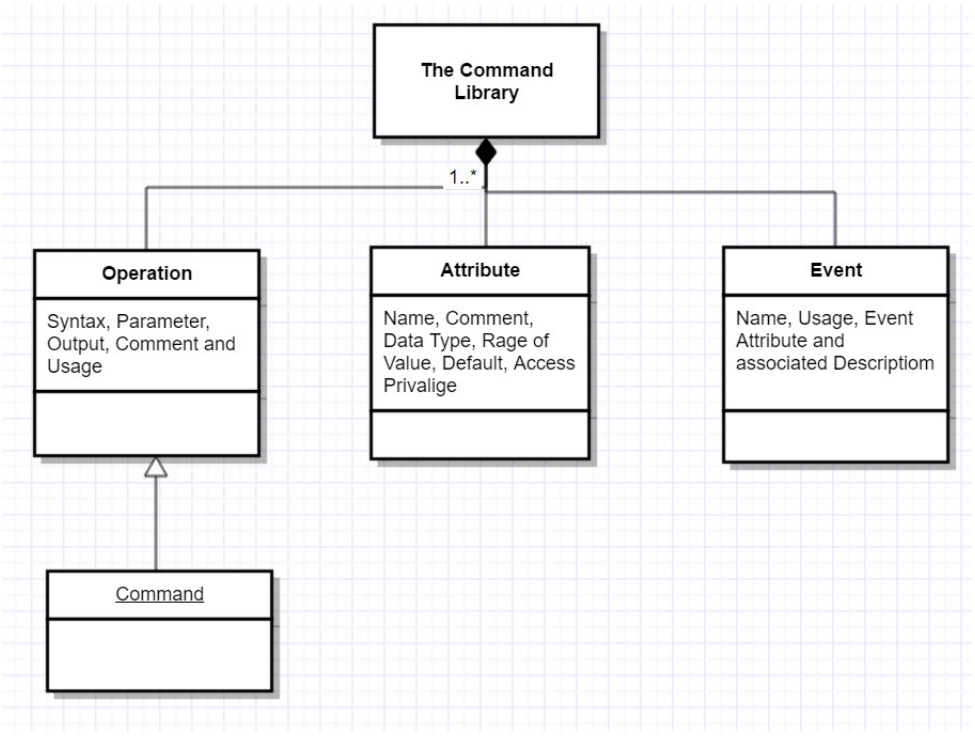

Figure 1: The UML Diagram for The Documentation

5.2 Querying Documentation with a keyword——Interaction Viewpoint

5.2.1 Design Description

Querying information with keywords requires the program interacts with the documentation and the user interface. First, the user can activate this function by clicking the search button. Then, the function reads the keyword and search it through the built-in documentation.

When users open the search window and input keywords, User Interface will invoke the query program. Then, this program will implement searching APIs from querycsv.py, to query the documentation. After that, it will modify and return the query result to User Interface. The query result may contain the found information or error messages. User Interface will display the result in the search window.

5.2.2 Key Elements

- **Program Properties**
  1. The trigger function built in User Interface
  2. Not concurrently run
  3. Do not need a separate process
  4. Take advantages of querycsv.py
- **Involved entities**
  1. **User Interface**: reads inputs and pass to the query function
  2. **The query program**: takes input keyword and query information from the documentation
  3. **The documentation**: contains programmatic information related to the software.
- **Events in the interaction**
  1. User Interface invokes the query program.
  2. Query program searches the documentation with keyword.
  3. The query program returns search results to User Interface.
  4. User Interface display results.
- **Program States**
  1. The query program is triggered by clicking search button on the user interface.
  2. The query program is deactivated when a query work is complete or not invoked at all.

5.2.3 Design Diagram

5.3 Code Auto-completion——Context Viewpoint

5.3.1 Design Description

The code auto-completion is an addressed requirement of this software. Tektronix engineers emphasized that they had a hard time using the old communication tool because of the compound command syntax. They wanted a code auto-completion feature in the new software so that they could get rid of programmer manual and save time.

