Software Testing by Standard Software Metrics Method; Study Case "Mission Planner" as UAV Ground Station Software

Adi Susila Putera [1], Fatwa Ramdani[2]
[1]*Faculty of Computer Science, Brawijaya University, Malang, Indonesia.*
[2]*Geoinformatics Research Group, Faculty of Computer Science, Brawijaya University, Malang, Indonesia.*
adhie065109319@gmail.com

***Abstract*—This paper explains the testing of a complexity of software and complexity of flight data from Mission Planner as UAV Ground Station software. Tests were conducted using the software metric method. The analysis, testing, and calculations were applied using the method of software metric to investigate the complexity of the software and the flight data on Mission Planner. The tests were performed on three different OS (operation systems); Windows, Linux, and Mac. The result shows that Windows OS has the most reliable software environment and flight data than the other two software.**

***Index Terms*—Ground Station; Mission Planner; Software Metric; Complexity Software; Windows; Linux; MAC OS**

I. INTRODUCTION

In general, ground station software is used to monitor the UAV behavior when it is operated. Flight plan can be defined with the starting point coordinates, while the path and the coordinated path will be observed. The monitoring process can be accessed from a telemetry connection, which is placed on the ground for the purpose of monitoring and observing the condition of UAV.

The Mission Planner (MP) is a software for autopilot of the aircrafts, helicopters, or rovers. This software is compatible with Windows. In addition, MP can be used for monitoring the situation and condition of autopilot on the ground and to receive data from telemetry for the production of many commands that control the flight parameters of UAV.

Some of the things that can be done with the MP are controlling the vehicle (Auto Pilot), optimizing performance, saving and loading autopilot autonomy mission with simple point-and-click, as well as analyzing the mission logs and flight simulator. With the telemetry hardware we can monitor the status of the operated UAV. Telemetry logs will record all information onboard log autopilot, view and analyze telemetry logs from the ground station.

Software module complexity assessment is crucial in software engineering study [1].This study will use MP software to assess the software complexity as well as its data complexity on three different operating systems (OS). The testing phase itself is done using the calculation of the implication by using the software metric method. [1]

The initial testing phase is determined based on Metric standard quality and validation[2], which calculates the implication software and data based on Metrics for specification quality, Design model metrics, System Size, Depth, stripe and AN ratio.[3] The results of the test, the implication software and the data on MP can help us know the Institutionalization indicators implication on three OS that were tested.

The architecture of the MP software is shown in Figure 1. Software testing will illustrate the efficiency and reliability in a way that is measured. This paper is aimed at calculating the metric software for testing as well as the complexity, efficiency and reliability of MP software.

II. METHODOLOGY

The testing phase is done using several stages of the testing phase with the poaching *Software Metrics*. In the testing phase that uses the method of calculation of the metric implication on MP software, the test was conducted to know the indicators of the complexity of the MP software.

This test is done on three different OS (i.e., Windows, Linux, and MAC OSX) to know how the complexity of the software when it is tested and running on three different OS.

The MP software is open source that runs on on Windows OS, Linux and MAC. It can be downloaded and learnt by anyone.

To generate the real flight data, we used a fixed-wing type UAV (Unmanned Aerial Vehicle) with APM 2.6 autopilot. We used the MP software as the ground station to monitor the UAV behavior and finally get the detailed flight information.

The software testing is done with the process of loading the data through the MP software. The data inputted can be seen and monitored through MP software Graphical User Interface (GUI).[4]

III. PREVIOUS RESEARCH

There are many studies and discussions related to the topic of this research and the common testing method used is the metric calculation. Standard software metric assessment had previously been discussed in Fenton and Neil [5] focuses on testing the software control of a ground station. However, the software used is not open source.

Software and monitoring have an important role in the operation of the ground station. In this paper, the development phase of the monitoring and control software verifies the test and system architecture using metric, system size, and depth metric. The testing phase is done with the analysis and verified based on metric software.

