AERO INSTRUMENTS & AVIONICS, INC.

Aero Instruments & Avionics, Inc.
7290 Nash Road
North Tonawanda, NY 14120

Date:	October 15, 2018
To:	Tom Campbell Head of Engineering Aero Instruments & Avionics, Inc.
From:	Amit Blonder, ATE Software Engineer
Subject:	Recommendation Report for Decreasing Failures from ATE Software Updates with Improved Software Development Methods

Attached is my report for the study, "Decreasing Failures from ATE Software Updates with Improved Software Development Methods: A Recommendation Report". This report was conducted in order to assess the recurrent problem of software failures impacting technician performance after updates to the RADA ATE software.

This study was primarily conducted using primary research of the present state of the software at Aero. I have interviewed three technicians that use the software developed by the ATE software team in order to garner reviews of the effectiveness of the software. I have also employed the use of the ATE usage history recorded in our database server to find patterns of downtime corresponding to software updates. Finally, I researched software development methods that could be applied to small teams, and assessed the viability in this company, along with the other software teams.

My main finding was that the sporadic updates and consequent bugfixes disrupt the workflow of several, but not all, technicians. For stations that do update frequently, new bugs may get introduced into the software and cause unexpected failures before they can be fixed. This has been found to be attributed mostly to the lack of a version control structure to the software development of the ATE software.

It has been found through my research these types of failures could be averted by reorganizing using an Agile workflow, which includes preset release dates and an organized version control system, usually involving Git. I have also found that it is feasible to install a private repository for the TEW software at Aero, and so I recommend we instantiate an improved software development procedure for the ATE software.

I appreciate your time taken in reviewing my study and hope that my findings are found reasonable and my recommendations viable. If you have any questions regarding my research or the execution of my recommendations, please feel free to contact me at (716) 730-9924, or through my email at amit@aeroinst.com.

Decreasing Failures from ATE Software Updates with Improved Software Development Methods: A Recommendation Report

Prepared For: Tom Campbell
Head of Engineering
Aero Instruments & Avionics, Inc.

Prepared By: Amit Blonder
ATE Software Engineer

October 15, 2018

Aero Instruments & Avionics, Inc.
7290 Nash Road
North Tonawanda, NY 14120

Abstract

"Decreasing Failures from ATE Software Updates with
Improved Software Development Methods: A Recommendation Report"

Prepared By: Amit Blonder
ATE Software Engineer
Aero Instruments & Avionics, Inc.

A study of the organizational method of current ATE software development team at Aero was conducted in response to a trend in software-related failures on the RADA stations. Technicians were found to periodically halt their workflow after software updates were conducted. The goal of this study was to ascertain the underlying cause of the software failures, and then recommend a possible improvement that could be made to mitigate these failures. This study was completed principally using primary research methods, which include interviews of the various technicians and related departments, as well as secondary research into possible solutions. In general, the interviews and research data revealed a strong correlation between frequently updating the software and experiencing consequent failures unrelated to the units. These failures appear to rise due to bugs being introduced into the software with no verification time before they are introduced into the live RADA stations. Possible solutions to these failures include maintaining a consistent development logbook and segmented development procedure. I recommend we transition to an Agile methodology, using GitHub changelogs and a ZenHub development planning board.

Keywords: RADA, ATE, software failures, software updates, regression faults, Agile

Table of Contents

List of Figures

Introduction

This report was developed in order to assess the current methods of software organization and version control implemented at Aero Instruments & Avionics, Inc., and their effect on software effectiveness. During my two years as a software engineer at Aero, I've discovered that technicians are frequently delayed in their work by various hardware faults and software errors. While there are many factors that play into the delays in processing of units and consequently the hours involved in repairing them, this study focuses on the role the software of the RADA ATE's has had on them. Specifically, this study targets the topic of software organization and management, and how it propagates to end-user inefficiencies and disruptive faults.