To achieve this goal, the designer decides to take advantages of IntelliSense, the code auto-completion program from Visual Studio. This program will read keyboard inputs while users are typing commands in the programmatic window[3]. Then, the program will provide a list of command syntax that starts with being typed letters. Users

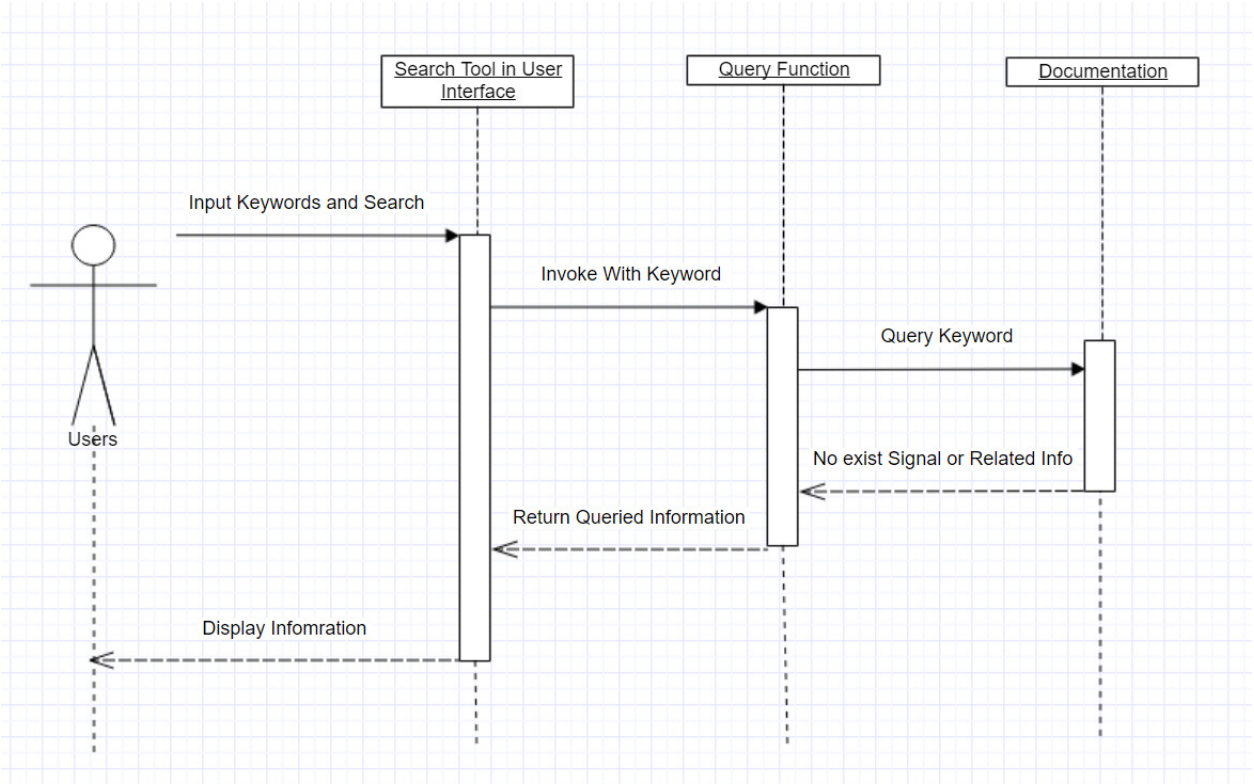

Figure 2: The UML Diagram for The Query Program

can use the list as a guild to manually type the commands. Users can also select a command from the list and ask the auto-completion program to complete it. The size of the list will shrink down as the user typing. If a typed in command does not exist in the documentation, the program will highlight it and ask users to double check it.

The designer will rewrite the program in Python to meet the programming language requirement from our client and prepare the API for User Interface.

5.3.2 Key Elements

- **Design Entities**
  1. **Actors**: Users who are typing into the program window.
  2. **Black Box**: Code auto completion function.
  3. **Input Information**: Keyboard inputs or Selected command ID
  4. **Output Information**: A list of commands syntax starting with input letters or a completed command syntax in the command line.
- **Design Relationships**
  1. Users provide keyboard inputs to the code auto-complete program. Then,The code auto-complete program will return a list of command syntax starting with input letters to users.
  2. Users select a command from the command list. Then, the code auto-complete program export the full syntax to the command line.
- **Design Constraints**
  1. The responding time of the code auto-complete program should be short enough to follow the user typing speed. As a result, users would get instant syntax reminder.

2. The code auto-complete program is not allowed to provide incorrect command syntax. Illegal or not-found letters will be marked.
3. The code auto-complete program should be compatible with the User Interface. User Interface shall leave space for code auto-complete program APIs.