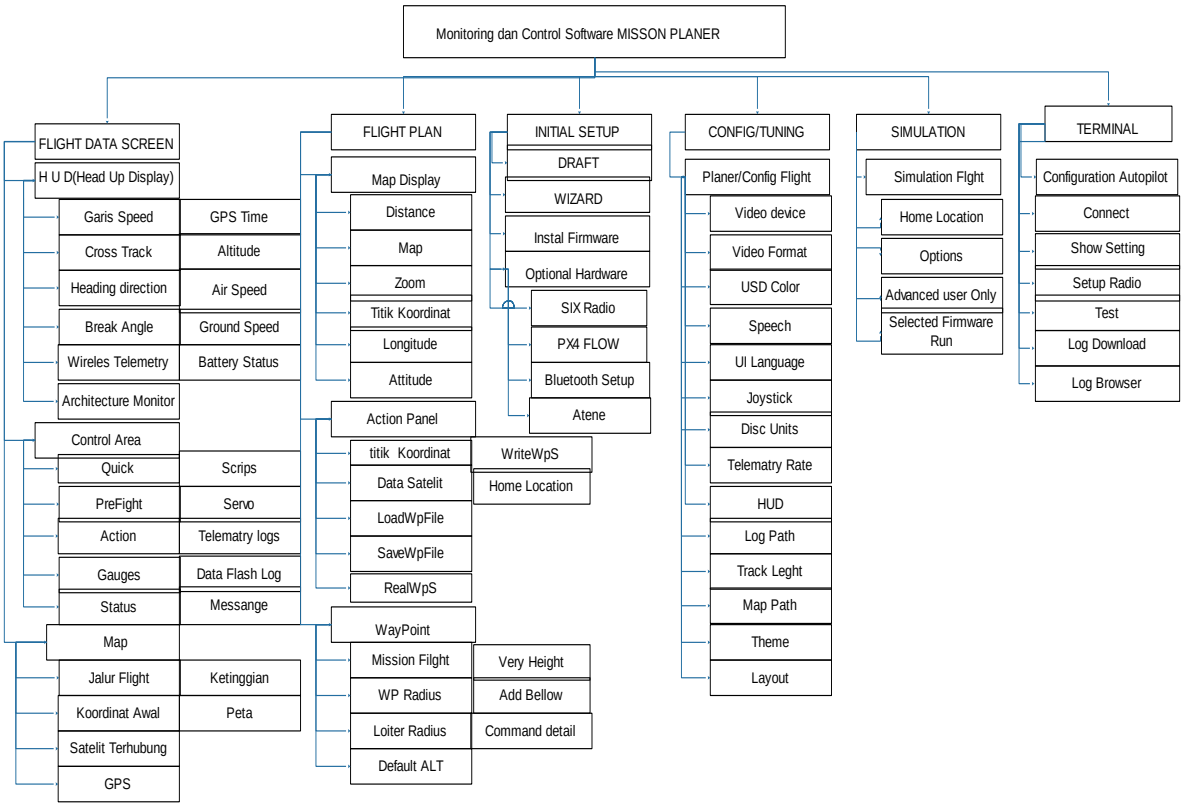

Figure 1: Software Architecture of Mission Planner

Hamayun and Soomro [6] revealed that the overall complexity of software can be tested, but there has been no explanation of the test evaluation tests on software reported by all the literature. They tend to focus only on when the testing process and the results of loading data are in accordance with the command in the input on unmanned aircraft.

The development of the Automatic Voice identifier (ASR message)[7] on MP software is a system that can be integrated on the ground control station from MAVs to know the activities and a voice command. There are two papers that relate to this work. The first one is about the design of an application aerospace with a voice command and the second paper focuses on the development and integration of the ability of a message of ASR against the Ground Stations.

In relation to the test using the algorithm above and the development of integration on a typical ground station and on MP based on MAV, the paper concludes that the evaluation of the voice mail message ASR laboratory and discussion about the steps should be adapted to form a system that can cope with the real application scenarios.

