

ARCHITECTURE FOR AUTOMATIC SOURCE CODE COMMENT GENERATION



R.M.N.S Samarasinghe

(199360G)

This dissertation was submitted in partial fulfilment of the requirements
for the degree of MSc in Computer Science Specializing in Software
Architecture

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2022

DECLARATION

I declare that this is my work. This Thesis Report 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 previously published material written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or other media. I retain the right to use this content in whole or interest in future works.

.....
R.M Nuwan Shashika Samarasinghe

.....
Date

I certify that the declaration above by the candidate is accurate to the best of my knowledge and that this project report is acceptable for evaluation for the Thesis Report.

.....
Dr. Indika Perera
(Supervisor)

.....
Date

ABSTRACT

This research tries to give a highly available software architecture to a source code commenting tool. Code commenting is helpful for many phases like code updates, bug fixes, etc. Developers need to know which task each code is performing, and also, adding comments for each code is essential. But most developers take more time to add comments which has created the problem of wasting time. Using an automated comment generator tool for source code, we can avoid that problem and save time for another task. Currently, there is not a properly defined architecture for an automated comment generator tool. Therefore, we are trying to give an evaluated architecture for that tool that supports overcoming the problem.

In this architecture, we plan to follow the microservice architectural concepts. Then modularization of each component based on the service it is going to provide. Also, plan to use ActiveMQ for the queue technique, Scheduling techniques to reschedule things, No SQL databases for data saving, Caching techniques to store temporary data, etc. This procedure ensures that it will provide a highly available architecture. Also, this architecture will have the following quality attributes: Maintainability, Performance, Interoperability, Usability, Availability, Reliability, Testability, Modifiability, Scalability, and Reusability. Finally, we use presentations, proof of concept, and ATAM & CBAM methods to validate whether the architecture is commercially viable or not.

Key Words: Development, Zookeeper, ActiveMQ, REST, Database, Queue, Asynchronous, Cache.

ACKNOWLEDGEMENTS

My profound gratitude goes to Dr Indika Perera, my supervisor, for the knowledge, supervision, advice, and guidance provided with his expertise throughout making the thesis a success.

My appreciation goes to my family for the motivation and support provided throughout my life. Also, I would like to thank my colleagues in the MSc batch and at my workplace for the help and support provided in managing my research work.

TABLE OF CONTENTS

1	INTRODUCTION	7
1.1	IMPORTANCE OF CODE COMMENTING	7
1.2	IMPORTANCE OF SOFTWARE ARCHITECTURE.....	8
1.3	PROBLEM	9
1.4	MOTIVATION	10
1.5	OBJECTIVES	11
2	LITERATURE REVIEW	12
2.1	SOFTWARE CODE COMMENTING COMPONENT	12
2.2	CHALLENGERS OF AUTOMATIC CODE COMMENTING.....	15
2.3	CLASSIFICATION TYPE 01.....	16
2.3.1	TEMPLATE-BASED TECHNIQUES	16
2.3.2	KEYWORD-BASED TECHNIQUES.....	19
2.4	CLASSIFICATION TYPE 02.....	20
2.4.1	VSM/LSI BASED COMMENT GENERATION ALGORITHMS	20
2.4.2	CODE CLONE DETECTION-BASED COMMENT GENERATION 22	
2.4.3	LDA BASED COMMENT GENERATION	22
2.5	OTHER INFORMATION RETRIEVAL-BASED	23
2.6	DEEP NEURAL NETWORKS-BASED COMMENT GENERATION	23
2.6.1	RNN BASED COMMENT GENERATION	24
2.6.2	OTHER NEURAL NETWORK-BASED COMMENT GENERATION 25	
2.7	QUALITY EVALUATION OF GENERATED COMMENTS	26
2.8	COMPONENT ARCHITECTURES	27
2.8.1	TEXT FILE READ (I/O OPERATIONS)	27
2.8.2	SOFTWARE ARCHITECTURE DESIGN	30
2.8.3	DOCKER ENVIRONMENT	31
2.8.4	DOCKER SWARM	33
2.8.5	KUBERNETES.....	34
2.8.6	LOCAL STACK	36
3	METHODOLOGY	37

3.1	HIGH LEVEL ARCHITECTURE BUILD FOR CODE COMMENTING TOOLS	38
3.2	WHAT ARE WE TRYING TO GIVE IN THE SYSTEM?	41
3.2.1	FILE UPLOADER (CLIENT-END APPLICATION)	43
3.2.2	COMMENT GENERATOR HELPER SERVICE API.....	46
3.2.3	COMMENT GENERATOR	46
3.3	IMPROVED ARCHITECTURE TO USE DOCKER ENVIRONMENT ..	47
3.4	DOCKER CONVERSION	48
3.5	DOCKER SWARM MIGRATION.....	51
3.6	KUBERNETES MIGRATION	52
3.7	COMPONENT CHANGERS.....	52
4	EVALUATE ARCHITECTURE.....	55
4.1	SELECTED QUALITY ATTRIBUTES IN THE DESIGN	55
4.1.1	MAINTAINABILITY	55
4.1.2	PERFORMANCE	56
4.1.3	INTEROPERABILITY	56
4.1.4	USABILITY	57
4.1.5	AVAILABILITY	57
4.1.6	RELIABILITY	58
4.1.7	TESTABILITY	58
4.1.8	MODIFIABILITY	58
4.1.9	SCALABILITY	59
4.1.10	REUSABILITY	59
4.2	PRESENTATIONS	60
4.3	FORMAL REVIEWS AND STRUCTURED WALKTHROUGHS	62
4.4	PROTOTYPES AND PROOF-OF-CONCEPT SYSTEMS	63
4.5	SCENARIO-BASED ASSESSMENT APPROACHES	64
4.5.1	ATAM (ARCHITECTURE TRADE-OFF ANALYSIS METHOD)..	64
4.5.2	SWOT ANALYSIS FOR ARCHITECTURE EVALUATION	71
4.5.3	COCOMO MODEL ARCHITECTURE EVALUATION.....	73
4.5.4	COST-BENEFIT ANALYSIS METHOD (CBAM)	78
4.6	EVALUATION (INDUSTRY EXPERTS).....	82
5	PERFORMANCE.....	84

6	SUMMARY	90
7	PROBLEMS AND CHALLENGES	92
8	FUTURE WORK.....	93
9	CONCLUSION.....	94
10	REFERENCES.....	95
11	APPENDIX.....	99
11.1	LIST OF FIGURES.....	99
11.2	SOFTWARE DEVELOPMENT TASK COST DOCUMENT.....	101
11.3	COST ANALYSIS FOR AWS RESOURCES	103
11.4	INDIRECT COST ANALYSIS	104
11.5	INTANGIBLE COST ANALYSIS.....	104
11.6	TOTAL CALCULATED COST	104