AERO
INSTRUMENTS
& AVIONICS, INC.

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

Date: November 2, 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 sporadic updates and consequent bugfixes disrupt the workflow of several technicians. For stations that do update frequently, new bugs 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 using a changelog and a regular development schedule. Consequently, I recommend that we instantiate an improved software development procedure for the ATE software using Agile methodology and a Git repository.

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
Aero Instruments & Avionics, Inc.

November 2, 2018

AERO
INSTRUMENTS
& AVIONICS, INC.

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

No table of figures entries found.

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. Many of the units are tested on Automatic Test Equipment (ATE's). Of the ATE's in the facility, the newest and most powerful ones are the 5 RADA ATE's. 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).

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 research was focused on the presence of "Regression Faults", where functionality updates to the code break other, unrelated parts of the software inadvertently.

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

1. Assess technician satisfaction of reliability of the present software
2. Find patterns of ATE usage times correlating to software updates
3. Determine current software development practices in other teams
4. Analyze alternatives to the current software organization method

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. 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. Find patterns of ATE usage times correlating to software updates
3. Determine current software development practices in other teams
4. Analyze alternatives to the current software organization method

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 with three technicians 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. Find patterns of ATE usage times correlating to software updates

Correlating the times of when software updates were released with the usage logs of the ATE's that were already available provided 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. I used our database's detailed ATE usage history, which tracks

all RADA ATE's statuses, measured in hours/day. This method was chosen to highlight the quantitative effect of the faults.

Objective 3. Determine current software development practices in other teams

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

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

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 ("Technician Satisfaction Interviews", 2018). 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.

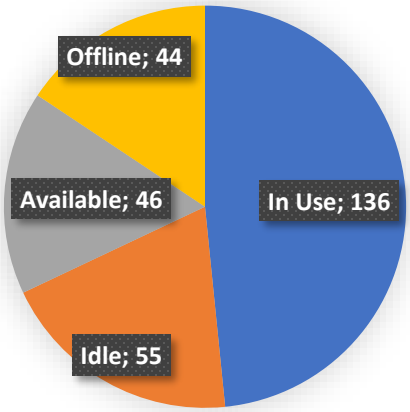
The second technician is Brian, who is a veteran employee who operates mainly RADA #1, and is experienced in repairing units. Brian has been updating frequently and has been having frequent errors which cause him to stop his work and present the bug to the software team. In the case of the updates, he 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!" ("Technician Satisfaction Interviews", 2018). He posits that if he'd know 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.

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, using a new software GUI which was designed for his station. The development for this software proved effective ("Technician Satisfaction Interviews", 2018), since the testing cycle was short, so it was finished quickly. No updates have been made to the software since the three months it has been in place, and no new problems have sprung up.

Objective 2. Find patterns of ATE usage times correlating to software updates

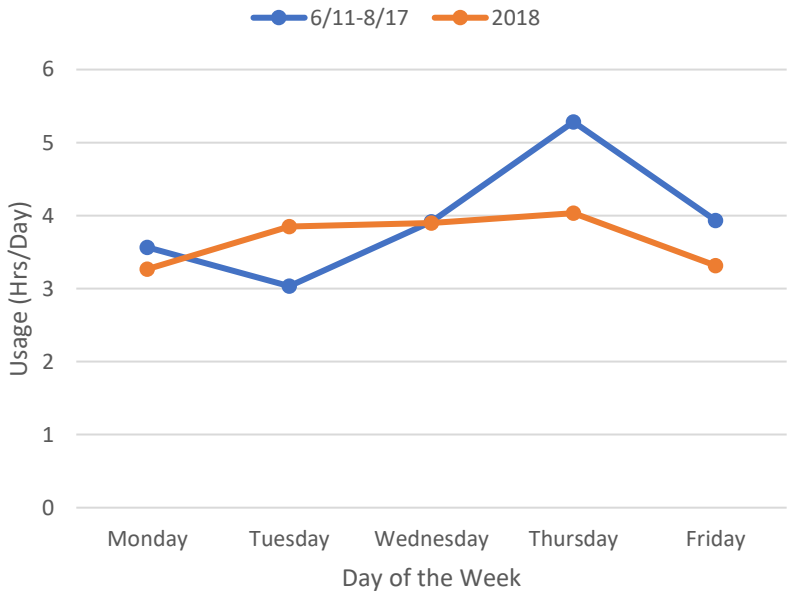
RADA #1 was selected to be tested for prolonged periods of unavailability or disuse, spanning the whole day. Figure 1 below shows the proportion of days in which the ATE was either "Offline", "Available" or "Idle" for the whole day, out of the measured 281 days from 2018. For more than half of the days, RADA #1 was unused for the whole day.

Figure 1: Total ATE Usage for RADA #1 for 1/01/2018 - 10/11/2018 by # Days in Status



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



Objective 3. Determine current software development practices in other teams

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" ("Aero Software Structure Interview", 2018). 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

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, using the following improvements:

- Use proven, standard procedures to increase motivation and stability
- Incorporate a rigid development procedure with a development logbook to track version changes
- Continuously documentat the code to disperse knowledge from the 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, are 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 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.

Regression Fault – An error in software introduced as a side-effect to an update incorporating a change to an unrelated section of the code.

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

[ATE Usage History for Aero RADA ATE's]. (2018, October 11). Unpublished raw data.

K., B., & J. (2018, October 11). Technician Satisfaction Interviews [Personal interview].

S. (2018, October 11). Aero Software Structure Interview [Personal interview].

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: Technician Interviews

1. Jeff - PressureControllerWPF
2. Kevin - Rada #3
3. Brian - Rada #1

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

1. Right now it's very reliable, only problems have been human error and has been working well (note: no updates have been made for a while)
2. Has been working really well! (quite impressed, transitioning from high voltage tech work)
3. Notes below

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

1. Never really had to stop since old system still existed.
2. Never had to stop while he's been working on it (note: new, 7mo.)
3. Rather often while units are being worked on

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?

1. Usually, switch to the older software, bring it up when next meet with dev.
2. N/A
3. Have to stop working on the unit and work on something else…. Or **work *around*** the bug in the meanwhile

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

1. N/A
2. Never really had to use it.
3. Walk upstairs and hand you another screenshot of another bug!

Miscellaneous

1. When first creating it, there were a few problems (revealed filter problem)
2. Didn't know about whether he should be updating or not - gets updated infrequently
3. Notes below

Brian

"It's working now that you're not messing with it!"

- Updates many times bring problems (he's the one who's been most vocal about this)
- Many times, when there *is* an update, it doesn't go through with it (though it was fine this morning)

Suggestions

1. Segmented update and testing - add new features and test them, certify them, and once we know those are working then we can update all the machines, direct update should only be certified features and bugfixes
2. Tell them! "Hey, we updated this last night" - Unit was running fine yesterday at 3pm, come in this morning, and it's got errors
    1. Have to go through the test and figure out if the unit broke overnight or if there is a new problem in the software ("Half a day spent checking this unit" - go to test another unit, sees that the problems are the same, goes up to software development to notify, "Hey, this is your problem!")
    2. 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, and he could give feedback much quicker about the change
    3. Leads into version control - if we track when we make changes and what they were, we could correlate them to errors in the units and find problems faster, as well as have less hours of frustration looking for problems that aren't there
3. TPS backup of ASCU was too old - when the current one got corrupted, had to go back and take old one and re-apply all modifications and fixes to it