At Aero, we offer testing and repair services to airliners who discover faulty units in their airplane or simply need to test their units. The work, therefore, is often time-sensitive, since the airplane must be grounded if there is no replacement for that unit available while it is being sent out for testing. Some units are tested manually using front panels, while many are tested on Automatic Test Equipment (ATE's), which come equipped with an oscilloscope, a DMM, a Switching Matrix and many other measurement instrumentation and varying power supplies. Ideally, testing on an ATE can reduce the test time from days to hours. Of the ATE's in the facility, the newest and most powerful ones are the 5 RADA ATE's, which include industrial computers that recently switched to using Windows. Those computers run our own proprietary Test Executive (TEW) software, which in turn executes the instructions for the Test Program Sequence (TPS) for the particular Unit Under Test (UUT). We shall henceforth use both the acronyms and full names of the aforementioned software and hardware interchangeably in the report.

This report seeks to ascertain the impact of the current method of software organization in the facility on the problems appearing both in the test software and test sequences. The software for the TEW, as well as the Test Program Sequences, are stored on the local network and maintain only a singular working version with a handful of backups. Therefore, the current organization is such that it becomes difficult to see what is being worked on, tell when and how changes are made, and to rewind mistakes once they are found. This leads to a recurrent theme of "Regression Faults", where functionality updates to the code break other, unrelated parts of the software inadvertently. In addition, the separation of code between the different teams leads to the small teams having little communication with each other regarding their software, and as such it becomes difficult to integrate software from one team with another's. This can lead to further disruptions in the technicians' workflow when problems arise.

To review this issue, I focused my research into the following objectives:

1. Assess technician satisfaction of reliability of the present software
2. Determine patterns of ATE usage times throughout weeks with many software updates and those with fewer updates
3. Query the other software development teams at Aero for their software organization practices
4. Analyze alternatives to the current software organization method, specifically version control, and their viability in this company

The results of the assessment of technician satisfaction generally show contentment towards the present state of the software. However, the overall consensus is that updates to the software are often dreaded, since they frequently accompany new faults that cause them to stop their work and refer to the ATE programmers directly. In addition, it turned out that faults in the software have caused many instances of ATE downtime while bugfixes were being implemented. Most of these faults were regression faults, where they appeared as the result of an update, and so were preventable with a version control system.

It has been found through my research that small teams are best organized using an Agile workflow, which includes preset release dates and an organized version control system, usually involving Git. I have found that it is feasible to install a private repository for the TEW software at Aero, but less so for the TPS's. As such, I recommend that the TEW software be migrated to a private GitHub repository, and that a more consistent software modification process and release dates be adopted.

The following section provide more information on the methods used in the study, the results of those methods, the conclusions drawn from the study and research, and the recommendations moving forward.

Methods

In order to effectively evaluate the shortcomings of the current software organization method at Aero, my research was broken down into the following objectives:

1. Assess technician satisfaction of reliability of the present software
2. Determine patterns of ATE usage times throughout weeks with many software updates and those with fewer updates

3. Query the other software development teams at Aero for their software organization practices
4. Analyze alternatives to the current software organization method, specifically version control, and their viability in this company

Objective 1. Assess technician satisfaction of reliability of the present software

Primary to any study on present flaws is the polling of the dissatisfaction of the users of that system. By demonstrating that there exists some dissatisfaction or distress about the results of the software system, then we can show that it has a need to be improved. Technician satisfaction was polled through one-on-one interviews in order to select the most frequent users of the ATE's and to garner more detailed insight about when and where failures occur, how they affect each technicians' workflow, and how they could be remedied. As each technician incorporates the ATE's differently into his overall workflow, I found it more suitable to interview them individually.

Objective 2. Determine patterns of ATE usage times throughout weeks with many software updates and those with fewer updates

Correlating the times of when software updates were released with the usage logs of the ATE's that were already available could provide some insight as to how much the ATE's are used after new updates are released. Showing more "Idle" or "Available" statuses indicate disuse of the ATE's, which would demonstrate a reduction in productivity and, in turn, revenue production as a result of the software updates.

Objective 3. Query the other software development teams at Aero for their software organization practices

Some of the software updates for the TEW are concerned with interoperability with the other software teams, such as access to the database server, as well as the aforementioned usage logs. Therefore, if there exist communication deficiencies between the software teams concerning their code, then miscommunication and misunderstanding could lead to more faults later on. So each of the small software teams, ATE software, IT software and Data Loader software, were queried as to their method of code organization and how they interface with the rest of the system.

