Project Report 2

Frameworks to Support Continuous Delivery

with Kostas Kontogiannis (?)

Gurpreet Singh & Paul Bartlett

 # 250674134 & 250753008

Software Developers

Contents

CS4470Y **2**

Project description 2

Roles 2

How roles have changed 3

Revised System Requirements 3

Revised Project Plan 5

Project Deviation 5

CS4470Y

Project description

The focus will be to design and develop tools to deal more efficiently with challenges pertaining to shortening bug fixing and maintenance cycles and, achieving continuous delivery. There are three sections to this project:

Data Collection

In this section we need to design and implement infrastructure for collecting and modeling system data from diverse information sources such as application logs, user newsgroups, and technical reports related to application failures and application performance,

Data Processing

Next goal is to design and implement tools for processing, analyzing, and amalgamating the acquired information into analytics reports, so that these can be efficiently disseminated to the appropriate stakeholders such as project managers, system architects and, developers and,

Data Integration

Finally we need to integrate our findings into existing development environments by designing and implementing enhancements to existing IDEs (e.g. Eclipse) and collaborative development frameworks, in order to assist the appropriate stakeholders to quickly act upon the acquired information in their native work environment, and better plan and shorten the time required for micro-services application development, maintenance, and deployment.

Roles

Gurpreet Singh

- Lead Architect
- Documenter

Paul Bartlett

- Project Manager
- Lead Requirements Analyst
- Lead Tester & Quality Controller

How roles have changed

Over the last milestone roles have not changed other than both members trying to get in contact with the supervisor and failing.

Revised System Requirements

Section A : Data collection and modeling

- **Feature 1:** Git Based Data Collection
- FR 1: Extract commits from a .git directory
- FR 2: Analyze and record classes/methods effected by each commit
- FR 3: Process and record the amount of code changed with each commit
- **Feature 2:** Test Based Data Collection
- FR 1: Record results of the test suite each time it is run
- FR 2: During test phase inject previous risk analysis
- **Feature 3:** Issue Based Data Collection
- FR 1: Record issues filed for the code base
- FR 2: Connect issues to their resolution and make an inference from the association
- **Feature 4:** Application log based Data Collection
- QR 1: Allow parsing from diverse options (User newsgroups, technical reports, etc)
- **Feature 5:** Data combination

- FR 1: Standardize and combine Git data into a central data model
- FR 2: Standardize and combine test data into a central data model
- FR 3: Standardize and combine issue data into a central data model

Section B : Creating report processing software

- **Feature 6:** Conduct Risk Analytics
- FR 1: In the data model figure out risky parts of the code and mark them
- FR 2: Identify issue keywords that are higher risk than others
- FR 3: Identify impact when a test is changed (how much coverage it offers)
- **Feature 7:** User Views
- FR 1: Create a PDF report for stakeholders to see progress in codebase
- FR 2: Create a Web view for exploring the risk of the codebase
- FR 3: Create Github, IFTTT and Slack integrations for viewing framework data model

Section C : Creating enhancements to dev environments

- **Feature 8:** Continuous Integration
- FR 1: Create test integration so developer can see risk rating when conducting tests
- FR 2: Develop command line interface to the framework
- FR 3: Research deployment frameworks and integrate risk analysis into pipe line
- **Feature 9:** IDE integration
- FR 1: Develop VIM plugin for viewing risk rating of each method/class
- FR 2: Develop IDEA plugin for viewing risk rating of each method/class
- **Feature 10:** Microservice Plug-And-Play library

- FR 1: Make the framework easily deployable into a microservice
- FR 2: Reduce dependencies and deliver a minimally intrusive product

Revised Project Plan

We have no idea how well we are doing in respect to what the supervisor wants therefore all efforts are on trying to contact the supervisor and then development can continue afterwards. This is because we don't want to waste time developing out of specification. Our supervisor has not commented on our initial project plan or our initial system requirements therefore lots of development is not possible.

Project Deviation

- Still waiting for professor first point of contact
- Still waiting to hear what the professor wants for this project