ASRC Federal

Metrics Database
Project Initiation Document

Prepared by: Semih Sahin
Team Name: Tortoise
Sponsor Company: ASRC Federal
Sponsor Representative: Christopher R. Bartley
Version: 1.0
Prepared On: 11/10/19

Related Projects

Related Projects	Status	Challenges	Team
Software quality and productivity visualization tool	Stretch Goal	Share the database access	Yaks

Defined Terms

Term	Description
The Team	Refers to the Product Manager, Scrum Master, and Development Team (see Section 7, Organization and Governance)
The Sponsors	Refers to the individuals who is from ASRC Federal
GCOV	Code coverage tool that collects metrics on % of code with test coverage
CPPCHECK	Tool that scans for error generating code
UCC	Tool that generates complexity and sloc count from code.

Stakeholders

Stakeholders	Description
The development team	Consists of people who submit code to the repository.
The management team	Consist of people who oversee the dev team and evaluate their performance.

Table of Contents

1 Purpose

The idea of this project was sponsored by ASRC Federal in order to be implemented by Rowan Software Engineering class. The metrics database program will be able to help the developers and management take their version control system one step further by simply adding a Git hook to their machines that will execute whenever a scheduled event is triggered by the user.

This trigger could happen in two ways such as via push/commit or both. The program will be a light-weight script that will be compatible with any version control system that supports hooks. A hook is script file that will execute during a triggered event. This tool will use many other great tools that are already available and pick the most important metrics generated by them and then store those metrics for feedback to the developer and the management.

A developer can get instant feedback after a push/commit in order to see if there was any run-time or compilation-time errors generated by their submitted code. Developers catching their errors before a huge software compilation could make sure that hours spent on compilation was not a waste.

Another great benefit that the generated metrics can provide is the ability for management to monitor performance of its development team. Certain metrics collected can be used for calculating estimation of future projects by using previous performance of the development team. This will give you the ability to give accurate deadlines for the customer and the development team.

2 Project Overview

This project will take advantage of a feature that most version control systems such as Git have which is a hook folder. A hook folder is a folder that is within a system files of a version control system. This folder will enable us to execute any script on trigger placed within the folder. We will then take advantage of four tools that can be used within a Linux environment.

The first tool that will be used is Jenkins. Jenkins is a free and open-source automation server that can be used for many things such as automated tasks and compiling projects in many instances at once. The feature that we will use in Jenkins is the continuous integration feature. Continuous integration is when instead of compiling all the code at a certain scheduled time which sometimes takes hours by having many instances that pull the code from the repository whenever it detects new submission. This will enable us to execute the hook script whenever new code is submitted to the repository in many instances in order to provide the developer with instant feedback rather than wait for more submissions.

The second tool that we will use is UCC (Unified Code Count). UCC is a counting tool that will scan source files for lines of logical and physical SLOC. A SLOC is a line of code other than a comment that serves a function or a purpose within the code. SLOC's will be needed as a metric in order to calculate a developers SLOC/H given their labor hours on the project.

The third tool that we will use is CppCheck. CppCheck is a static code analyzer for C/C++ code. This tool will enable us to gather information on if the code submitted by the developer contain any memory leaks, mismatching allocation-deallocation, buffer overrun, and many more system critical errors that can cause headaches in the long run or compromise system security to certain attacks.

The last tool we will use is GCOV. GCOV is a compilation code analyzer that will generate metrics on how many times an exact line of code is executed throughout the program and provide percentage of code coverage.

All the metrics generated will be combined and cleaned into a single file that will be submitted as a batch to a SQL database. We will use MySQL as the implementation of the database. The database will keep track of all the metrics by date, project id, and author.

3 Scope

3.1 In Scope

The following is a list of components that will be completed within this version:

- Single file hook script design for portability to any version control systems.
- Ability enter labor hours by the developer.
- Being able to trigger the hook by commit, push, or Jenkins.
- Creation of a log file for each batch of metrics.
- Sorting metrics by date.
- Partial insertion of metrics if one tool fails.
- Generating metric from the success/fail of a Jenkins compilation.
- Custom error messages caught by the tools will be sent to the database.
- Handling more than one instance of Jenkins depending on the size of projects.
- Handling of simultaneous insertion to the database.
- The database will be implemented using MySQL.

- Batch of metrics being submitted to the database will be in the format of a CSV.
- Calculation of SLOC/H using labor hours.
- Securing the key file to the database using MySQL.
- Exporting simple metrics through MySQL.
- Storing tag info from commit as a metric.
- Creation of an ER diagram for the database.
- Exporting metrics through certain date range.