Objective 4. Analyze alternatives to the current software organization method, specifically version control, and their viability in this company

In order to conceive alternatives to the current software organization method and offer a recommendation to replace it, I've completed a literature review of peer-review articles on how to organize software teams. This research focused on simple methods for small teams, with a short transition period and effective practical results. Thorwart's (1998) article, for instance, demonstrates how to transition a small programming team into an effective quality assurance model. The practices demonstrated in the results of the research were compared to how the different software is developed and used in the company to assess how viable it is to conduct this transition.

Results

This section presents the results of the aforementioned research methods, focusing on the responses and data most relevant to this study. The outcomes of the interviews, data extraction and secondary research is outlined below.

Objective 1. Assess technician satisfaction of reliability of the present software

For this study, three technicians working in different sectors were interviewed in order to provide a diverse response set. The first technician, Kevin, is a new employee who joined only 7 months ago, and he works at RADA #3. According to Kevin, the TEW software has been working really well, and he never had to stop while he's been working on it. RADA #3, however, is infrequently updated, and Kevin admits to skipping updates since he wasn't aware whether or not he was allowed to execute them.

The second technician is Brian, who is a veteran employee who operates mainly RADA #1, and is experience in repairing units. Brian has been updating frequently, had has been having frequent errors which cause him to stop his work and present the bug to the software team. In the case of the updates, he elaborates that many times they either don't work, or bring about new problems. Since there haven't been many updates recently, however, he comments that "it's working now that you're not messing with it!"

Brian also offered his own observations into what goes wrong with the software updates, and even how to remedy them. He notes that there have been cases where his unit was running fine one day at 3pm, and when he comes the following morning, the unit has errors. Consequently, he has to go through the test and figure out if there is a new problem in the software by detailed inspection or cross-referencing with another unit, before he can bring it up to the software team. He posits that if he'd known there was a change, he could be on the lookout for what was changed and if that affected any of his tests, he could give feedback much quicker about the change.

In addition, Brian brings up another previous issue, where the TPS backup of the Anti-Skid Control Unit (ASCU) is too old. The current one got corrupted at some point, and the software developers had to go back and take the old one, and then re-apply all modifications and fixes to it. Brian also offered some additional suggestions, which will be discussed in more details in the "Recommendations" section.

The last technician that was queried is Jeff. He does not work using the TEW at a RADA station, but rather he tests units using the Kollsman Pressure Controller. A new software GUI was designed for him and another technician, which was designed to replace a program running on old hardware. The development for this software proved effective, since it was done in parallel to an already working older system that could be relied upon as backup. In addition, the testing cycle was short, so it was finished quickly, and no new problems have sprung up since the three months it has been in place, since no updates have been made to it since then.

Objective 2. Determine patterns of ATE usage times throughout weeks with many software updates and those with fewer updates

Due to the lack of a set organizational method for the current software practice, a detailed software version history was not found. There is, however, a detailed ATE usage history, which tracks all RADA ATE's statuses, measured in hours/day. We are mostly concerned with "Running" and "Waiting for Manual Intervention" status codes, which represent periods of activity in the stations. Figure 1 is such a graph taken over the period since it was installed in late 2017.

