International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Quality Assurance Metrics in Embedded System

[1]Manjunath G, [2]Nagaraja G S

[1]M Tech, Depatment of Computer Science and Engineering, RVCE, Bengaluru, Karnataka
[2]Professor and Associate Dean, Department of Computer Science and Engineering, RVCE, Bengaluru, Karnataka

---

Abstract - *The Substation Automation System (SAS) is an integral part of the power grid which is responsible for the transmission of electricity. A very important component used in the SAS is the Intelligent Electronic Device (IED) that forms the building blocks of any modern day electricity supply system. The IED is a smart device that ensures safety by performing a variety of protection and control functions. With the help of IED, the various values that are measured using the sensors need to be transmitted to the substations so that appropriate action can be taken for a particular circumstance. In this paper, we come across the SAS and IED, its introduction and working principles and also few of the quality assurance metrics such as real time debugging, static code analysis and synthetic benchmarking. The paper also provides examples of the relevant tools that are used in the product companies to ensure that their products work efficiently during operation.*

Key Words: **Substation, Intelligent Electronic Device, Quality assurance, real time debugging, static code analysis.**

1.INTRODUCTION

Over the years, the revolution of the protective relay has occurred from the early day electromechanical relay which was of induction disc type that worked using electromagnetic force and limited operational characteristics. It was then followed by the static relay which had no moving parts but made use of discrete analog electrical components and selectable operating characteristics. However, in the present day scenario, we make use of the numeric relay which is capable of converting sampled analog values in to digital signals and having sophisticated communication features.

In layman's terms, the SAS is nothing but a network of devices that safely handles the job of delivering electricity from substation to the consumer's home and it also ensures that the system is in safe working condition. The IED is one of the key component that has a major role in SAS.

The embedded systems have become more prevalent these days and particularly in complex industrial systems. A lot of emphasis is laid on the dependability of the embedded systems because many of the processes rely on the function that is performed. The number of embedded systems as well as the complexity of the software running on them are increasing at a rapid rate. This makes the developers face challenges with regard to the robustness of the software and also against the faulty environment condition. [1].

With the advancements in technology, new reporting and monitoring requirements for electricity supply has been created. Any fault in the power grid has to be identified and isolated as quickly as possible ensuring high quality of distribution energy [2]. The traditional way of doing this is by adding more functionality to the IED.

The substation includes multiple zones of protection. The relays called IED operate the contactors which open to disconnect the faulty part of the Substation model. The Protection Panel operates as required under internal and external fault conditions. The Panel is capable of peer to peer communication between protective relays (IEDs) using GOOSE messages [3].

IEC 61850, as a communication protocol in the substation automation provides efficient performance through the exchange information in real-time. Moreover, one of the main motivations of using this standard is interoperability. However, connecting several IEDs manufactured by different vendors which are compliant with IEC 61850 standard is not a trivial task [4].

The IEDs are programmable digital protective relay which are capable of send and receiving data and control signals across the entire substation [6]. The various functions that an IED performs are event reporting and fault diagnosis, metering and power quality analysis, self-monitoring and external circuitry monitoring and programmable logic and breaker control. The operations can be performed both from within the substation or even through remote access

The paper is designed to highlight the use of quality assurance in embedded system. Particularly, the paper revolves around the substation automation which essentially deals with transmission of power from power generation station to the consumers. We start with the introduction in section I, followed by the communication standard IEC 61850 in section II. The remainder of the paper discuss about the various components used in a substation system and most importantly, the quality assurance metrics in section III and section IV respectively.