Bukhari et al. [8] discussed some of the proposed metrics for the software development process and quality evaluation of the software. In this paper, we reviewed the metrics proposed for the selection. The metric was based on the external measurement as the first step toward the determination of the model method metric software. With respect to the software metric, the same approach in the stage of development and evaluation of the software to measure the quality of the software has been used.

The determination of metric software in the software testing is important for the purpose of choosing the most appropriate metrics for the evaluation and assessment of the software. The determination of the method and method selection software metric is based on the external measurements and determination of testing and analysis software.

In addition to using it as a test and monitoring the flight ground station[9], it can also be made part of a ground station that can be used as nano and pico satellites. Flight line testing on grandstations are located at the frequencies VHF, UHF, and S-band around songs MHz, 435MHz, and 2.3GHz. Ground stations are ideal and suitable for some mission that allow for flight and seamless path of the ground flight stations to the flight ground stations. A ground station is designed in some modular so that it can be tested in various conditions and paths of the different flights.

Flight safety [10] involves the time to get attention on the UAV. The paper focuses on the development status, predicting errors and UAV flight management. Monitoring the UAV flight simulation systems includes determining the low state, analyzing real-time, modelling security systems, and monitoring ambient conditions. This paper focuses on a research related to the UAV flight monitoring system in real time. In this case, during the occurance of unexpected conditions, the commando commands instruction to the system, which quickly intervene the situation by providing a remote control, and made the recording process that can be stored and analyzed for the entire flight process.

This paper proposes the development of UAV technology that is able to be controlled [11]. The proposed process is able to fly independently and track the position of the flight. The proposed mathematical models include artificial technology consisting of artificial algorithm fed with a predetermined process, such as the ablity to walk independently, track the trajectory, break accurately, and record at all stages. The PID loop is designed to achieve stability at all stages of the

process. In the analysis process, it was found that the UAV system was unstable.

All the frequency reference functions to explain the results of the analysis and testing phase. The multitude of testing using the MP is to test the software only. The complexity of the MP in terms of the quality of the software and the data is not known. Therefore, this research was done in order to know the complexity of both the software and data on MP.

IV. Software and Hardware Used In The Study

A. Software

The testing process in this paper uses MP software, an open source and easy to use software. MP software is equipped with complete tools and GUI-based. The command on the unmanned aircraft determines the order of execution. MP software provides users with the opportunity to work on one application for the process of the program, testing and monitoring, and tools. It not only allows the development of user-friendly and graphics GUI, but also gives programmers the freedom in choosing what to display to the user at a specific time.

B. Hardware

In this study, we employed Ardu Pilot Mega (APM 2.6), a hardware IMU autopilot based on the Arduino Mega platform. This autopilot hardware can control fixed-wing, multi-rotor helicopter as well as traditional helicopter. The autopilot hardware has the ability to stabilize the UAV navigation point and two-way telemetry with Xbee wireless module. It also supports 8 channel RC with 4 serial port. APM consists of the main board processor and shield IMU. The open source control software is constantly updated with new features and improved by a team of around 30 core developers, supported by communities of more than 10,000 members [12].

We installed the APM 2.6 in the fixed-wing UAV type, and flew it for 15 minutes to generate the real flight dataset. The result can be downloaded through MP log information. The format of the flight data is [dot] log.

C. UAV navigation point Metrics for specification quality

The development of the software consists of several steps: The first step in the process of the development of the software is the model of analysis described by the quality of the specification agreed by the consumers and the developers [15]; the metric has been brought down to the quality of the model specification analysis. MP software process that shows the transition diagram can be seen in Figure 2.

$$a_r = a_f + a_{nf} \quad (1)$$

where: a_r = total variable needs,
a_f = the number of functional needs, and
a_{nf} = the number of non-functional requirements.

Both of these parameters are generally calculated based on the requirements in the engineering project phase.