3.2 Out of Scope

The following is a list of components that will not be completed in this version:

- Running the metric collection tools on files that have changed only.
- Making the hook run server-side.
- Connecting labor hours through another database rather than manual entry.
- Sharing a key on the database with team Yaks.
- Having more security on the database key.

3.3 Future

Possible future phases of the project can include:

- Creation of a single tool that will do the job of 4 tools currently used.
- Modification of the tools to create only metrics that are wanted in order to be more efficient.
- Creation of an external application that can be used by Management and Developers for feedback and estimation.
- Having more than one way of exporting metrics.

4 Feasibility Analysis

The metrics database tool is a feasible project. All the tools that will be used on making this project happen already exist. Two people in the scrum team already are experienced in implementing and designing a database. We also have another member is has experience in working with bash scripts on a Linux environment.

As for other parts, the Scrum team will be able to learn enough about the metrics tools being used in the project in the given time. Considering that the team already has experience in most of the tools being used, spiking should only be needed on tools such as Ucc, GCOV, Cppcheck, and Jenkins. The rest of the team should be handle the workload when certain group members are spiking.

5 Assumptions, Dependencies, and Constraints

5.1 Assumptions

The assumptions made while working on this project are:

- Certain needed ports open on the Linux server.
- The developers will have root access on the Linux server.
- The Linux server will be up most of the time other than daily restarts.
- Linux server will support MySQL and more than one instance of Jenkins without hurting performance dramatically.

5.2 Dependencies

The project is depending on certain items below:

- Linux Server that can handle the load of more than one Jenkins continuous integration instance running.
- The user giving the right flags when pushing or committing to the repository.
- The use of Jenkins automatic integration system.
- Code coverage generated by GCOV.
- SLOCs generated by UCC.
- Error messages generated by CppCheck.
- The use of a MySQL as the database to store metrics.

5.3 Constraints

System Environment
The environment that the metrics tools can run on has been constrained to the use of Ubuntu Linux server. This constrain has been set in place because almost all tools that are used in this project can only be installed on a Linux machine.

System Memory
The RAM of the memory has been constrained to at least 32 GB in order to handle Jenkins running more than one instance of compilation. The RAM requirement can increase depending on how many instances of Jenkins that the user needs depending on number of projects and the number of developers working.

MySQL
The server hosting MySQL must have port 3306 inbound open in order to communicate with the script. If the exporting feature is used, port 3306 outbound must also be open in order to export.

Jenkins
The server hosting the Jenkins instances must have the ports 8080 open both inbound and outbound. This is needed for the Jenkins to pull from the version control system and to insert any compilation comments to the database.

6 Initial Project Plan

Below is the initial plan of the project split into 5 sprints:

Sprint 0

- Fill the product backlog with project high-level requirements.
- Breakdown the high-level requirements into smaller concepts.
- Create user stories for each item in the product backlog.
- Give business value to each item in the product backlog.
- Give difficulty value to each item in the product backlog.
- Meet with Myers to setup an AWS for the project.

Sprint 1

- Download and setup MySQL on the AWS.
- Research updating a database (DML).
- Write a Git hook that will execute when user pushes code to Git repository.
- Add a feature to the hook to distinguish between push or commit.
- Add a feature to the hook so it can gather commit info.
- Develop a list of metrics generated by GCOV.
- Develop a list of metrics generated by UCC.
- Develop a list of metrics generated by CppCheck.
- Generate an ER diagram for the database.
- Create tables in MySQL for each tool.
- Test the if the Git hook can differentiate between push and commit.

Sprint 2

- Add the execution of CppCheck to the Git hook.
- Add the execution of UCC to the Git hook.

- Cleanup the output from CppCheck into something readable.
- Cleanup the output from UCC into something readable.
- Test the execution of CppCheck and UCC.
- Insert the output from CppCheck and UCC to the database.
- Test that the metrics are inserted correctly into the database.

Sprint 3

- Add the execution of GCOV to the Git hook.
- Cleanup the output from GCOV into something readable.
- Insert GCOV metrics to the database.
- Test that the metrics are inserted correctly into the database

Sprint 4

- Combine the output of all the tools metrics into a one single file so it can be inserted into the database as a batch.
- Setup Jenkins on the AWS.
- Get Git Hook to work on Jenkins so the hook can work server side.
- Insert the output of the Jenkins into the database.
- Test to see that the Git hook executes properly on Jenkins.