2. IEC 61850 COMMUNICATION STANDARD

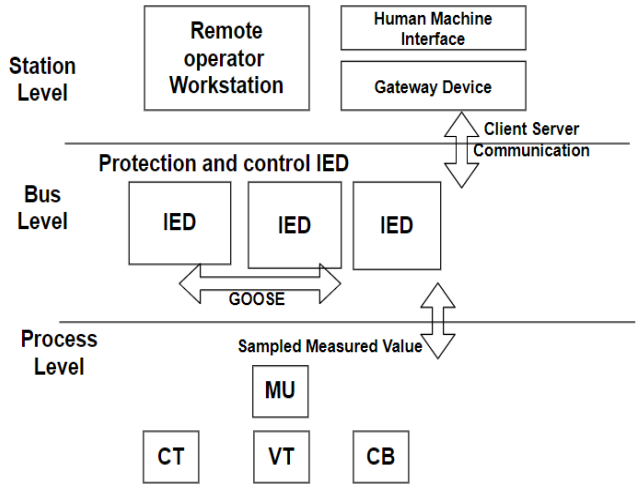

Fig. 1: IEC 61850 Architecture.

During the early days, the substation system did not have a standard protocol for communication between various components from different vendors. The vendors used their own proprietary communication protocol to interface between different devices. As a result, interoperability was not possible between components of different vendors. This led to invention of a new standard for communication between the components of substation system from different vendors called as the IEC 61850 (International Electrotechnical Commission). It was published in the early 2000s and the main aim of this standard was to provide all the requirement specification of the substation automation system and allow interoperability among components of different vendors. At the same time, a development was going on in parallel in the US by EPRI (Electric Power Research Institute) called as Utility Communication Architecture (UCA). It was aimed at providing interoperability among different controlling and monitoring equipment in substation system. Finally, both the organizations collaborated and published the worldwide accepted standard which is today known as the IEC 61850.

The main motive behind the standard IEC 61850 protocol was to provide a unified communication standard for the substation system. The objectives of the standard are mentioned as follows:

a) The standard must support good levels of integration between IEDs of different vendors.
b) Since the standard is used in safety critical systems, the standard must ensure that the response time is minimal and lies in the range of milliseconds.
c) The standard must allow flexibility in the configuration so as to meet the requirements of different architectures.
d) The standard must be stable and technology neutral so that the advancement in the future computer and communication technology does not affect the standard severely.

It is a commonly known fact that a new technology always tries to improve the existing system in terms of convenience, reliability, cost and power consumption factors. Although the adoption of IEC 61850 has introduced some cost, configuration and maintenance of certain equipment used in the substation, it has proven to be more cost effective as well as reliable and efficient when compared to the older substation systems. The use of network messages instead of hard wired combined with installation, design and deployment of the operation using the IEC 61850 compensates the increased cost and maintenance factors that occurs. The benefits of the protocol can be listed as follows:

a) The speed of data exchange between the IEDs is very high, thus ensuring high operation quality.
b) Interoperability between components of various vendors.
c) The hard wired communication is replaced and network elements are used which makes it possible for using peer to peer communication.
d) The use of Substation Configuration Language (SCL) in XML format is useful for the exchange of information between various software tools.
e) The concepts of object orientation, hierarchical data model and logical locations are supported.

3.COMPMONENTS OF A SUBSTATION AUTOMATION SYSTEM

The substation automation system as we have seen in the previous section consists of three levels and there are many different components that must work in coalescence so that the entire system can perform the desired tasks in an efficient and reliable manner. The following are the main components that an SAS in built with.

A. Servers

The servers are the computer system on which the SCADA (Supervisory Control And Data Acquisition) application will be installed. The purpose of SCADA is to gather the real time data in a timely manner and analyze them. It is primarily used in industries for monitoring and control activities. There are basically two kinds of servers employed in the SAS, which are the Database Server and the gateway server.

The SCADA application is installed in the database server. The task of this server is to receive the data signals from the IED and to send the response back to the IED so that the necessary action can be taken. The gateway server on the other hand takes the responsibility of communication outside of the substation and also manages the operation of all the substation present in a particular geographical area. Another important task of the gateway server is to convert the data from IEC 61850 to the required format so that it could be sent

over a communication channel. It can be argued that the servers are a single point of failure in the architecture. In order to avoid failure, a redundant server is always kept ready in a hot standby mode, if the need maybe.

B. Network Components

There are many network components involved in the system. However, the two most important devices used are the managed Ethernet switch and the redundancy box. The devices used are industry grade and not the commodity hardware. The advantage of using the managed switch over the conventional plug and play switch is that it allows configuration to be done as per the desired requirement. Some of advanced features of the manageable switch includes VLAN and RSTP.