5.3.3 Design Diagram

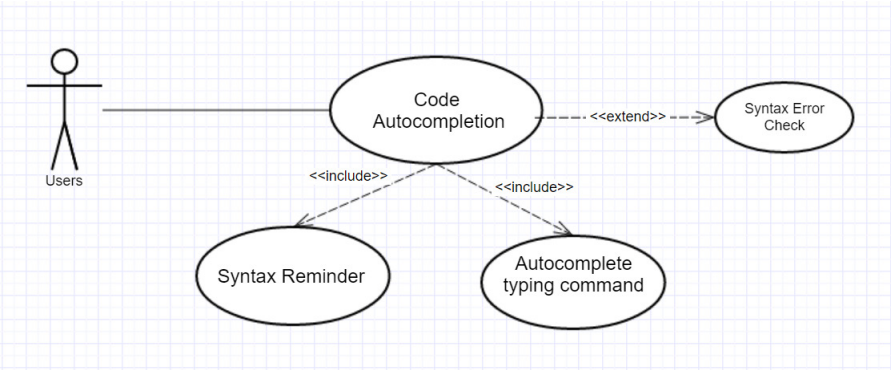

Figure 3: The UML Diagram for The Code Auto-completion Program

5.4 Device Type Identification——Interaction Viewpoint

5.4.1 Design Description

The identification of a being operated device is another requirement from Tektronix engineers. They claimed that the identification information could help them to keep tracking properties of the target device so that they might operate it efficiently. Products from Tektronix are various including, oscilloscopes, analyzers, signal generators, and other test & measurement devices. This fact implies that different command families exist. With the identification information, Tektronix engineers can get rid of distractions from other command families so that they can improve the work efficiency.

To show accurate identifications, the program needs to interact with the device and User Interface properly. First, the program calls the identification API[4] through the internal communication interface to get IDN number when a user selects a device to work with. Then, it sends an unique signal to User Interface. Finally, User Interface shows the identification information to users. The Identification information contains different font colors in the programmatic window and the name of the target device type in the programmatic window title.

5.4.2 Key Elements

- **Program Properties**
    1. The trigger function built in User Interface
    2. Not concurrently run
    3. Do not need a separate process
    4. Take advantages of PyVISA.py
- **Involved Entities**
    1. **User Interface**: Trigger the Device Identification Program and Change the font color and programmatic title based on the received signal.
    2. **Device Identification Program**: Invoke IDN API, analyze returned IDN Info, and send ID signal to User Interface
    3. **Connected Device**:Receive IDN request and send back IDN Info.
- **Events in The Interaction**
    1. Trigger the Device Identification Program.
    2. Identify Connected Device
    3. Show Identification Information
- **Program States**
    1. The Device Identification program is triggered When user selected a device in User Interface.
    2. The Device Identification program is deactivated when a Identification work is complete or not invoked at all.