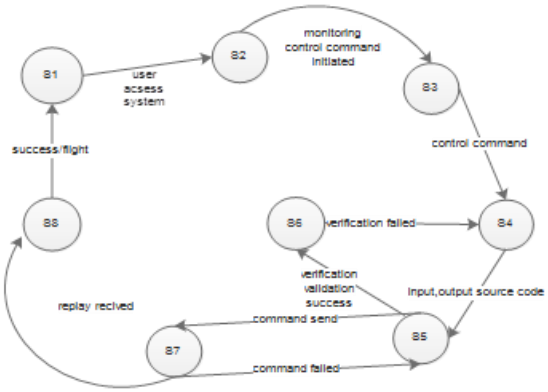

Figure 2: State transition diagram of MP software

a. Metrics for vagueness in architecture

Based on the above statistic, [13] derived various quotients for the specific quality. The metric to determine the ambiguity in this analysis was developed by [13]. These metrics provide the size of the entire software requirements specification.

$$K_i = \frac{a_{ui}}{a_r} \quad (2)$$

where a_r is the number of the total needed, and a_{ui} is the require ents inferred by the developer team.

b. Metric for completeness

Another other aspect of a model analysis stated that all functional requirements must be identified. K2 gives us the completeness of a functional software system.

$$K_i = \frac{a_u}{a_i * a_s} \quad (3)$$

where a_u specifies the number of unique functional requirement, a_i is a figure denoting number of inputs and a_s shows the amount of state specifically.

c. Validation metrics

The metric defined for the validation is as shown in equation 4:

$$Ki = \frac{a_c}{a_c * a_{nv}} \quad (4)$$

where, a_c shows the correct validation statement, *anv* shows the statement of validation failed or is not executed until the specified time.

D. Design model metrics

Once the updated analysis model is completed, the next step is still under construction with the software models of the design. The design or architecture model stresses on the structure and the effects of the module that are different from the previous models. Metric design can come without knowing the internal working principles of the module. The design metrics are closely linked to the complex ecology software, and there are three fruit metrics considered important in the metric development of this software [14]

The structural complexity is defined as:

$$S_{(j)} = f_{out}^{2}{}_{(j)} \quad (5)$$

where f_{out} shows the number of the module in the module j. This is also referred to as the fan-out.

The complexity of the data gives the interior of the complexity of the interface on the module j.

$$D(j) = \frac{\mathrm{V(j)}}{[fout(j) + 1]} \quad (6)$$

where, V(j) shows the number of variables I/ P and O/ P variables against the module j.

The total complexity is the number of a combination of the structural complexity $S(j)$ of the MP software S(j) and complexity of the data D(j). It is given by the folllowing equation:

$$T(j) = S(j) + D(j) \quad (7)$$

E. System Size, Depth, Width and AN ratio

The complexity of the software can also be defined with mathematical models using the metric as described in [15]. The simplest way in the process of the calculation of the metric is by describing the complete system that knows the process that ran and failed. The next stage is dividing the system and subsystems in describing the related variables, as shown in Figure 2. The calculation of the specification can be measured and calculated from the amount of nodes and graph output on each node.

$$Size = n + a \quad (8)$$

where: n = numbers of node,
a = arcs

The system depth is a way of determining the complexity of the vertical direction. It specifies the number of the module from the top to below or vice versa. The complexity of the software is defined according to horizontal direction. This shows that the number of the module on the level of the variables is the same as the software.

Ratio 'an' shows the reliability of designing the whole connectivity of the system and the software specifications.

$$AN\ ratio, an = \frac{\alpha}{\mathrm{n}} \quad (9)$$

V. Results and Discussions

Software metrics are defined and described in section V and applied to obtain the results of the following section:

A. Metrics for Specification Quality

Software specifications on the development phase are needed in the analysis, design specification, assessment, testing and verification of SOW (Statement of Work) that can be built and received by the third party. The result is shown in Figure 3 below.

