Project Initiation Document

Software Integrity Tester

ASRC Federal

Software Engineering I

Team Penguins

By: Cole Christensen

Table Of Contents

Project Scope:

In Scope

In the scope for our project, our scrum team must work together to complete an integrity tester. This security tool attempts to reveal possible weaknesses for Ada programs. Inputting Ada code will allow our program to be parsed for the weaknesses discussed using a list of twenty-five common vulnerabilities, researched on our own. Familiarizing our group with Ada more than we already know is in our scope first as weaknesses will be easier to exploit.

Out of Scope

Out of scope options would include fixing the errors we identify instead of generating a report on how properly the user program runs, and attempting to create a program with add-ons to possibly expand for multiple languages.

This project is deliverable in phases as the goal does not change and we do not anticipate it to. Small features will be added when the main functionality exists as undecided stretch goals that we need to figure out if we can accomplish in sprints.

Project Background:

Sponsors

This project was created by our sponsors, Michelle Darby, Craig Sobieralski, and Thomas McLoughlin. They are a part of ASRC Federal.

Ideology

This project was created because there is not a program already that solves problems with Ada and our sponsors discussed how important Ada is in terms of government usage. The idea of this software project came as a result of an existing C/C++ error parsing software which prioritizes security reports by risk level of flaws. These flaw risk levels show up on the reports with the amount of flaws, or “hits” (we will need to know hit density for our program). Using coded algorithms for well-known problems within the C/C++ database and taking the source code needing a check (ignoring comments or other documentation), the algorithms identify certain patterns of code without actually processing the semantics of the code. This is a syntax only tool.

Project Feasibility:

The feasibility is hard to satisfy due to a couple of obstacles. Studying Flawfinder, syntax correction is a main pattern captured by hit density. This project cannot be easily done as we do not know semantics of other source code. We may try to highlight errors in code and be wrong about what we tried to identify. This is where we are deciding a skip functionality to move on to the next caught error and one of our sponsors made it clear that the rest of the sponsors were on board with suppressed warnings. If this were not the case, we would be forcing the user to correct their work when it may not be wrong.

Project Requirements:

Expectations

From the requirements, we understand limitations and expectations. We are expected to parse and identify code so we don’t have to go as far as fixing code. We must depend on the progression of code to be complete which was requested by our contacts. In other terms, there is assurance that if the inputted is not finished, the functions that are already designed are at least complete. We were asked to filter the weaknesses in the source code parsed as well which will use a drop down box. The hit density of the program may include coloring but will start with a basic line number detail. A command line was requested and can be very helpful to the users of our software. This was not confirmed with all contacts yet and will be a later goal if necessary. On a company level, the software will allow the company to plug in new weaknesses researched in the future so our program can be maintained to continuously work.

Constraints

C++ is what we would like to use to program this assignment and this may be an obstacle due to the lack of garbage collection. Java provides this but our group decided against it. Not knowing anything about the code is a constraint too. We have to perform successfully when this software is used every occasion even without knowledge of semantics. The text box is not a compiler which gives constraint to having to use other party software overall. We were told not to have a software dependent on third party software through code. The user can simply just go to a different environment on their own.

Project Initiation:

Framework

We have an initial plan coming together. We decided that we could construct a GUI first and then have the next step to have an editable text box for the input code. This is only a frame without functionality and it will satisfy viewing pleasure. Parsing will be our next goal as we want to parse all code and store it somewhere for future purposes (loading and saving). Building a compiler is out of the question as the user will have to make sure his/her code compiles. The graphical design work in progress will at this point, be close to done.

Functionality

We can get this far without having to write code for exploiting weaknesses, which will be the hardest part theorizing and implementing ideas. We can relate to certain algorithms that Flawfinder uses for similarities between Ada and C++.

Our scrum master will ensure that the development team divides these tasks in a timely matter so we can move on to the goal of this software, error identification.

Team Organization:

Sprint 0 and the Future

For team organization, we obviously learned how to use Trello so PBI’s are being added to our backlog. We have not hit Sprint 1 and no one has moved a PBI to the sprint backlog yet like directed. Once we reach this sprint, the development team will be organized as they can divide tasks evenly. We communicate through Slack daily and our meetings for daily scrums take place during Monday and Wednesday class as well as Tuesday’s at 1.

Organization

Our scrum master, Joe is writing a document keeping track of all the daily scrums or other scrum activities, just as he should to keep the team in order. Our manager is keeping in touch with our contacts to make sure our meetings will be organized and to relay any other questions or concerns. Cole Christensen is taking this first deliverable and we have not yet decided the plan for the rest of the deliverables. We need to organize the plan for the rest of the deliverables as they have not been assigned to anyone. The sprints from here on out will in order, build a framework GUI, parse and save source code, write methods to analyze code for errors, organize risk identification while adding result filtering, and finally, the vulnerability add-on for new identification techniques or risks for the business only.

Project Risk and ROI:

Risk and Mitigation

Returning to the semantics evaluation of source code provided by a user, risk takes place. Our warnings and skip functionality will be accounted as mitigation strategies. The drop down box limiting which vulnerability weaknesses to search for is also a risk due to the uncertainty of labeling the wrong problem. Weaknesses need to be immediately prioritized to reduce the likelihood of failure to label problems. For example, looking at spacing can be a label itself, as well as punctuation. We must account for users not inputting code that is complete which can be a caught for an error message. Any source of software error should save work in case of unexpected crashing if we cannot return to our home screen for the Integrity Tester. We expect users to be slightly confused when first using our software. We want easy to view commands for software usage even though a first time user has no prior experience. This is where our command line will avoid human complication

Cost and ROI

We will return to the estimated cost of the whole production as we are unsure as to how to measure this section yet with certainty. Main usage of Ada is for government purposes. We will not have any extra costs once the software is up and running due to the user only needing to input their data. The feature to add more functionality from the business could cost more money. The time it would take to self-check code would be cut due to the computer vs. human run-time of parsing and identifying which saves cost in terms of time. The return of investment will rise because more money can be made with the amount of time that has been saved. This software does not directly communicate with hardware so there is no maintenance involved there. The life time of our project is long as Ada is old and still commonly used in certain business settings which will reduce extra costs in the future.