5.4.3 Design Diagram

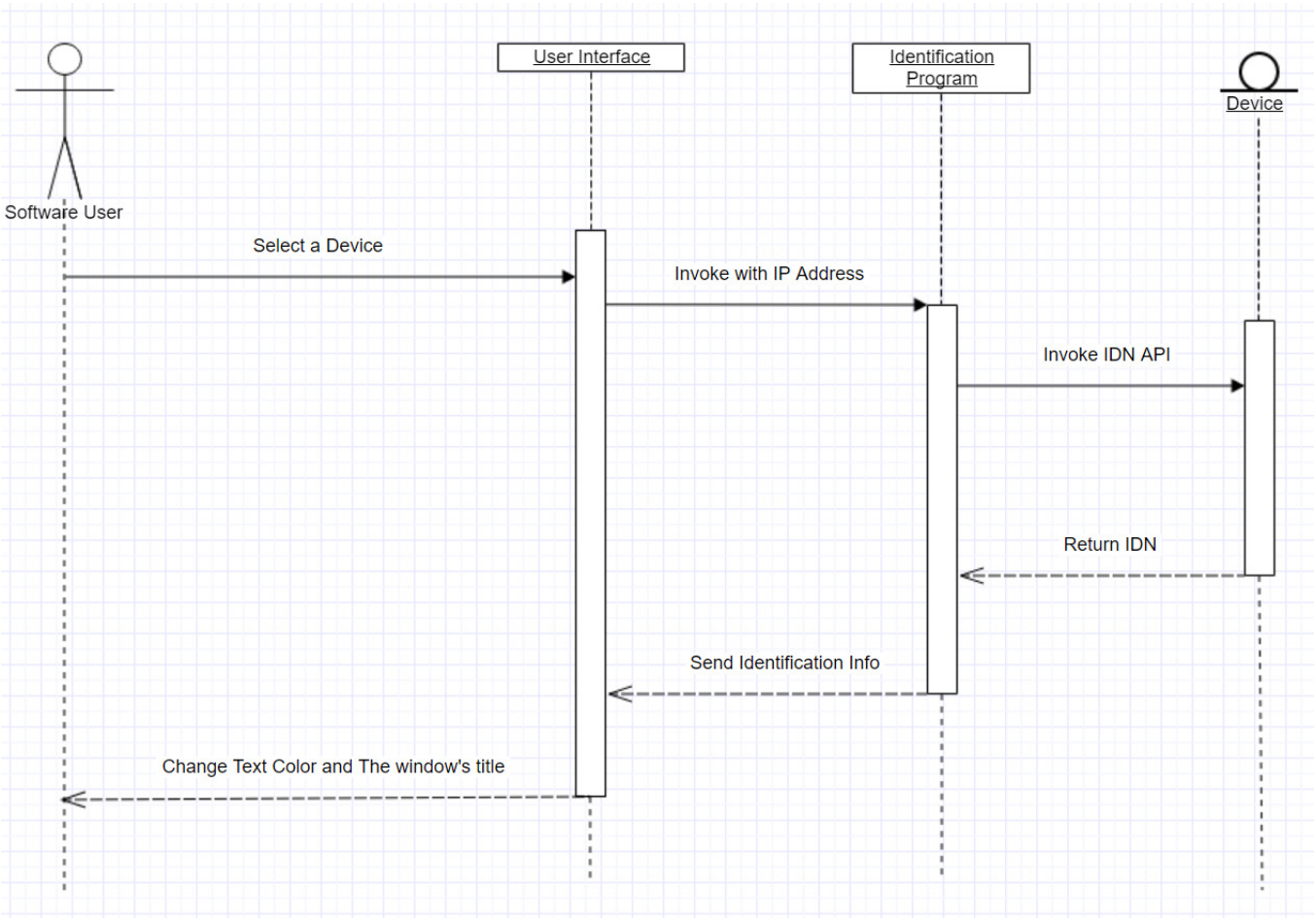

Figure 4: The UML Diagram for The Device Identification Program

6 Design views of requirements assigned to Chenliang Wang

6.1 VCT GUI Design——Structure Viewpoint

6.1.1 Design Description

This section is about the user interface design. The best option what we chose last time is Balsamiq Mockups. Using this software, we can design our VCT user interface online with a high level concept and change easily.

Through the Balsamiq Mockups, we can drag the button, create the connection between button and new windows. It is not the realty VCT is, but it is a good example about what is VCT looks like with almost connection. When we done this, users can check the user interface design through Balsamiq Mockups and point out the problem easily and quickly. All in all, Balsamiq Mockups.is not just the statics pictures, it is the basic user interface design of a high level.

To implement this part, we need research about similar concept such as other VCT. From these, we can get some idea about our VCT. We can discuss about the advantage and disadvantage in order to improve our VCT as better as we can. After this, we will use Balsamiq Mockups to create the user interface. During the create process, we can research online to watch some video of how to use Balsamiq Mockups. It maybe not useful for us of how to use but we can get some good idea during this process. According to video, we can understand all the items exactly for. From the online resources, we know how to create project of Balsamiq Mockups, how to use items in it, how to create link of the buttons[6-8].

6.1.2 Design Elements

- **Key Proportion**
  1. Reasonable layout: size, color, buttons and text position are all suitable.
  2. Include all function buttons: such as write, query, drop down.
  3. Include all items: such as command input text, text display area.
- **Design Entities**
  1. UI whole model: the whole blank model.
  2. Navigation bar: users can find functions easily.
  3. Command text: input command.
  4. Function buttons: such as write,query.
  5. Command auto complete area: display auto complete command.
  6. Connection information area: display the IP, Hostname, model of the device.
  7. History drop down list: see the recently 25 history commands.
  8. Process feedback area: display the situation of VCT.