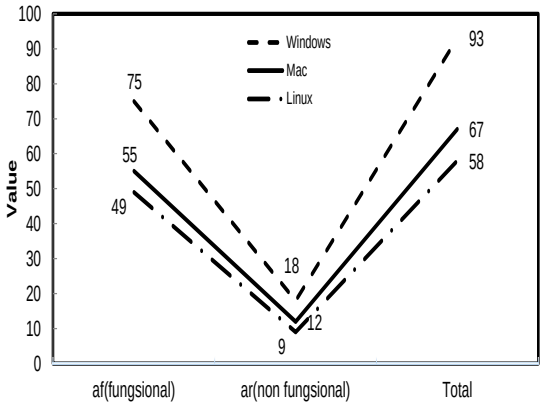

Figure 3: Metric for specification quality

a. Vagueness

The implementation of Equation 2 of each OS is shown in Figure 4.

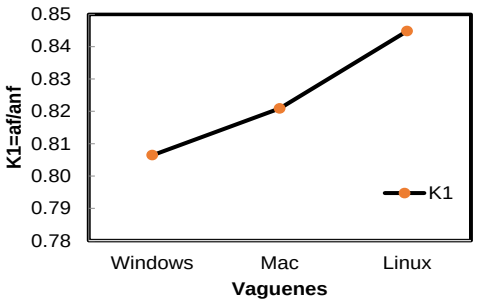

Figure 4: Vagueness of MP software on three different OS.

The ideal values of the specification K_1 must be the same with 1. It will never be greater than 1.

b. Completeness

Equation 3 is used to determine the completeness of MP software. The result is shown in Figure 5.

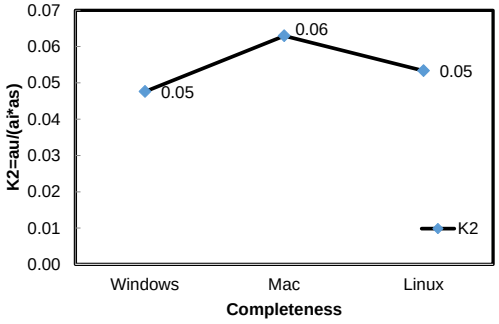

Figure 5: Completeness of MP software on three different OS

c. Validation

Equation 4 is used to implement validation as shown in Figure 6.

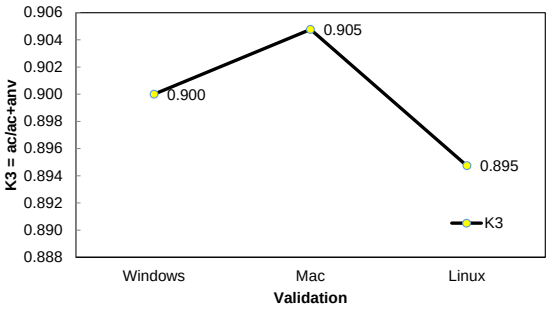

Figure 6: Validation of MP software on three different OS.

B. *Design Metrics*

a. *Structural Complexity*

Figure 7 shows the results for the application of Equation 5. Figure 7 depicts the structural complexity of the MP software on three different OS.

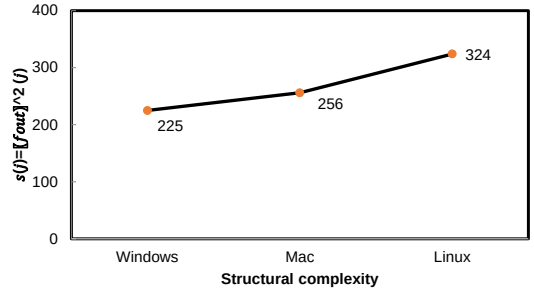

Figure 7: Structural complexity of MP software on three different OS

b. *Data Complexity*

Each module subsystem has different input specification processes and outputs on the MP software. The command inputted with the value specified by the user is related to the command specification on the software. The result is shown in Figure 8.

