**Project Proposal**

**Symbolic Execution**

**Course: SE305 Software Project Lab – I**

Submitted by

|  |  |  |
| --- | --- | --- |
| *Toukir Ahammed* | *Roll: 0806* | *2015-2016* |
| *Aba Kawsar* | *Roll: 0825* | *2015-2016* |

Document Version 1.0

Submitted to

**SPL I Coordinators**

**Dr. Mohammad Shafiul Alam Khan, Assistant Professor, IITDU**

**Amit Seal Ami, Lecturer, IITDU**

**Md. Saeed Siddik, Lecturer, IITDU**



**Institute of Information Technology**

**University of Dhaka**

[15-02-2017]

Date: 15-02-2017

SPL I Coordinators

Institute of Information Technology(IIT)

University of Dhaka, Dhaka, 1000

Subject: Application for approving Software Project Lab(SPL-2) proposal.

Dear Sir,

With due respect, we have the honor to state that I am a student of 5th semester of BSSE 08th batch. As our software project of SPL-2, we have selected a project titled “Result Management System” supervised by Md. Rayhanur Rahman, Lecturer, IIT, University of Dhaka.

We hope that, you would be kind enough to approve our project proposal and oblige thereby.

Yours Sincerely, Supervised by,

Toukir Ahammed Md. Rayhanur Rahman

Roll: BSSE0806 Lecturer, IIT,

University of Dhaka.

Aba Kawsar

Roll: BSSE08025

**Title: Result Management System**

**01. Introduction**

Symbolic execution is a useful technique of analyzing a source code to identify which inputs cause each part of a program to execute. Symbolic execution means executing a program with symbols as input rather than concrete values. During a normal execution the program takes a concrete input value and the program proceeds according to that particular value. But during a symbolic execution the program takes a symbolic value (e.g., α) rather than a concrete value and then proceeds with that symbol as an input value. As the input value is concrete in case of normal execution, the control flow path is fixed for that particular value. But in symbolic execution the symbol (e.g., α) can take any value, so the program can take any feasible path and thus can explore multiple paths simultaneously.

**02. Background**

In the field of software industry testing is a very important thing. Symbolic execution is a very popular technique in software testing. Solving path constraints (PC) is one of the important parts of symbolic execution. Before starting this project some backgrounds are must needed about symbolic execution, program paths, path constraint (PC) and obviously basic programming knowledge.

**03. Description**

This project is about symbolic execution of a source code. The input of this program is a source code file written in C language. Then the task is to detect basic blocks and branching statements where the program can be divided into branches. According to the conditions of all branches the program will calculate the path constraints (PC) for each individual path.

**04. Motivation**

Software testing is very important in software industry because every year billions of dollars are lost due to software system failures. More than one third of this cost could be avoided if better software testing is performed [1]. So it is necessary to test the software properly. Symbolic execution is very useful and effective technique for software testing. So I am interested in this project and I am hopeful that it will help me achieving proper knowledge and skill regarding software testing as well as improving my programming skill. This will also help me in my future projects.

**05. Objective and Scope**

The objective of this project is the symbolic execution of a program to find the path constraints for each individual path in program flow graph.

The scope of this project is the symbolic execution of a simple C source code.

**06. Dependencies**

To run this program, an executable and a well indented source code and C++14 compiler is required.

**07. Risk and Limitations**

The limitations of this project are it will not work when the numbers of infeasible path are large, there are infinite loops in the source code. It can not handle a very complex source code as the number of feasible paths in a program grows exponentially with an increase in program size and can even be infinite in the case of programs with unbounded loop iterations [2].

**08. Assumptions**

It is assumed that all source codes are written in C programming language and are properly indented. It is also assumed that all source codes are written following the “**Recommended C Style and Coding Standards”** [3].

**09. Timeline**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task / Week** | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 |
| Project Analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Proposal Submission |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Documentation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Report Submission |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**10. Conclusion**

Throughout this project I can learn new things, develop my skill in programming and achieve knowledge about symbolic execution. After all, I can gather a lot of experiences to handle a project which will help me in the future projects.

**References**

01.**”Software Errors Cost U.S. Economy $59.5 Billion Annually, NIST report 2002.”** <http://www.abeacha.com/NIST_press_release_bugs_cost.htm> (Last accessed on: 09/02/2017).

02. Anand, Saswat; Patrice Godefroid; Nikolai Tillmann (2008). ["Demand-Driven Compositional Symbolic Execution"](http://link.springer.com/chapter/10.1007%2F978-3-540-78800-3_28?LI=true). Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science. (Last accessed on: 09/02/2017).

03.“**Recommended C Style and Coding Standards”** <https://www.doc.ic.ac.uk/lab/cplus/cstyle.html> (Last accessed on: 09/02/2017).