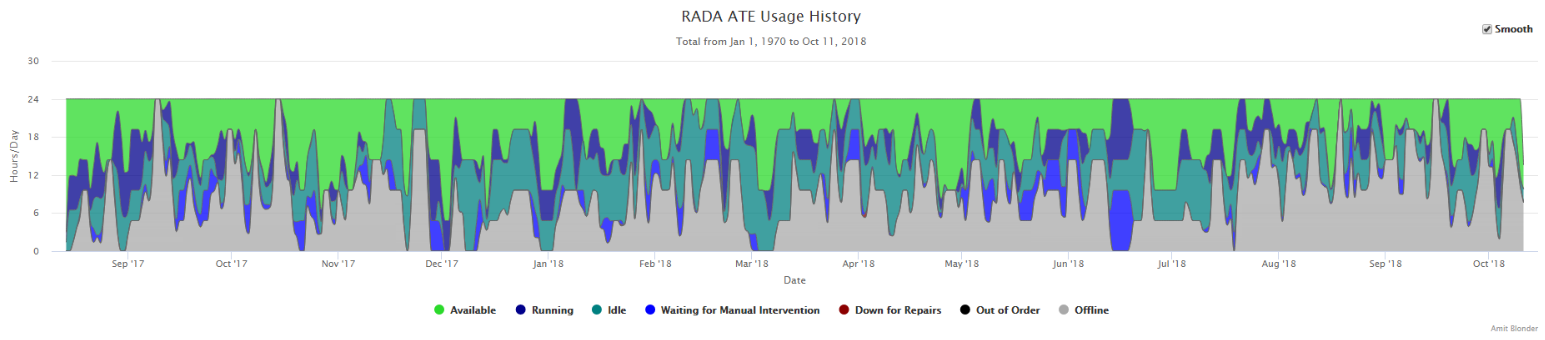


Figure 1: Total RADA ATE Usage History

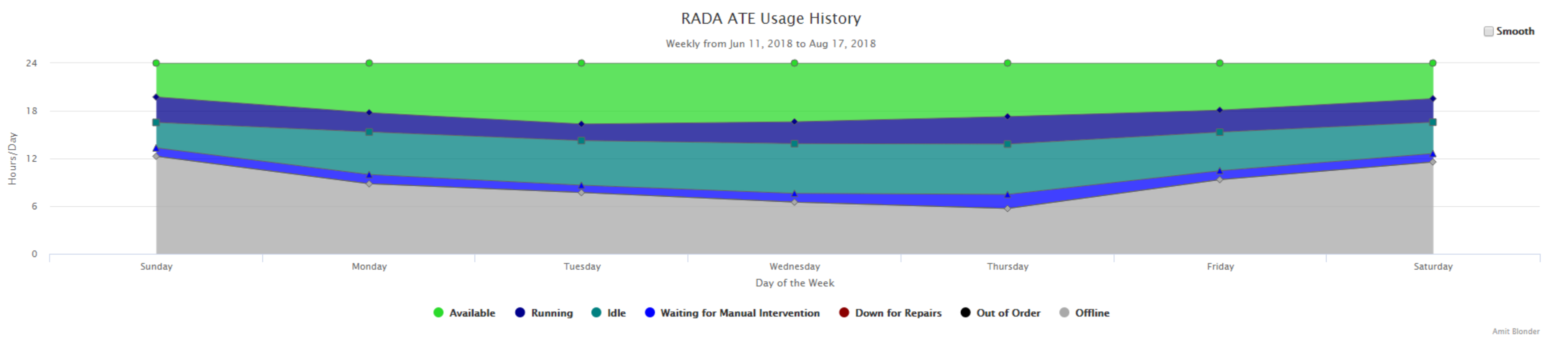


Figure 2: Weekly RADA ATE Usage History, 6/11/2018 – 8/17/2018

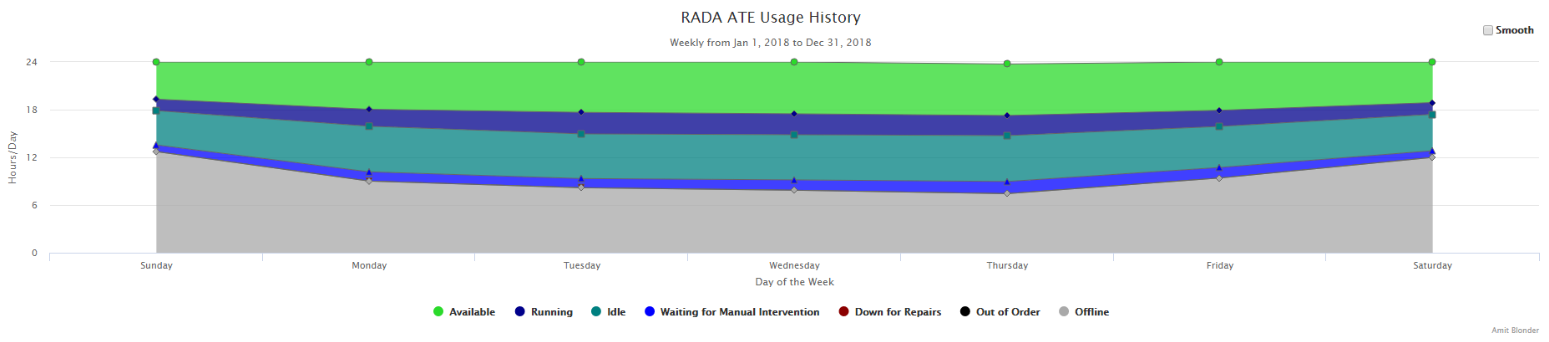
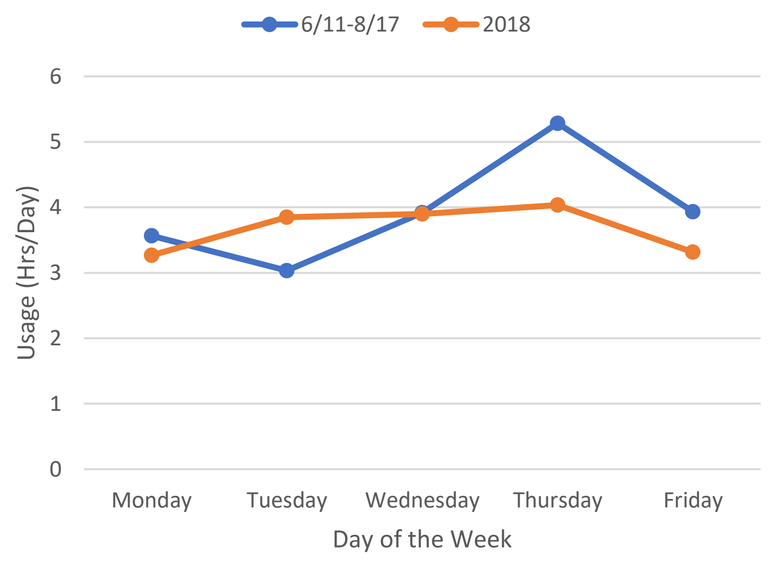


Figure 3: Weekly RADA ATE Usage History, Year of 2018

The total ATE usage graph shows sporadic dips in usage hours and several prolonged periods of unavailability of roughly 3 days.

The period from 6/11 – 8/17 was one of an abnormally high software update rate. During this period, there was more downtime in the middle and early work week, as shown in Figure 2. Usually, there is more station usage during the midweek, as shown in Figure 3, measured across all of 2018. The two time periods are compared in Figure 4, where it can be seen that the ATE's were less active earlier in the week, when software changes are usually tested and implemented.

Figure 4: Weekly RADA ATE Usage (Running + Waiting for Manual Intervention)



Objective 3. Query the other software development teams at Aero for their software organization practices

Scott, the head of the IT department at Aero, elucidated how the other software teams organize their software. Specifically, his IT team does not use a Version Control System (VCS) such as GitHub. Rather, they use a changelog which they "comment the Bejeebers out of". In addition, their code is stored on the local network, and the database exists on their own SQL server, all of which is protected using Windows Authentication. The DataLoader team, on the other hand, uses Microsoft's Foundation Server, in conjunction with the Visual Studio IDE. Incidentally, Visual Studio is used in all of the ATE, IT and DataLoader software teams at Aero.

Objective 4. Analyze alternatives to the current software organization method, specifically version control, and their viability in this company

Thorwart's article on the usage of the AMETIST process provided insight into how quality assurance can be incorporated into a small software development team. His report focuses on how the "heroic" method of programming that was use for ROSE Informatik was not scalable, and had to be transformed using the AMETIST process in order for software development growth to be viable. The improvements involved were:

- Specifying the desired software characteristic for Quality Assurance
- Modifying the software development process using proven, standard procedures, such as the SynQuest Assessment, to increase motivation and stability
- Incorporating a rigid development procedure with a development logbook to track version changes
- Continuous documentation of code to disperse knowledge from an individual to the team

As a result, Thorwart found that there were 60% fewer "delivered" errors, activities are delayed less often, and general communication with both staff and customers has increased (Thorwart, 1998). The specified changes, such as a development logbook and using a standard software development procedure, is already in place at Aero, with the IT department's changelog and the DataLoader team's Foundation Server, respectively. As such, they are viable to incorporate into the ATE software development team.