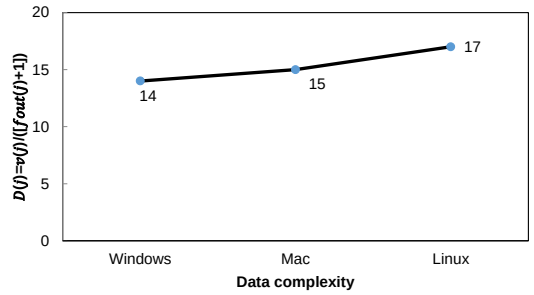

Figure 8: Structural complexity of MP software on three different OS

c. *Total complexity*

The total complexity is mainly dependent on the structural complexity. Figure 9 shows an image of the total complexity of MP software on three different OS. The graph is similar to the structural complexity.

C. *System Size, Depth, Width and AN ratio*

The next stage determines the size system, depth system, width and AN ratio. This stage employs Equation 8 and 9. MP software specification comparison using metric and its application on three different OS is shown in Figure 10 and 11 .

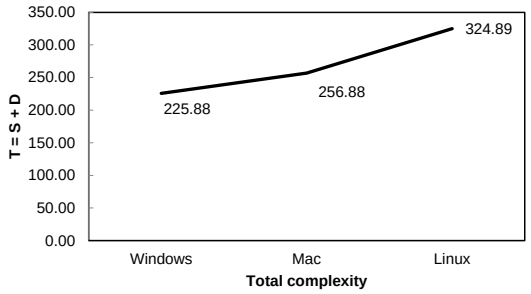

Figure 9: The total complexity of MP software on three different OS.

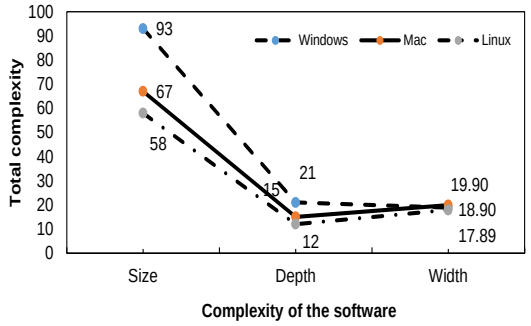

Figure 10: Size, depth, and width of MP software on three different OS.

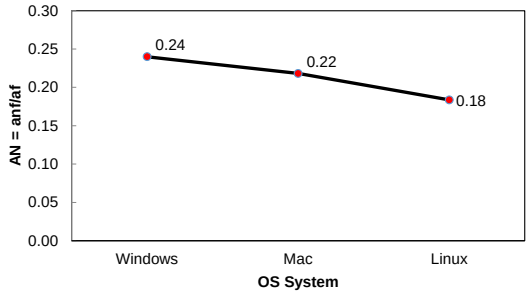

Figure 11: AN ratio of MP software on three different OS.

D. *Data Complexity Testing*

a. *Data Load Graphic Complexity on OS Windows*

In Window OS, the stored data dialing phase process is running a one time process only, although there are display of the information.

The stored data can be called as well as its display and results are presented in the graph.

b. *Data Load Graphic Complexity on MAC OSX*

Data dialing on MAC OSX is almost the same as on the Windows OS, as the iteration data dialing process one time only. However, there was an error at the time of the calling data with the process and display of the graph for large data as the bug error process is more than one time. In this case, the process of calling and raises the implication graph requires a long waiting time which influences the performance from the MAC OS and software.

The data shows a more detail and complex information which is confusing for the reading process and viewing the data implication through graphic. Further, the details of the data table on MAC are more complex than on the Linux and Windows and it is difficult to understand.

c. Data Load Graphic Complexity on Linux

Data dialing on Linux is different compared to other OS because it not directly called, but use the commands inputted in a terminal in Linux. At the time of the first dialing, failure occurred resulting in up to four times testing. This situation shows that it does not facilitate to use of MP software using Linux OS.

d. Complexity of Mission Planner

Using the Windows OS, Linux and MAC OSX, this system can be run and the data can be called on Linux and Mac OS, although there are some commands and different processes. Calling the data on the Windows and MAC is easier than on Linux. The structural complexity of MP software on the Windows OS is more efficient and easy to use, as it is based on the GUI. Installation process on Windows OS is a much faster process than on MAC OSX or OS LINUX.