The redundancy box on the other hand is commonly used only in case the network is deployed using the IEC 62439-3 standard. It uses the Parallel Redundancy Protocol(PRP). The redundancy box comes in to the picture when the components used in the substation system are not supported by the PRP and it can then be passed through the redundancy box.

C. Intelligent Electronic Device

The Intelligent Electronic Device(IED) is a smart device that forms the heart of the substation system. The primary responsibility of the IED is protection, control, monitoring and load balancing. The IED is located at the intermediate level which makes it possible to communicate both with the field device as well as with the SCADA application. The advanced features of the IED allows it process analog signals and convert them into digital signals. It can also send consolidated reports over the network.

D. Workstation Systems

The routine maintenance task and the various operational activity are performed using the two main workstation namely, the operator workstation and the engineering workstation. The Operator Workstation(OWS) is usually located in the substation for easy access to the operator. The OWS provides a graphical user interface(GUI) for the user to communicate with the server. The Engineering Workstation(EWS) is designed to cater to the needs of a specific role. It is mainly confined to configure the setting and parameters that the IED are supposed to work with. It must be ensured that the EWS is always secured as it provides access to any of the IED available in the substation system.

E. Time syncronization using Satellite clock

The aspect of time synchronization is very important in the SCADA applications of the distributed automation system. When the data is sent over a communication channel, it becomes very essential to know at what time the data was recorded. This is the reason why time stamp of the data is appended as it is recorded so that redundant data can be handled. The time synchronization can be achieved using the Simple Network Time Protocol(SNTP).

4. QUALITY ASSURANCE

A. Motivation

The life cycle of an embedded system product does not end once the development process is finished. In fact, it is very much necessary to confirm and validate that the designed product will function as per the requirement and ensure that it is also very reliable on the operational site. This step is very essential because if quality assurance is not done there are chances of malfunctioning of the device, the values may be lost or inaccurate, some of the components might have failed or damaged. Some cases may lead to damage of hardware used in the system and in the worst case scenario it might also lead to death of operational engineers who are working on site. There are chances that the faults may reach the end users and also damage the products used by the consumer as well as putting the consumers at a potential risk

It is essential to the development of software quality, especially software that can endanger the lives of human beings, as it is the case of embedded software components. The study and use of Software Engineering with its disciplined activities and principles to guide developers of such products to the process used for its development has been growing [5].

B. Real time debugging using Lauterbach's Trace32.

Real time debugging essentially means that the debugging activity that is removing and analysis of errors in the program can be carried out without halting the program under execution.

Lauterbach is a company based in Germany. It is the world's largest producer of microprocessor development tools. The company is known for developing real time trace and world class debugging tools since 1979. In this paper, we come across the product called as the Trace32 system. It supports technologies such as the JTAG, NEXUS, ETM with the embedded systems. The most outstanding feature is that it supports almost 100 cores deployed across more than 5000 chip families such as ARM, ARC, Power Architecture and many more. The software package provides a ready to run environment for Linux OS. Therefore, Trace32 makes analysis and debugging easier on the targets running Linux.

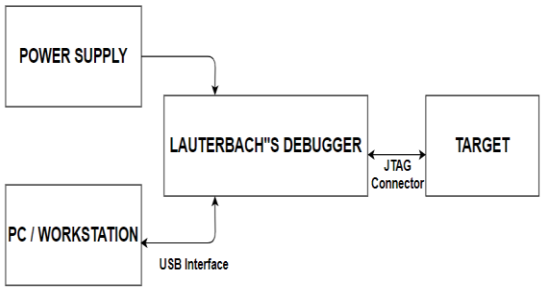

Fig. 2: Block diagram of the debugging setup.

The basic connection setup of the real time debugging using Lauterbach's Trace32 is as shown in Fig. 2. The debugging hardware uses a power supply of about 12 volts. It also provides an USB interface using which we can connect our personal computer or laptop to the debugger. It also provides another interface such as the JTAG to communicate with the target hardware.

