Requirements Specification

11/20/2017

Sponsor: Harlan Mitchell, Sys. Tech. Manager

Honeywell Aerotech

Mentor: Austin Sanders

Project Members:

Josh Baker

Rex Rogers

Emilio Sifuentes

Summer Stapleton

Accepted as baseline requirements for the project:

For the client: _____ (Signature) _____(Printed Name) _____(Date)

For the team: _____ (Signature) _____(Printed Name) _____(Date)

Table of Contents

1. Introduction

With roughly 5,000 flights in the United States airspace at any given time, the importance of keeping these planes properly functioning cannot be understated. With the potential for extensive damage of property and horrendous loss of life, there have been a number of safeguards put into effect. One such safeguard would be that of the Engine Control Units (ECUs), which can be likened to the brains of a plane's engine, able to read input from plane sensors and supply output as well. The Federal Aviation Administration (FAA) maintains high standards when it comes to the functioning of these units and requires extensive testing and certification of the ECU model before it may be put into operation. The client of this capstone project, Honeywell Aerospace, is a manufacturer of these ECUs and needs to abide by the regulations set forth by the FAA, and so needs an efficient means of running rigorous testing on them.

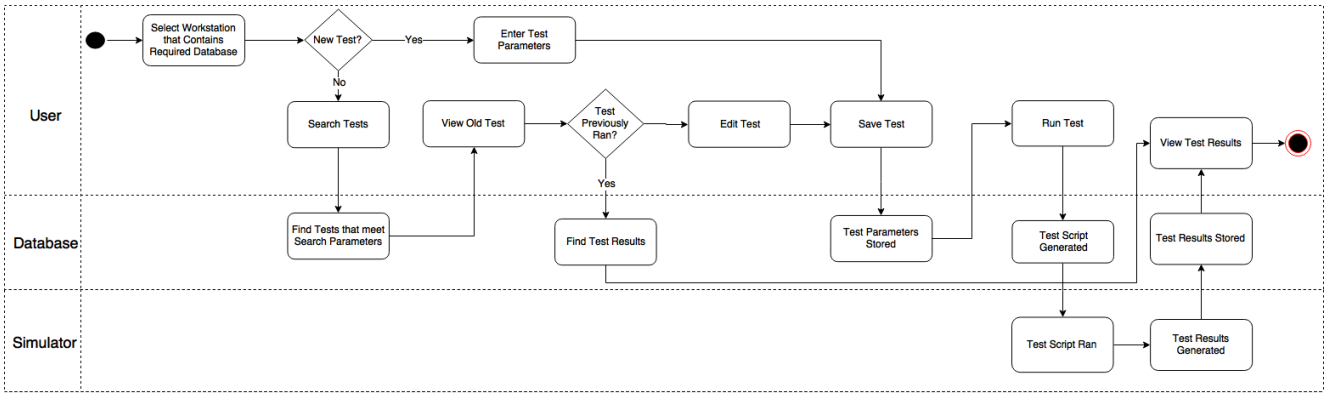
The project under discussion within this document is the Next Generation BTU (Bench Test Utilities) Proof of Concept, and the sponsor is Harlan Mitchell, a Systems Tech Manager for Honeywell Aerospace. This project consists of creating a web-based application and a centralized database that offers end-users easier interaction with the ECU testing system already in place at Honeywell Aerospace, allowing for the creation, running, storage and viewing of tests and their results from any machine with a connection to the network and a proper browser. The issue with the current system, despite its working state, is that it is outdated and tedious to use. The front end is running on outdated Windows XP operating systems, and the back end consists of separate databases running on Microsoft Access 2003. Testers and engineers must access this system from designated machines that are physically located in a particular room, and may only access certain test/results from certain machines.

In this document, the requirements of the solution will be carefully laid-out to allow for proper planning and effective implementation.

This is a Requirements Engineering Capstone Project for The College of Engineering at Northern Arizona University for the 2017-2018 academic year. The Capstone Team name is Horizon Analytics and consists of: Josh Baker, Rex Rogers, Emilio Sifuentes and Summer Stapleton.