VI. CONCLUSION

In the system testing phase, the performance of the MP applications were tested and verified in three different OS; Windows OS, Linux OS, and MAC OSX. The testing phase analyzed the calculation of matrix. The testing software does not only calculate the structural complexity but also the complexity of data. This is to ensure the reliability and wastage of MP software.

Calculating the complexity is to know the efficiency, reliability, and speed, but the most important and often overlooked is a parameter from clarity and complexity that describes the stage and development process and the required function of the software. This paper discusses all validation metrics calculation. This calculation can also be used by interactive upside down to ensure that the algorithm used to produce the result in the restrictions can be accepted. The test, which measures the implication of the data on the software was conducted to identify the complexity of the existing data. Calling data that can affect the implication on MP software, although data saved on Windows OS can be used to open MP installed on OS Linux and Mac OS. However, when data was dialed in MAC, there were error several times. This study shows that the development of MP on the Mac OS and Linux are less developed than in Windows OS. Users are recommended to use Windows OS when operating MP to get the most reliable flight data visualization but easier to understand.

REFERENCE

[1] Pizzi, N. J. (2011). Mapping Software Metrics to Module Complexity: A Pattern Classification Approach. *Journal of Software Engineering and Applications*, *4*(7), 426–432. http://doi.org/10.4236/jsea.2011.47049

[2] Srinivasan, K. P., & Devi, T. (2014). Software Metrics Validation Methodologies In Software Engineering. *International Journal of Software Engineering & Applications*, *5*(6), 87–102. http://doi.org/10.5121/ijsea.2014.5606

[3] Fenton, N. and Bieman, J. (2014). Software Metrics. 3rd ed. Hoboken: Taylor and Francis.

[4] Mission Planner Home — Mission Planner documentation.Ardupilot.org.. *Mission Planner Home — Mission Planner documentation*. Available at: http://ardupilot.org/planner/index.html

[5] Fenton, N. E., & Neil, M. (2000). Software metrics. *Proceedings of the Conference on The Future of Software Engineering - ICSE '00*, 357–370.

[6] Humayun, S., & Soomro, M. H. (2013). *Application of Standard Software Metrics*

[7] Rahul, D. K., Veena, S., Lokesha, H., Vinay, S., Kumar, B. P., Ananda, C. M., & Durdi, V. B. (2016). Development of Voice Activated Ground Control Station. *Procedia Computer Science*, 89, 632–639. http://doi.org/10.1016/j.procs.2016.06.026

[8] Bukhari, Z., Yahaya, J., & Deraman, A. (2015). Software metric selection methods: A review. Proceedings - *5th International Conference on Electrical Engineering and Informatics: Bridging the Knowledge between Academic, Industry, and Community, ICEEI 2015*, 433–438. http://doi.org/10.1109/ICEEI.2015.735254

[9] Fischer, M., & Scholtz, A. L. (2010). Design of a multi-mission satellite ground station for education and research. *2nd International Conference on Advances in Satellite and Space Communications, SPACOMM 2010*, 58–63. http://doi.org/10.1109/SPACOMM.2010.1

[10] Pengbo, X., Guodong, J., Libin, L., Lining, T., & Jigan, N. (2016). The Key Technology And Simulation Of UAV Flight Monitoring System, 1551–1557.

[11] Mallick, T. C., Ariful, M., Bhuyan, I., & Munna, M. S. (2016). Design & Implementation of an UAV (Drone) with Flight Data Record.

[12] APM Planner 2 Home — APM Planner 2 documentation, *Ardupilot.org*, 2017. Available: http://ardupilot.org/planner2/.

[13] A. Davis, et al., "Identifying and Measuring Quality in a Software Requirements Specification", *Proc. of First Int.. Software Metrics Symposium*, pp. 141-152, 1993

[14] D.N. Card and R.L. Glass, Measuring Software Design Quality. Prentice Hall, 1990

[15] Fenton, N. E., & Neil, M. (2000). Software metrics. *Proceedings of the Conference on The Future of Software Engineering - ICSE 00*, 357–370.