6.1.3 Design GUI

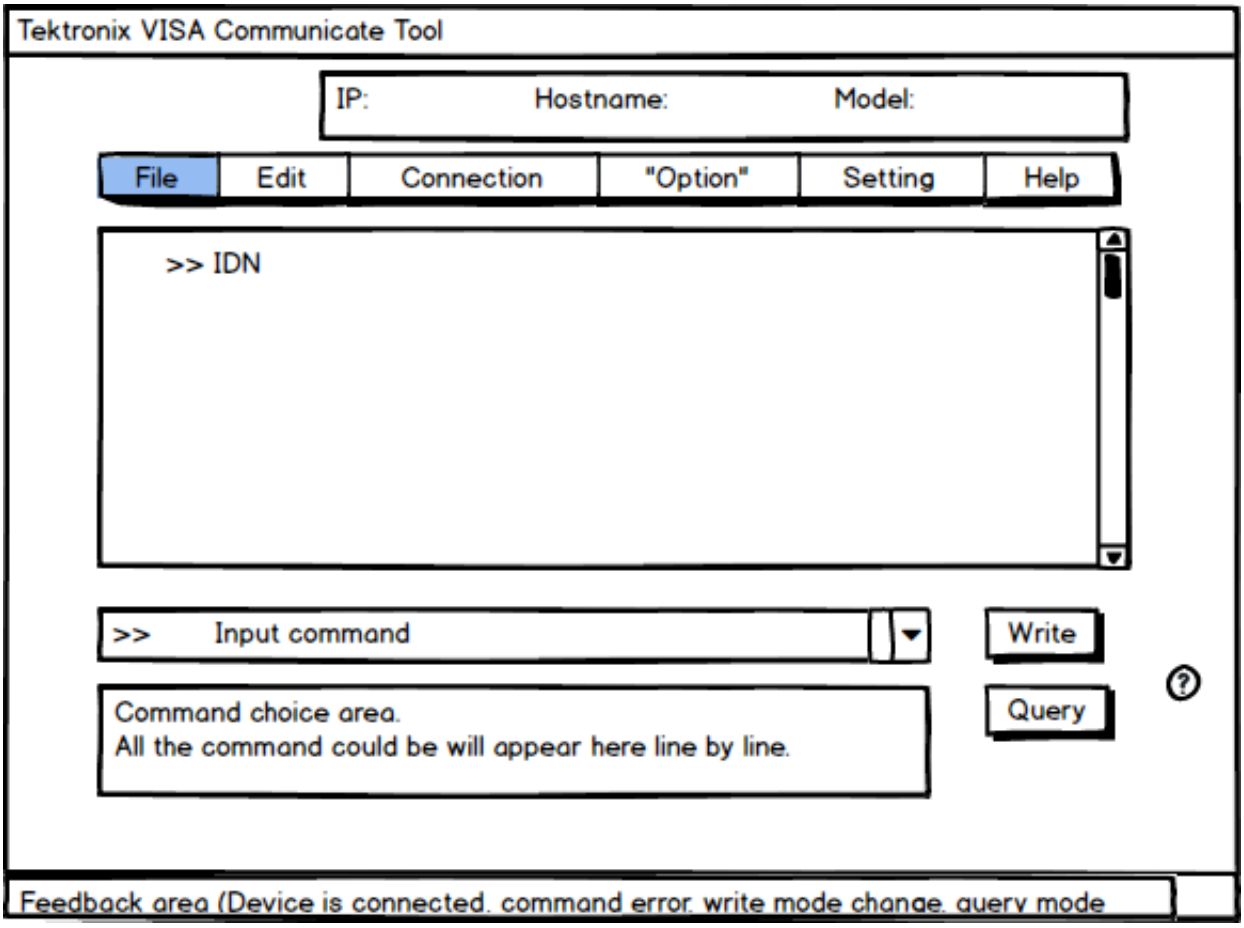

Figure 5: The draft GUI design of VCT

This is the darft of GUI design throught the Balsamiq Mockups. It include the all design elements. Clients can understand the layout of the VCT and we can change easily if somewhere aren't what they want.

6.2 VCT UI Framework——Patterns Viewpoint

6.2.1 Design Description

User interface is the one piece in our project. From the last document of technology review, we research 3 options of user interface framework by python, and we choose the best option is pyqt.

Using pyqt to create a user interface is a good choice. It is an option with GUI, so it can make use to create our project user interface easily. We can drag some item to our screen and type code to improve it.