2. Problem Statement

When testing a new engine control unit (ECU), several steps are performed in a specific order in order to ensure proper testing of the ECU. When a new design is submitted, an engineer must install the ECU at the simulator then moves to the bench test utility(BTU) to build the tests to be executed. These tests are created by generating python scripts which are then sent to the simulator which adjusts parameters read by the ECU. The results are recorded as a pass or fail based on how the ECU reacts to these changes and output to the engineer in an html format.

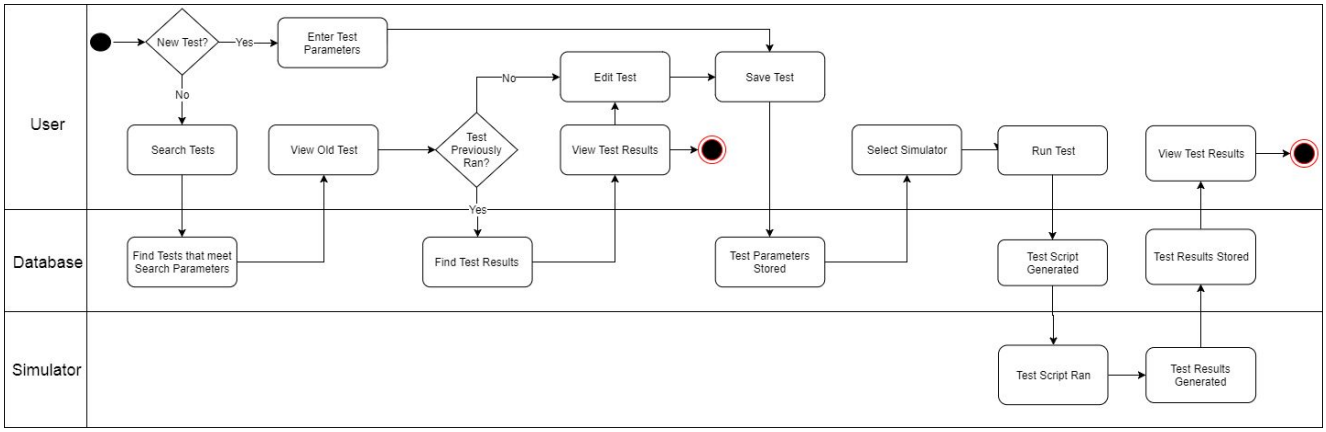


With the current workflow, once an engineer installs the ECU at the simulator and begins building the tests, that system is no longer available for use by other engineers until testing is complete. In addition, the current bench test utility utilizes legacy software which forces the current system to be operated offline for security reasons.

- Bench test utility can only be used by a single engineer at a time
- Windows XP operating system and Microsoft Access 2003 database place the system at risk
- Engineers must be physically present at the simulator to build the tests
- Engineers are not able to build tests in advance thus slowing the work flow

3. Solution Vision

To solve the problem, a web-based Bench Test Utility is being developed. It will allow engineers to connect to it via the internet, allowing them to write and run their tests from their usual workstations. Making the new BTU web-based will remove legacy dependencies that the previous version required by allowing any web-browser to connect instead of requiring old operating systems.



From the web portal users will be able to:

- Write new tests
- Save tests to a database
- Search and filter previously written tests from the database
- Retrieve previously written tests from the database
- View the tests queued and being run on each simulator
- Send a test to a simulator to be run
- View the results of any tests previously run

Additionally, the system will:

- Generate python scripts from the written tests to be executed on the simulator
- Send the results of the tests from the simulator to be stored on the database

This design keeps all the functionality of the previous iteration of the Bench Test Utility, while saving valuable Engineer time by removing the need for them to wait for a BTU to be available and by allowing them to access the BTU without getting up. Further, it allows them to completely remove the current computers running XP that are hooked up to the simulators currently.

This solution does require a small change on behalf of the client where they will need to add a network connection to their simulators since they are currently offline. The upside to this is that it allows written tests, and the results of any tests to be directly uploaded onto a central database to be checked from anywhere, instead of having tests and results be local to a given BTU.

4. Project Requirements

The requirements outlined in this section will serve as a record of expectations from Horizon Analytics. The exact means of attaining these requirements are subject to alterations as the team progresses through their execution.

4.1 BTU Application

4.1.1 Description

The BTU application is the backend of the web interface that handles the work of the website. It has all the functionality of the current BTU system that exists. It needs to handle the writing, saving, retrieving, and running of tests and test results.

