

A MONITORING FRAMEWORK
FOR SOFTWARE ARCHITECTURE DEGRADATION
IN DEVOPS PRACTICE

Gonagala Withanage Piyumi Sampath Gunarathna

158216D

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

March 2019

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

The above candidate has carried out research for the Masters thesis under my supervision.

Name of the supervisor: Dr. Indika Perera

Signature of the supervisor:

Date:

Acknowledgement

I am deeply grateful to my supervisor Dr. Indika Perera from the University of Moratuwa Department of Computer Science and Engineering whose help, motivation, suggestions and encouragement helped me in all the time to complete this work. Dr. Indika Perera inspired me greatly to work in this project.

I would like to thank my family and friends for their understanding during the period of the project and gave me support. Without their help I would face many difficulties in the time of the project. And finally my grateful thanks to all the people who help me during the project.

Abstract

This is a research report that desired to carry out to find the software architecture issues and possible monitoring techniques to overcome those issues in DevOps practice. DevOps is a new philosophy that helps software organizations to innovate faster and to be more responsive to business needs, it promotes collaboration between developers and operations which improves quality of software development and more frequent software releases.

Continues delivery in shorter development iterations and deploy a software system faster are the key practice of DevOps. And also because of that faster practice, there is huge risk of occurring a software failure unless the continuous monitoring. Therefore the DevOps teams use automated tools for monitoring functional criteria and the performance. But still, most of the teams are not monitoring the architecture of the application.

Unless there need to be done a change intently, the architecture of the system need to pertain its initial designed and finalized architecture in DevOps culture, in shorter development iterations. Therefore it is more important to monitor the software architecture without leading to a software drift or erosion. Therefore this research objective is to build a Monitoring framework for architectural degradation in DevOps practice.

The end goal is Continuous Testing and Continuous Monitoring. Testing and Monitoring are what will prove that the new built is the right required application, that functions and performs as designed and desired.

As the main research objective it identified a missing area of software architecture monitoring methodologies and analyzed and identified a way to prevent software architecture erosion using that. This research is more focused on unconventional usability of the solution and project file contents and how it can be leveraged to capture the architecture of the application and how it can be used as an effective architecture design monitoring framework.

This research states a methodology which uses project file and solution file content to detect the architecture specific information from the code base and a mechanism to capture them and compare them with a pre-defined architecture rule set. An empirical and theoretical evaluation has been done to prove this concept actually works in real life scenarios. It opened up a new area of architecture conformance checking to the future researchers of the field of software architecture.

Keywords

DevOps, Continuous integration, Continuous deployment, Defect warnings, Continuous monitoring, Software development lifecycle, Quality assurance, SDM

TABLE OF CONTENT

Declaration	i
Acknowledgement	ii
Abstract	iii
TABLE OF CONTENT	iv
LIST OF FIGURES	vi
LIST OF TABLES	viii
1. INTRODUCTION	1
1.1 Background	2
1.2 Importance and novelty of the problem Background	3
1.3 Research Problem	5
1.4 Objectives	5
1.5 Overview of the Document	6
2. LITERATURE REVIEW	8
2.1 DevOps Practice	10
2.2 Continuous Integration in DevOps	11
2.3 Software Architecture in Practice, Architecture Issues, Software Erosion and Software Drift	14
2.4 Source Code to Architecture Mapping Tools	18
2.4.1 Doxygen	19
2.4.2 Architexa	20
2.4.3 Code-maps	21
2.5 Defect Warnings Techniques.	21
2.6 Layered diagrams	23
3. METHODOLOGY	26
3.1 Identify the Possible Solution	27

3.2 Developing a Proof of Concept	28
3.3 Evaluation of the Proof of Concept	29
4. IMPLEMENTATION	31
4.1 Understanding the Solution File	32
4.1.1 Solution File Contents	33
4.1.2 Methodology of Observing the Solution Content	35
4.2 Understanding the Project File	37
4.2.1 Project File Contents	38
4.2.2 Methodology of Observing the Project Content	47
4.3 Generate Architecture Design	41
4.3.1 Component Draw and Placement in Document	42
4.3.2 Map Relationships of Project Components	44
4.4 Design diagram	47
5. EVALUATION	48
5.1 Evaluate the Correctness of the Analysis of Solution Architecture by SDM	50
5.1.1 Accuracy Validation	50
5.2 Performance Testing Of SDM	52
5.2.1 Performance Testing By the Complexity of the Solution Architecture and Dependency Projects	53
5.2.2 Performance Testing By Number of Project Included in the Solution	54
5.3 Analytical Evaluation of the Impact of SDM When It Is Added To a Continuous Integration Flow	56
6. CONCLUSION	58
6.1 Research Contribution	59
6.2 Research Limitations	60
6.3 Future Work and Conclusion	61
REFERENCES	62

LIST OF FIGURES

Figure 2.1.1: DevOps	10
Figure 2.2.1: Continuous Integration flow	13
Figure 2.6.1: Layered Diagram	24
Figure 2.6.2: Layered Diagram in Visual Studio	25
Figure 3.2.1: Visio Tool Usage for Design Documents	28
Figure 3.2.2: High Level Architecture Diagram of the Monitoring Framework	29
Figure 4.1.1: Properties of a Solution	33
Figure 4.1.1.1: Sample Solution View	34
Figure 4.1.1.2: Sample .sln File	35
Figure 4.1.2.1: Implementation of Solution File Observer	35
Figure 4.1.2.2: Implementation of Solution Project	36
Figure 4.1.2.3: Solution Directory Hierarchy	36
Figure 4.2.1.1: Overview of Project File	38
Figure 4.2.2.1: Overview of XML File Element Hierarchy	39
Figure 4.2.2.2: Implementation of Project Content Reader	39
Figure 4.2.2.2: Implementation of Project File Node Observer	40
Figure 4.2.2.3: Generated XML Structure	40
Figure 4.2.2.3: Observed Project Detail Hierarchy	41
Figure 4.3.1: Implementation of Design Helper	41
Figure 4.3.1.1: Implementation of Design Drawer	42
Figure 4.3.1.2: Implementation of Geometry for Design Document	43

Figure 4.3.2.1: Implementation of Mapping Relationship	44
Figure 4.3.2.2: Component Placement in Document	45
Figure 4.3.2.3: Automates Unit Tests	46
Figure 4.4.1: Generating Architecture Document	47
Figure 5.1: Basic Flow of the Solution Design Monitor	49
Figure 5.2: Basic Components of Solution Design Monitor	50
Figure 5.3: Unit Test Summary	50
Figure 5.1.1: List of Used Projects	51
Figure 5.2.1.1: Average Execution Time with the Complexity of the Project Dependencies	54
Figure 5.2.2.1: Change of the Average Execution Time with the Number of Projects in a Solution.	55
Figure 5.3.1: Build Pipeline before Adding SDM	56
Figure 5.3.2: Release Pipeline before Adding SDM	56
Figure 5.3.3: Build Pipeline after Adding SDM	57
Figure 5.3.4: Release Pipeline after Adding SDM	57

LIST OF TABLES

Table 5.1.1: Architecture Evaluation Results	52
Table 5.2.1.1: Average Execution Time of SDM with the Complexity of the Project Dependencies	53
Table 5.2.2.1: Change of the Average Execution Time with the Number of Projects in a Solution.	55