According to "Getting started with Pycharm and Qt 4 Designer", it is the example about make a simple user interface based on python. From this example, we can know how to use pycharm and pyqt to make a user interface. It includes how to create file, drag item, save file, run file even some places of how to use code[9]. Also, from the YouTube, we found a good resource about the code of the python. It is clear from how to install the software to done the whole user interface. As the code of this, it separate to different parts of functions. Such as init, home, run, etc[10]. This is a good way for coding and debug in the future. It can find error easily as the parts are separate. Also, accoding to "Create a Basic GUI Using PyQt", we understand more about pyqt, and pyqt is good for cross-platform with Windows, Linux, and Mac[11].

To implement this part, we will double check the user interface sketch in order to make sure there are no any problem.

Then, we will use pycharm and pyqt to create as same as my sketch. We will drag the all buttons first and use code to improve and implement the functions of all buttons. In order to make the user interface friendly, we will make the feedback of the software displayed at the below of the user interface. When users do some actions, the information will be display that users can know exactly what is software do and what is happen. The buttons which can't use will be unable to click. It will avoid some wrong actions. Users will know clear that this button will not be able to click at this time. Before I choose the button, we will research the similar software to make sure the buttons are as similar as others. This is a good way to make users can learn and get this software quickly.

6.2.2 Design Elements

- **Key Proportion**
  1. Implement GUI design to really tool: implement the VCT user interface base on design.
  2. Navigation bar works: users can click the navigation bar to choice what they want.
  3. Feedback was displayed.
  4. Also buttons works.
- **Design Entities**
  1. Pyqt: use this to create user interface.
  2. Size of VCT windows: the size of VCT should be suitable.
  3. Navigation bar skip: navigation bar could be use.
  4. Command text.
  5. Function buttons.
  6. Command auto complete area.
  7. Connection information area.
  8. History drop down list.
  9. Process feedback area.
  10. Device connection interface.

6.2.3 Code example of Pyqt

```
__author__ = 'Drona@elementz'
import sys

from PyQt4 import QtCore, QtGui
from testlgui import Ui_Form

class MyForm(QtGui.QMainWindow):

    def __init__(self, parent=None):
        QtGui.QWidget.__init__(self, parent)
        self.ui = Ui_Form()
        self.ui.setupUi(self)

        self.ui.pushButton.clicked.connect(self.helloworld)

    def helloworld(self):
        self.ui.textEdit.setText("Hello world")
        print ('Hello world again')


if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

Figure 6: The code example of create UI based on Pyqt[9]

This is the sample of code with click and text display based on pyqt. From the several examples, we can get how to use pyqt to create user interface of VCT.

6.3 Function Testing——Structure Viewpoint

6.3.1 Design Description

Testing is one important part for a project. It is not an option that no one software can by successfully without testing. From last technology review, unit testing is the best option for function testing.

As we do this project based on python, we have a good unit testing can be use of python. According to python doc online, "unittest — Unit testing framework", there exist a library called "unittest", there are many testing in it. When we want to use this library, we just need "import unittest". For our project, we need create the VCT, there are a lot of functions in it. From the GUI create to whole software done, all the function what we build all need testing by unit test. During the coding process, we will separate the code based on different parts. The ideal way is each functions have their own function. Therefore, we can do the unit test by each functions one by one. It is not only easy to create unit test function, but also easy for us to figure out the error and fix it. "The unittest module can be used from the command line to run tests from modules, classes or even individual test methods" [12].

To implement unit testing, we will list all the function what we create during the coding process. We also will check the unit testing doc to make sure we did the right unit testing. The code of unit testing will such as below.

6.3.2 Design Elements

- **Key Proportion**

1. Create unit testing function: create unit testing function based on each function.
2. Unit testing: run the testing function to check the each function what we use.

- **Functions**

1. Click button: When users click button, it should do the right action or jump to the right interface..
2. Navigation bar: all button in the navigation bar can be use with correct action.
3. Command write and query: users can input command with write or query.
4. History: users can check the history with recently 25 commands.
5. Feedback: users can view the correct feedback of the VCT.
6. Command auto complete: users can see the list of option for the auto complete commands.
7. Device connection interface: Users can connect the device by ip or hostname, and the information will displayed on the top of the VCT.

6.3.3 Code example of unit testing