4.1.2 Functional Requirements

Test Functions - A list of functions that a test is capable of running will exist.

Function Parameters - Possible values for each function that exists will be defined.

Function Scripts - Each function will have a python script associated with it.

Database Communication - The application will be able to communicate with the database.

Database Query - The application will be able to send SQL queries to the database and receive results of those queries.

Database Writing - The application will be able to write new entries into the databases tables using SQL.

Python Script Templating - The application will be able to produce a python script file that contains the function scripts put together in an order based on a written test.

4.2 Web Interface

4.2.1 Description

The web interface will allow engineers to access the testing utility from their own desk. When accessing the web interface, the engineers will have the option to search for a pre-existing test, review test results, create a new test, or run a test.

4.2.2 Functional Requirements

Browser - The web interface will be designed and tested for use with with the Firefox browser.

Dynamic Information Display - Content on the interface needs to be able to dynamically update based on user actions and database updates.

Test Search - Tests may be searched by Test Creator, Test ID, Product Type, or Name.

Test Results - Results will be viewable directly from the web interface.

Write Tests - Users will be able to create tests by selecting test parameters provided on the web interface.

Save Tests - Users will be able to provide a name for, and save tests created within the web interface.

Run Tests - The user will be able to select a simulator to run the test on, and then send the test to that simulator for execution.

4.3 Login System

4.3.1 Description

The login system will allow users to have a unique ID with which they will use to access the web interface. The login system will serve to protect the testing software and database from unauthorized access.

4.2.2 Functional Requirements

Store User - The login system will be able to store user's login information in a database.

Security - The login system will prevent unauthorized access to the testing software and database.

4.4 Central Database

4.4.1 Description

The Central Database will handle the raw data created by the Engineers via the tests and the test results. This database will be web based.

4.4.2 Functional Requirements

Parameter Storage - Test creation parameters will be stored within the database.

Test Result Storage - Test results will be stored within the database

Test Storage - Tests will be stored within the database

Test Access (web interface) - The web based application will be able to access the database for its functional purposes.

Test Access (simulator side) - The simulator side of the implementation will be able to receive information from the database for its functional purposes.

Test Results Access (web interface) - The web interface will be able to access the test results within the database for its functional purposes.

Format Tests - organize tests by a standard set by the information available within the parameters.

Format Test Results - organize test results by the information available in the results.

Find Tests - Find tests that meet search parameters.

Find Test Results - Find test results that meet search parameters

Test Parameters Tables - Database tables holding the test parameters..

Results Tables - Database tables holding the test results.

4.5 Networking

4.5.1 Description

Networking Protocols are required to have communication between each of the major components of this project. This project will deploy the use of HTTP networking.

4.5.2 Functional Requirements

Test Parameter Transference - The networking protocols will allow the transfer of test parameters from web application to database.

Test Parameter Transference - The networking protocols will allow the transfer of test parameters from database to Simulator side.

Test Result Transference - The networking protocols will allow the transfer of test results from simulator side to the database.

4.6 Non-Functional Requirements

The non-functional requirements for the Next Generation BTU Proof of concept will be defined as follows:

4.6.1 Performance

Since the Bench Test Utilities already exists in a different form, it may be used as a means to measure expected performances of the solution as far as computational speeds are concerned. It has been determined that the speeds of the following computations may not exceed 150 percent of the computational times of the current system:

- Data retrieval from the database
- Loading of tests/test results for viewing in the BTU application
- Data transfer from application to simulator, and back again

These values will be obtained by taking the average computational runtime of 10 runs for each on both systems.

Given the ease of access of the proposed solution, and therefore less labor spent in the creation and the running of these tests, there is confidence that overall performance will still increase so long as computations fall within this range.

This may be tested by simply comparing the two systems side-by-side for these processes, as both systems will have these processes in place.

4.6.2 Effectiveness

The sponsor has expressed the importance of an effective design for the solution, with this factor being a driving force behind seeking an improved system. Allowing a user to access the entire database from a single machine of their choice will logically increase the effectiveness of the workflow as users will no longer need to move to another room and search through many unconnected databases.

Of course this requirement will prove itself difficult to quantifiably measure, with the main source of measurement likely coming in the form of user feedback as well as department reviews of labor and performance.