Sprint 5

- Modify the tags on the hook to accept labor hours as input.
- Add a way to calculate SLOC/H using labor hours.
- Add functionality to be able to export a simple excel datasheet from the database using date range, commit info, and author info.

7 Organization and Governance

- The project's contributors are Parth Darji, Marcus Penate, Jeffrey Paulson, Semih Sahin, Nicholas Santoro, and AJ Schenker.
- The project's sponsors are Christopher R. Bartley.
- The Team consists of three main groups: The Product Owner, the Scrum Master, and the Development Team
    - The Product Owner, Semih Sahin, is the primary communicator between the Team and the Sponsors. Semih Sahin is also in charge of overseeing and delegating the project's tasks to the Development Team.

- ○ The Scrum Master, Jeffrey Paulson, is tasked with assisting the Development Team. Jeffrey Paulson will also ensure the Agile Scrum methodology is being followed.
- ○ The Development Team, consisting of Parth Darji, AJ Schenker, Nicholas Santoro, and Marcus Penate, are the primary architects, designers, and coders for the project. It is their responsibility to follow the guidance set forth by the Sponsors and Product Owner and implement the project.

8 Communication Plan

- The team meets for 15-minute daily scrums 3 times a week at the beginning of the Software Engineering class. They also meet later after that class for hour and a half for a secondary scrum.
- Discord is used for conference-call meetings and discussions as needed.
- Sprint reviews will be held every two weeks on a Monday at 3:30 pm for 30-minutes starting on 9/30/19 in order to keep the stakeholders updated.

9 Quality Plan

9.1 Functional Requirements

This is a list of the functional high-level requirements which define the project:

- The git hook will execute when the user specifies in the task.
- UCC will run when specified by the tag of the commit.
- GCOV will run when specified by the tag of the commit.
- CppCheck will run when specified by the tag of the commit.
- The script will automatically sort through the metrics.
- User can state labor hours when giving the tag.
- SLOC/H will be calculated automatically once the hours are given.
- MySQL will be able to export excel datasheet given certain date range.

9.1 Non-Functional Requirements

This is a list of the non-functional high-level requirements which define the project:

- The database can handle more than one submission at a time using temp table.
- The hook script will be portable in a way where it will work on any version control system that supports hooks.
- The metrics will still insert to the database as long as one of the tools do not fail.
- The database key file will be encrypted by MySQL.
- The Jenkins server must be able to handle more than one instance.

- The data tables for the metrics will be able to handle growth.
- The database will need to be up in order to export/insert metrics.

9.2 Deliverable List

The following are documents that will be prepared for the sponsor throughout the project:

- Project Initiation Document
- Requirements Document
- Design Document
- Validation Plan, with Traceability Matrix
- Test Plan, with Test Scripts
- User Manual, with Poster and Abstract

10 Risk Assessment

The following are risks that this project may encounter:

Database Availability
The metrics tool will not work in the case of database being down. The tool will not be able to insert any metrics gathered by the tool. The export functionality will also not work until the database comes back up. While there is no way to mitigate the export functionality there is a work around the insertion. The script can be modified to check if the database is online and, in the case, that it is offline, it can save a copy of the metrics locally. The next time that the tool is run on that machine it can check for any local batches and submit them along with the current one.

This is a low likelihood and a moderate severity risk. The likelihood comes from the fact server being down can easily be mitigated by having servers in more than one location with backups. The risk is moderate considering it completely disables the export function, but you can still import on the next submission. This would affect the availability of the product.

Data Overwrite
In the case that two users submit metrics to the database at the same exact moment in milliseconds, one user might overwrite the other one’s metrics. This can result in some data points missing or overwriting the current one. This can be mitigated by having a temporary table in the database that will hold the data before actually writing to the database.

This is a low likelihood that this will happen. Even if someone tried to do it exactly at the same that they might still miss each other by some milliseconds. Although the probability of this happening is near impossible, if it does happen the effects will be very serious such as overwriting or missing data. This would affect the reliability of the product.

11 ROI

There will be no financial cost or return on investment on this product. However, this product will still benefit ASRC in many ways if they plan to continue develop or use this product. These benefits include:

- Ability to detect software errors before a long compilation is started thus saving time.
- Being able to see the performance of an individual person or a whole group of development team.
- Using the metrics gathered to estimate future projects accurately.
- Being able to get your code as a developer get tested before compiling it with others code changes.
- Ability for developer to see the location and the type of error that occurred from their code.