```
import unittest

class WidgetTestCase(unittest.TestCase):
    def setUp(self):
        self.widget = Widget('The widget')

    def test_default_widget_size(self):
        self.assertEqual(self.widget.size(), (50,50),
                         'incorrect default size')

    def test_widget_resize(self):
        self.widget.resize(100,150)
        self.assertEqual(self.widget.size(), (100,150),
                         'wrong size after resize')
```

Figure 7: The code example of unit testing based on python[12]

This is the sample of code with unit testing based on python. From the several examples, we can get how to create unit testing function and how to test each functions.

7 Design views of Requirements Assigned to Lucien Tamno

- **Introduction** As communication piece of the VCT software, the module heavily relies on the information received from user interface and has a goal to properly channel data from the receiving end to the sending end without altering, delaying or interrupting the course, or either: running them into the wrong path.Therefore, this process requires among of software interfaces, made up a greater number of classes, functions, subsystems as summarized on following table (Table 1) and further down details into subsections.

Organization Design viewpoints		
VISA API INVOCATION	Data Management	Connection with Hardware
Viewpoint Description	Viewpoint Description	Viewpoint Description
Key Components and utilization	Interconnection Data Management	Key Components testing techniques
Design diagram Associate	Design diagram Associate	Design diagram Associate

Table 1: SUMMARY COMMUNICATION DESIGN

7.1 Communication Interface-VISA API INVOCATION————-Logical Design Viewpoint

7.1.1 Viewpoint Description

VISA API Invocation viewpoint is to provide details to any stakeholder of the VCT system on how the communication interface handles and channels data form the streams coming from the user interface all the way to interact with the hardware of a hosting device. essentially, written in python via Eclipse [13] which implements an efficient Pydev free of cost and packed with with powerful features for Python programming language. alternatively we will use for the windows system the most appropriate IDE named PTVS which can in replacement of Eclipse play effective role development on windows environment . So the Invocation viewpoint will be a set modules and each module made up with classes, and in turn, each class made up with functions or subroutines. All the modules put together will be as subsystem known as: VISA API Invocation subsystem and that subsystem will use as describe in the next section.

7.1.2 Key Components and Utilization

1. **Interface Stream-in : communication-userStream-in interface**

   this interface has the unique goal to look at the incoming stream traffic from the user interface and check the following conditions:

   - Format of incoming streams comply with any implemented module format:"ex:*class-COMUSER-FormatStream-in* ".
   - Enough viable stream to manage: "ex: *class-COMUSER-ManagementStream-in*".

   When validations occur at this level the stream is received and sent to the management Interface, which will be elaborated on the section data management.

2. **Interface Stream-out : communication-userStream-out interface**

   The goal of the interface of communication is to look at the availability of user interface and to stream data out.Checking the availability of the user interface, the outgoing stream has to have functionalities of sending queries and getting responses as follows:

- the query was issued to the user interface and response approved : "ex: *class-queryUser-out, class-Userresponse-okay* "
- : the outgoing stream encapsulates with the correct format: "ex:*class-formatUserstream-out* ".

those data doing to the user interface were collected from another interface called: command query and auto-completion.therefore, we have make sure those data had been properly received as well.

3. **Interface Stream-in : communication-Autocompletion-in interface**

   Here data are received from the interface query and auto-completion reading the memory of the instrument plug into the system and has to comply to:

   - Incoming streams form query Interface format:"ex: *class-FormatStream-AutoInf-in*".
   - Enough viable stream to manage from the query auto completion interface : "ex:*class-ManagementStream-AutoInf-in*".

   Same as the user interface the communication has to make sure the formats of data sent out are relevant to the auto-completion interface.

4. **Interface Stream-out : communication-Autocompletion-out interface**

   At this point the communication is interacting with the auto-completion counterpart and

   - the query was issued to the user interface and response approved :"ex: *class-query-out-Autocompletion, class-response-okay-Autocompletion* "
   - : the outgoing stream encapsulates with the correct format: " ex:*class-formatstream-out-Autocompletion* ".

7.1.3 Design diagram Associate

7.1.4 Testing Techniques

7.2 Communication Interface-Data Management————-Dependency Design Viewpoint

7.2.1 Viewpoint Description

7.2.2 Interconnection Data Management

7.2.3 Design diagram Associate

7.3 Communication Interface-Connection with Hardware————-Interaction Design Viewpoint

7.3.1 Viewpoint Description

7.3.2 Key Components and testing techniques

7.3.3 Design diagram Associate