Further discussions with the sponsor may be needed to discern reasonable and quantifiable expectations.

4.6.3 Usability

The solution must be intuitive and easy to learn in order to increase its effectiveness. It is the goal for this solution to allow for certain standards to be met with regard to the number of iterations of a particular workflow in a given space of time before a user has sufficiently learned said workflow.

The iterations for the different workflows are as follows:

- Creating tests: Users of the current BTU should be able to perform the unaided creation of a test on the new system after having performed at least ten test creations in a day, with at least twenty test creations within a week.
- Running tests: Users of the current BTU should be able to perform the unaided running of a test on the new system after having performed at least three test runs per day for five consecutive days.
- Viewing results: Users of the current BTU should be able to perform the unaided viewing of test results on the new system after having performed at least three test viewings per day for five consecutive days.

With working knowledge of how the current system functions among the current labor force, these goals should be easily realized while creating an accountability measurement of the usability of the solution.

4.6.4 Efficiency

As the application will need to handle the transfer of files to and from a number of different simulators, the database and end-user machines, with multiple users being able to access the system at any given time, it may encounter many transfers happening in parallel, it will need to accommodate this traffic without causing excessive degradation of speeds.

(Acceptable speed degradation was addressed in the *Performance* section above.)

4.6.5 Robustness

Given that the testing of these ECU units needs to be virtually error-free, the application must be robust enough to account for user error and improper database access as well any extraneous output from the simulator.

As proper certification through the FAA is required before these units may be put on the market, it is no surprise that this concern has been reiterated by the sponsor on a number of occasions, and so will need to have special care taken in its regard.

Though many measures will be taken to ensure the robustness of the solution, a final means of measuring this robustness still needs to be discussed with the sponsor as well as acceptable standards that must be maintained with regard to it.

One idea may be to run the application through a suite of tests with random errors that the application must be able to correctly identify. (Maybe not the exact values as these could be random, but just the very fact that an error has occurred and therefore the test results should not be used.) The random nature of the errors would prevent the development of specific solutions, forcing a general approach to be created that will catch a wider range of errors.

4.6.6 Maintainability

As the solution is the implementation of a simple prototype, maintainability may prove itself more important than in the current system or the full implementation of the system being prototyped. This may help to contribute to the continuous development and improvement of the proposed solution.

Discussions with the sponsor needs to be had that will clearly define a standard means of approaching and measuring the successful implementation of this requirement.

4.6.7 Documented

In order to ensure that the solution will be of use to future implementations of the BTU, documentation will be kept with explicit explanations of workflows and code.

Workflows will be carefully documented to explain the proper functioning of the application as well as the purpose to the different design choices, at the developer team's discretion.

Every directory of the source code of the application will include a README file detailing the purpose/organization of the files/directories therein. Inter-code documentation shall be included for:

- Functions, including purpose and parameters
- Variables of the different classes, when declared
- Explanation of complex calculations/algorithms, at the development team's discretion

The full solution shall also include user documentation to aide in the proper use and understanding of the application and its processes as well as how to handle any unexpected behaviour in a safe manner. (As an example, when the simulator returns obviously erroneous output.)

Ensuring proper and full documentation as expected by the client, ongoing communications with the sponsor may need to be had.

Potential Risks

At the highest level, incorrectly stored test parameters is the worst case scenario of the risks associated with this project.. Plane engine testing would be compromised in the case that an error would go undetected by the people performing the testing. This scenario would end in an improperly tested Engine Control Unit (ECU) test. In this case, these tests may indicate false positives and false negatives for the test reports. This is the worst case scenario in all of this project's involvement. Improperly tested control units would be dangerous if they make it past the testing stages. This would result in the loss of client trust, the endangering of aircraft passengers, hindering of the testing protocol, and costing time and money. If the testing protocols are secure enough, there is a system to back up lost data, and the database's tests are updated regularly, this should only be an inconvenience.

Negative User Experience is another risk that would hinder the effectiveness of the current solution goal. An unintuitive design would leave the users of the BTU confused and unable to work as efficiently and effectively as they currently do on the system in place. If the users do not

find the design easy to use in place of the current system, Honeywell would lose time in testing the ECUs and the money associated with that time investment.