Some of the features of the debugger can be listed as follows:

i. **Real time display of resources**: The debugger provides the display function that reveals the usage of system resources. It is possible to read the target memory while the application is still running in a non-intrusive manner.

ii. **Kernel / Application Debugging**: The debugger also helps us to debug the initialization phase, drivers, kernels and interrupt routines. It supports MMU to inspect both physical as well as logical address space.

iii. **Equipped with Run and Stop modes**: The various libraries and application bugged in "run-mode" in which case everything else is running except the application. It can also support the "run and stop mode" integrated in to one interface.

iv. **Task related breakpoints**: We can set break points using the software such that it is related to a specific task. Here, the break points are activated only if the particular task is enabled.

v. **Task context display**: The context of a task involves the program counter, the variables used, the register related windows and the function call stack. It is possible to switch to another task and view its contents as well while both the task are still running.

vi. **Dynamic Thread Performance Measurement**: As the OS is running on the target device, the debugger software evaluated the current running task and collects the results. The results include percentage of resources used by the task. It is represented graphically and updated permanently for later analysis.

vii. **Linux specific display of analyzer listing**: The data recorded in real time can be stored in a buffer. It can be later displayed and interpreted specific to a particular OS.

viii. **Static evaluation and graphic display of task run times**: The behavior of the system can be observed such as the analysis related the time duration that a task has run and also the switching of contexts among different tasks out of the trace buffer can be graphically represented.

ix. **Task related evaluation of function run times**: The flow of functions can be monitored both statically and graphically. The evaluation of function calls and function run times can be done dependent on the actual task under run.

x. **Static evaluation and graphic display of task run states**: The software package facilitates the tabular and graphical representation of the status of the tasks. It helps in prioritization, critical path, etc.

C. Synthetic benchmarking using Dhrystone

The dhrystone benchmark is a test for the general purpose performance of a system. The significance of the dhrystone benchmarking strategy is to provide an indication of the integer performance of a computer. This simple benchmark is proven to be very resilient and self-contained. It is easy to use and understand and also well documented. It is particularly used in embedded system environment. Dhrystone can be used to measure the performance of computers or the efficiency of different compilers that compile and run on the same machine. It is a test for integer performance of the system and care must be taken to ensure there are no floating point operation involved.

As a first step, we need to download the source code, there are two c files and one header file to be downloaded. Next, we need to compile the source code. It is noteworthy to mention that compiler optimization such as function inlining and multi file compilation are not allowed. Once the code has been downloaded, we have to modify the code as per our requirements. The modification includes changing of the clock rate according to the value specified on the target, changing the desired number of runs, etc. Once we compile and run the code, we get an output as shown in Fig. 3.

```
Dhrystone Benchmark, Version 2.1 (Language: C)
Program compiled without 'register' attribute
Please give the number of runs through the benchmark: 1000000
Execution starts, 1000000 runs through Dhrystone
Execution ends
Final values of the variables used in the benchmark:
Int_Glob:            5
        should be:   5
Bool_Glob:           1
        should be:   1
Ch_1_Glob:           A
        should be:   A
Ch_2_Glob:           B
        should be:   B
Arr_1_Glob[8]:       7
        should be:   7
Arr_2_Glob[8][7]:    1000010
        should be:   Number_Of_Runs + 10
Ptr_Glob->
  Ptr_Comp:          65288
        should be:   (implementation-dependent)
  Discr:             0
        should be:   0
  Enum_Comp:         2
        should be:   2
  Int_Comp:          17
        should be:   17
  Str_Comp:          DHRYSTONE PROGRAM, SOME STRING
        should be:   DHRYSTONE PROGRAM, SOME STRING
Next_Ptr_Glob->
  Ptr_Comp:          65288
        should be:   (implementation-dependent), same as above
  Discr:             0
        should be:   0
  Enum_Comp:         1
        should be:   1
  Int_Comp:          18
        should be:   18
  Str_Comp:          DHRYSTONE PROGRAM, SOME STRING
        should be:   DHRYSTONE PROGRAM, SOME STRING
Int 1 Loc:           5
        should be:   5
Int_2_Loc:           13
        should be:   13
Int_3_Loc:           7
        should be:   7
Enum_Loc:            1
        should be:   1
Str_1_Loc:           DHRYSTONE PROGRAM, 1'ST STRING
        should be:   DHRYSTONE PROGRAM, 1'ST STRING
Str_2_Loc:           DHRYSTONE PROGRAM, 2'ND STRING
        should be:   DHRYSTONE PROGRAM, 2'ND STRING
Microseconds for one run through Dhrystone:   24.6
Dhrystones per Second:                        40600.9
```

