Bug Hotspotter

Project Specification

Spencer Smith Nathan Reinhardt Dylan Williams

September 22, 2015

Revision 1.0.0

Table of Contents

1. Introduction..... 1
1.1 Business and Domain Description..... 1
1.2 Purpose..... 1
1.3 Concept of Proposed System / Scope..... 1
1.4 Product Overview..... 1
1.4.1 Product Perspective..... 1
1.4.2 Product Features..... 2
1.4.3 User Characteristics..... 2
1.4.4 Limitations..... 2
1.5 Definitions..... 2
1.6 Stakeholders..... 2

2 References and Other Standards..... 3

3 Specific Requirements..... 3
3.1 External Interfaces..... 3
3.2 Functions..... 3
3.3 Usability Requirements..... 3
3.4 Performance Requirements..... 3
3.5 Logical Database Requirements..... 3
3.6 Design Constraints..... 3
3.7 Software System Attributes..... 3
3.8 Supporting Information..... 3
3.9 Course-specific..... 3

4 Verification..... 4

5 Appendices..... 4
5.1 Assumptions and Dependencies..... 4
5.2 Acronyms and Abbreviations..... 4

1. INTRODUCTION

1.1 Business and Domain Description

The client seeks a system that, when provided a *git* code repository, identifies files within the repository that have a high likelihood of containing bugs. Existing systems provide similar results using various static metrics (i.e. lines of code, cyclomatic complexity, etc.). The system to be developed will instead employ dynamic metrics that take into consideration the number of repository commits for a given file and the relative frequency of the commits.

1.2 Purpose

The system will be utilized by code developers and project managers to reasonably identify the files within their code-base that are in need of closer scrutiny by experienced developers. Using the system can also provide a better understanding of a particular repository to new developers. By identifying and flagging frequently modified code, new developers can see more immediately where their efforts may be most needed as files that have remained static for a considerable amount of time are likely to be either bug-free or insignificant.

1.3 Concept of Proposed System / Scope

The “Hotspotter” system will predict the locations of bugs in a given code repository by identifying files that are being frequently modified in recent time. These “hotspots” will be given a score of the likelihood that they contain bugs.

1.4 Product Overview

1.4.1 Product Perspective

System Interfaces

The web server will interface with the web page and database. The web server will be handle calculating data, sending data to web page and sending data to the database for storage. The web page will display the data from the web server and send commands to the server. The database will handle storing and retrieving data for the web server.

User Interfaces

The user interface will be a simple web page to interact with the remote database through the web server.

Hardware Interfaces

Not applicable

Software Interfaces

The software stack used by the system will be MEAN. The application will be web based so it will be able to interface with any operating system will a supported browser. For simplicity the supported browser will be Chrome.

Communication Interfaces

The web page will communicate through standard Secure HTTP since some of the information could be sensitive.

Memory

The database will store every revision from the git repository so the amount of active repositories in the system will be limited.

Operations

Normal user operations include:

- Connect to repository
- Calculate hotspots
- Filter hotspots
- Update repository

Special user operations include:

- Specify update timeframe
- Remove active repository from database

Site adaptation requirements

Not applicable

1.4.2 Product Functions

The system will first connect a git repository specified by the user and download the said repository to a remote database. The system finds the hotspots in the repository with a user specified date range and dynamic metrics. The system displays the results in a visual and/or text format for the user to view. The results can be filtered by a file, function, developer or intensity of hotspot. The repository data is automatically pushed to the server by a given time frame or can be manually updated by the user. Multiple user can look at the results through different web interface instances.

1.4.3 User Characteristics

This software will be mainly used by software project team leaders and resource managers. The individual software coders may also find it useful for self-evaluation. The system won't require the user to have programming experience but will act as tool for them to understand where the troublesome code is located.

1.4.4 Limitations

The system will only be able to use git repositories for calculating hotspots. The data will be store on a remote database for processing and presented through a web interface.

1.5 Definitions

Github – Distributed Repository
Bug – Unplanned problematic piece of code.
Repository – A place in which a collection of code lives.

1.6 Stakeholders

Dr. Igor Crk – Product Owner
Nathan Reinhardt - Developer
Dylan Williams - Developer
Spencer Smith - Developer
Dr. Gary Mayer – Senior Project Professor

2. REFERENCES AND OTHER STANDARDS

"Semi-Agile Software Engineering (SAGE)

3. SPECIFIC REQUIRMENTS

3.1 External Interfaces

GitHub – This will serve as our primary data source. The project will pull in repositories held within GitHub and run tests on it.

3.2 Functions

Primary function will be to pull down a given GitHub repository. Run a sequence of tests and give a final score that will be used to find potential bugs within the project.

3.3 Usability Requirements

The project will meet its usability requirements if it can be given a GitHub repository and find potential bug hotspots in the code with a measurable metric.

3.4 Performance Requirements

The system will provide its results within a reasonable amount of time that may scale with the size of the repository provided. An indefinite amount of users will be able to view the saved data simultaneously.

3.5 Logical Database Requirements

Not applicable.

3.6 Design Constraints

No design constraints have been issued by the client.

3.7 Software System Attributes

No system attributes issued by client.

3.8 Supporting Information

References:

- http://google-engtools.blogspot.com/2011/12/bug-prediction-at-google.html
- https://github.com/igrigorik/bugspots
- http://landley.net/writing/git-bisect-howto.html
- http://macbeth.cs.ucdavis.edu/fse2011.pdf

Problems Solved:

- This software searches through git repositories and finds hotspots at the function level using dynamic metrics. The hotspots in the files are visualized with different shades of colors showing the intensity of the hotspot.

3.9 Course-specific

Semi-Agile Software Engineering (SAGE)

4. VERIFICATION

Not applicable

5. APPENDICES

5.1 Assumptions and Dependencies

- The GitHub repository uses git best practices
- The Github repository uses git bug tracking practices

5.2 Acronyms and Abbreviations

- Repo : Repository
- MEAN: Mongo, Express, AngularJS, NodeJS
- TDD: Test Driven Development
- Metric: a trackable code variable used to calculate metrics
- Metrics: a method of measuring something, or the results obtained from this.
- Hotspot: portion of code in a repository that is deemed a potential troublesome area.

Team Members

_____	_____
Signature	Date
_____	_____
Signature	Date
_____	_____
Signature	Date

Client

_____	_____
Signature	Date

Oversight

_____	_____
Signature	Date