User error is also on the list of user related risks. As humans are error prone, the project must account for the possibility of incorrectly created testing parameters as well. This kind of error would hinder the efficiency of the testing program and lose money relevant to that time wasted. Preventing this would involve a dynamic web application design preventing incorrect test parameter creation and storing.

With the handling of information sent back and forth between the application and the engineers working with it, the test configurations or the test reports within the database are in the most vulnerable position towards harmful intent. In the case that someone plans on sabotaging the information within the database, they could erase or manipulate files within the database. Fortunately, they would not be able to interact with the ECU or the simulator on the back end besides the faulty reports and tests that would occur in the case of an injection attack on the database. This means that if the previous methods of upkeep from insuring against malfunction and applying those to database system upkeep, this should be only a slight inconvenience as long as any issues are tracked carefully. It is possible that using SQL injection techniques would also be in attempt to steal information. However, the client this project is for has expressed that security is not of the utmost concern as of right now..

During the transference of data, incorrect transfers are possible. This would result in odd test results in the most likely case. Data from the test configuration being scrambled would have odd effects on the ECU handling of simulator tests. Most likely, this would result in faulty testing that would be harder to detect. If allowed to continue, at a rate of which that could influence final test results in the long run for ECUs, Honeywell would lose time and money in this case. However, covering the potential of this risk with proper upkeep on results and reading the reports would keep this issue from being out of control as a whole. Identifying bugs as configurations are developed would also handle this. The likelihood of issues in this case would be left up to the robustness of the code handling data transfer protocols and networking protocols, since data transference occurs on the front and back end of the project. Both AngularJS and Django frameworks have available security functions to the frameworks themselves. The networking protocol, HTTPS, is already made to be secure. Having these security measures in place will keep the data being sent back and forth safe for the purposes of this project.

As for the implementation of the project itself, this web-based application of the Bench Test Utilities should be an upgrade for the engineers. This will allow the project to have a solid foundation for its use. As long as the project does not stray away from its design, there should be no reason that an engineer would not want to use this application as opposed to a version that would require moving around to do work that can be done at personal computers with this application. As for the future of the project, updates will most likely take place, but the idea of this project is meant to make the ECU testing process more fluent.

Project Plan

Execution of this solution will involve a number of smaller tasks that will build on one another. Fortunately, this solution lends itself well to successive development.

The creation of a network model demo is already in progress. This demo will be proof of proper communication between the application and the simulator.

Once this has been established, work on the application itself may begin. As every facet of the solution is tied together by this code, and given that this is the biggest and most-stressed requirement set-forth by the client, it makes sense that work would begin here. In order to have something tangible to be able to present to the client sooner, work will start on the front end first.

Once both ends of the application are communicating properly, focus may be directed towards database development. With better understanding of the options needed for the users of the BTU will come a better understanding of how to structure the database to better organize the information for easy and quick retrieval.

With a basic database established, hard-coded values that were used to test the client side and server side communication within the application may then be replaced with database queries to ensure that communication between the application and the database is functioning properly.

Next, communication between the application and the BTU simulator will need to be established. The backend framework for the application should be able to handle this without issue, however if problems arise, a simple http protocol should be simple to put together to allow for the exchange of files with the simulator. Sending a simple test script and confirming that

expected output is received would be a simple means of testing this functionality. This may be a test script that is already in use with the BTU to ensure expected results.

Once the basic structure of the solution is in place, the development of the automated scripting portion of the solution can be implemented. This will be added to the server-side code of the application and may be tested similar to how the communication link between the application and the simulator was.

Though many of these steps may be conducted simultaneously, there will be a number of deadlines in place to ensure a smooth flow from one requirement to the next.

Conclusion

Plane safety starts before the plane ever leaves the ground. To ensure this safety, a plane's components require extensive testing by manufacturers before ever entering use. The goal is to create a more efficient and modern testing system for Honeywell, addressing the inefficiency that Honeywell has outlined with the current system of use of the Bench Test Utilities in place To accomplish this, we are developing a web-based Bench Test Utilities that utilizes a central database. This would allow the current functionalities that exist within the Bench Test Utilities to be effectively recreated on the web based application. This requirements specification document is meant to outline those requirements for the client and lay down a foundation of progress to complete for the project ahead. As tech demos are put together proving that the research up to this point is possible. We are confident that we can provide a web-based application that honeywell will find much more convenient than the system they are currently using.