Fig. 3: Output of Dhrystone benchmarking test.

The result Dhrystone per second is calculated using the formula:

Dhrystone per second = Number of runs / Execution time

In order to obtain a valid result, it is recommended that the test runs for at least two seconds. The next step involves converting the Dhrystones per second to DMIPS. For this, we need use the following formula:

DMIPS = Dhrystones per second / 1757

Here, 1757 is a constant that is obtained from the dhrystone score of VAX 11/780 at 1 MIPS/second. It is usually desirable to represent the DMIPS as a function of frequency of the target board. So the final step is to divide the DMIPS by the target board frequency.

DMIPS/Mhz = DMIPS / MHz

D. Static code analysis using Tics

Over the years, there are countless number of scenarios where in the system is forced to halt due to faulty software release. The true quality of the software is determined by a) the number of defects found after the release b) the fatality of the release c) the work required to resolve the issues. If it was possible to ensure the quality even before the release, then it would be very beneficial with respect to the time and effort utilized by the developers.

Luckily, these issues are addressed and “Tiobe Tics” provides the necessary tools to perform the static code analysis so that we can be sure that the release does not fail. There are many metrics to be considered to analyze the quality of the software. The Tiobe tics or shortly known as tics uses 8 most commonly used code quality metrics in the industry. The metrics can be listed as follows:

i. **Code coverage**: It is a common practice for the developers to perform unit test during the development cycle of a product. The unit tests are used to check a particular part of the system such as a single program. The results are compared to the expected values. The code coverage checks whether all the lines are code have been executed. The higher the coverage better is the quality.

ii. **Abstract interpretation**: This metric is synonymous to the reliability of the system. These abstract interpretation tools are capable of determining every possible error pertained to the control flow of the program. Examples include divide by zero, null pointer dereference, etc. The errors are detected by traversing all possible paths. The errors found are severe and may lead to crash.

iii. **Cyclomatic complexity**: This metric represents the number of independent paths in a program. For example, every time there is an “if” statement, one extra code path is added. A higher cyclomatic complexity indicates that the program is difficult to understand and hence more test cases are required.

iv. **Compiler warnings**: This metric indicates the reliability and transferability aspects. It is a well-known fact that the programs are compiled before running. When compiled, we are shown the errors and the warnings. However, the warnings may be ignored. Sometimes the warnings are a sign of program flaws

v. **Coding standards**: A coding standard is a set of rules that engineers should follow. These coding rules are about known language pitfalls, code constructions to avoid, but also about naming conventions and program layout. Since coding standards usually contain many different rules they can be mapped to most quality attributes. Most uses concern "Maintainability" and “Reliability”, but there are also rules available for “Transferability” and “Performance Efficiency”.

vi. **Code duplication**: Sometimes, the developers copy some part of the code instead of generalizing it. This lead to the problem where other parts of the program also need to be changed. It leads to code rework and affects the maintainability.

vii. **Fan out**: It is obvious that the program is divided into multiple modules. Each module depends on other modules in order to function properly. When one instance depends on many modules then it is termed as high fan out. It affects the maintainability of the program and less modifiable.

viii. **Security**: The security of a software tells us how vulnerable it is for security leaks and unauthorized access. Buffer overflows are an example of security leak.