Conclusion

From the above results acquired from my research, I am able to derive several conclusions about the state of ATE software development at Aero. Firstly, it has been found that periods of frequent updates to the software cause failures for technicians when they try to test units, which introduces delays and lost revenue from inactive RADA stations. The updates are not well communicated to the technicians, and cannot be traced back in a version log, so they frequently lead to problems. In addition, unlike the other software development teams at Aero, the ATE team does not have a set development procedure, and so is much more likely to create regression faults and would be less able to cooperate with those teams. Furthermore, through secondary research, it has been found that a development logbook and a rigid, standard development procedure is key to assuring quality code from a software development team of any size. I've concluded that we would be able to incorporate such changes into our software development architecture, using a Version Control System and changelog similar to what is already in place elsewhere at Aero.

Recommendation

In order to alleviate the problems present in our software development method, I recommend that we transition into a standard Agile software development architecture, using a private GitHub repository and using ZenHub for a Scrum work board. This transition will involve the migration of the software off of the network, as well as developing regular release dates with communication with the technicians. In addition, version control for the TEW software will be handled using GitHub, as well as the changelog and continuous documentation.

Furthermore, I propose that we follow Brian's recommendations as to how we can improve the software update process. Firstly, we would begin to follow a schedule of segmented update and testing. This would involve adding new features to test them, and then certify them. Once it is validated that those features are working, then we can update all of the machines. Direct update would then be reserved only for certified features and bugfixes. This would be accomplished using GitHub branching, which allows the team to create a separate branch for every new feature, and only merge it into the master branch once the change has been tested and verified.

The aforementioned changes are not expensive, as the only expense is the private GitHub repository for $5 a month. The only requirement is a time investment from the software developers into the transition, but I believe this transition will save more time and expense in the future.

Glossary

Aero – Shorthand for Aero Instruments and Avionics, Inc.

Agile – Agile software development: an approach to software development based on adaptive planning and flexible response to change.

AMETIST – Adoption of Methods, Engineering Techniques, Inspections and Software Tools: A process improvement experiment (PIE) conducted by Thorwart and his team to transform a “heroic” way of programming into an effective team-based one.

ATE – Automatic Test Equipment: An electronic machine composed of a myriad of instruments, power supplies and connectivity modules designed to test many different aircraft units.

GitHub – A version control platform based on the Git framework that allows for code repositories to be stored online with a history of previous versions and branches.

RADA – Israeli company that used to construct ATE’s for aerospace; used in this context to refer to the five RADA ATE stations at Aero.

Scrum – An adaptation of Agile software development with short feature-based software development cycles called “sprints”.

TEW – Test Executive for Windows: proprietary software designed to run the test sequences on the RADA ATE’s, and is the focus of the report.

TPS – Test Program Sequence: test sequence adapted from each UUT’s Component Maintenance Manual (CMM) that is executed by the TEW in order to run the appropriate tests for that unit.

UUT – Unit Under Test: the aircraft unit that is being tested for defects.

Version Control System – The management of changes to documents and computer programs which tracks the history of revisions so that they can be referred to or restored at a later date.

ZenHub – A GitHub-based platform implementation of the planning board used in Scrum.

References

Thorwart, K. 1. thorwart@rose. d. (n.d.). Retrieved from http://search.ebscohost.com.gate.lib.buffalo.edu/login.aspx?direct=true&db=aci&AN=13439766&site=ehost-live&scope=site

Appendix

Appendix A: Interview Outline

(Primer) How comfortable do you feel with the present reliability of the TEW and reporting software?

How often do you find yourself having to stop what you’re working on due to a software failure?

What do you do, usually, when there’s a problem with the software? Are there other tasks you can do while it’s being fixed?

How effective do you feel the current error reporting process is?

(Miscellaneous) Are there any other problems that have come up during your usage of the software?