Based on the above mentioned factors, a code compliance factor will be generated once the program is analyzed with tics. The weightage of the factors is as per Fig 4. The result includes the message of the error, the line number, the reason behind the error along with the synopsis, file name, rule id that has been violated and summary of the analysis. It basically classifies the various errors into various levels. It is the responsibility of the developer to resolve all the critical errors to ensure good quality release.

Metric	Weightage
Code Coverage	20%
Abstract Interpretation	20%
Cyclomatic Complexity	15%
Compiler Warnings	15%
Coding Standards	10%
Code Duplication	10%
Fan Out	5%
Security	5%

Fig. 4: Weightage of factors.

5. CONCLUSIONS

In this paper, we have seen what a substation automation system is all about. We started out with the basic architecture and then the communication protocol and three different levels namely station level, bay level and process level were discussed. The main motive of this paper was to throw some light on the quality assurance metrics that companies employ to ensure safe and reliable working of the product they deploy. In this regard, we have come across the real time debugging, static code analysis and dhrystone benchmarking. We have discussed each metric briefly for understanding purpose.

As far as future enhancement is concerned, it is recommended to avoid dhrystone benchmarking as much as possible. This is because it is obsolete. Other benchmarking strategies such as SPEC should be considered. The only reason why dhrystone is popular is because it is free. The other improvement areas include adopting newer and better hardware such as processor and sensors. The latest technology trends such as IoT is expected to be implemented in the near future.

ACKNOWLEDGEMENT

This study and research is carried out in R.V. College of engineering, Bengaluru, Department of Computer Science and Engineering, under the guidance of Dr. Nagaraja G S, I also thank our Head of the Department Dr. Ramakanth Kumar P and the principal of the institution, Dr. K N Subramanya for providing us all the required facilities and suitable environment to successfully complete this research.

REFERENCES

[1] Mateusz Kowzan, Patrycja Pietrzak, "Continuous Integration in Validation of Modern, Complex, Embedded Systems", IEEE/ACM International Conference on Software and System Processes (ICSSP), 2019
[2] Valtari, Verho, Hakala-Ranta, Saarinen, "Increasing cost-efficiency of substation automation systems by centralised protection functions, CIRED, paper 0764
[3] Pitambar Jankee ; Kehinde Awodele, "Design Of An Iec61850 Based Substation Automation and Protection Panel : Including peer to peer relay communication", 2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa, 2019
[4] Mayada Daboul ; Jaroslava Orságová, "Interoperability testing for IEC61850 based substation automation system", 2018 19th International Scientific Conference on Electric Power Engineering (EPE), 2018.
[5] Magda A. Silvério Miyashiro ; Maurício G. V. Ferreira, " Factors to be considered for the adaptation of "cyclic process" (CMMI-DEV level 2) — In the development of embedded components", 2016 SAI Computing Conference (SAI), 2016.
[6 G. M. Asim Akhtar, Muhammad Sheraz, Ali Safwan, M. Akhil Fazil, and Firas El Yassine, "Modern Power System Automation for Transmission Substations", Schweitzer Engineering Laboratories, Inc.
[7] Sudeep Balan, C S Lajitha, Joseph Mathew, L R Sreedhanya, Vijaya Bhaskara Rao "Design of A Novel IEC 61850 Based Merging Unit for Substation Automation "2018 International CET Conference on Control, Communication, and Computing (IC4)" Electronic ISBN: 978-1-5386-4966-4
[8] Rudy Alexis Guejia Burbano ; Martha Lucia Orozco Gutierrez ; Jose Alex Restrepo ; Fabio G. Guerrero "IED Design for a Small-Scale Microgrid Using IEC 61850" IEEE Transactions on Industry Applications (Volume: 55 , Issue: 6, Nov.-Dec. 2019)
[9] K N Dinesh Babu , P K Garzava , Isaac Ramalla"Design of A Novel IEC 61850 Based Merging Unit for Substation Automation "2019 IEEE Asia Power and Energy Engineering Conference (APEEC) ,Electronic ISBN: 978-1-7281-1590-0
[10] Power System Scada and Smart Grids by Mini S. Thomas and John D. McDonald., I edition 2015.