

**FEATURE ORIENTED SOFTWARE DEVELOPMENT
METHODOLOGY FOR STOCK EXCHANGE SYSTEMS**

Lasitha Harinda Konara

(168235T)

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2018

FEATURE ORIENTED SOFTWARE DEVELOPMENT METHODOLOGY FOR STOCK EXCHANGE SYSTEMS

Lasitha Harinda Konara

(168235T)

Thesis submitted in partial fulfillment of the requirements for the
degree Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

June 2008

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 dissertation, 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:

Name: K.M.G.L.H. Konara (168235T)

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

Name of the supervisor: Dr. Indika Perera

Signature of the supervisor:

Date:

Abstract

Many organizations that develop software use the traditional method of layered methodologies to develop their end software product or solution. In doing so, the code will be a more general one and there will be a lot of unnecessary elements included which make the system heavy and dirty. This would result in a lot of issues. Also there is a requirement to implement a system with a concept of features. The end system will be delivered as a set of features and the feature set could be decoupled at any time, according to the current requirement without harming any existing functionality.

This research has been narrowed down to a particular domain which is the stock exchange or trading domain. By narrowing down the domain, the end software product could be delivered in a tailor made manner so that its effectiveness will be very high. The final software product would be a feature oriented domain specific language (DSL).

The objective of the feature oriented DSL is to make it very effective even for business analysts to introduce new features without getting help from core software developers. The feature layer will be purely decoupled and presented in an independent way so that the end users will have full flexibility to introduce changes very easily. There is a clear separation between core code segments and auto generated code segments. Auto generated files serve the different features and core code segments will enable those features to function on top of them. Auto generated code should not be changed manually under any circumstance as per this design.

A code generator and the core controller is developed throughout this research exhibiting the above mentioned feature oriented software development principles and domain specific language principles.

Keywords : FOSD, DSL, FOP, AOP, ANTLR, Entity, Instance

ACKNOWLEDGEMENT

My heartfelt gratitude is given first and foremost to the academic supervisor of this research Dr.Indika Perera for his immense help, support and advice given throughout the course of this research. His encouragement and contribution in the form of alternative methods, ideas and concepts provided motivation for us to move forward with this research.

My special thanks goes out to my external research supervisors/advisors, Mr.Manoj Bandara, Mr.Sujith Gunawardhane, Mr.Surith Pinto and Mr.Sampath Thilakumara for giving me the opportunity to work on this research, providing the guidance .Also I would like to thank all the staff members in MillenniumIT Software (Pvt) Ltd.

Academic staff members' valuable guidance and advice since the very beginning of this research seminar series is also highly appreciated. Specially I should appreciate the support and advice given by Dr. Malaka Walpola for encouraging us to do research.

My gratitude is also extended to all the staff members of the Computer Science and Engineering Department who provided us with numerous advice and feedback, especially at reviews, feasibility study. I appreciate your guidance.

TABLE OF CONTENTS

Declaration	i
Abstract	i
Acknowledgement	iii
List of Figures	viii
List of Tables	ix
List of Abbreviations	x
1 Introduction	1
1.1 Background	2
1.2 Problem Statement	2
1.3 Objectives	4
1.4 Feature Oriented Software Development (FOSD)	4
1.4.1 What is Feature Oriented Software Development?	4
1.4.2 How FOSD could be used to provide a solution	5
1.5 Outline	5
2 Literature Review	6
2.1 Overview	7
2.2 What is a feature ?	7
2.3 Different phases of feature oriented software development	8
2.3.1 Feature Modeling	11
2.3.2 Feature Interaction	11

2.3.3	Feature Implementation	12
2.4	Related Implementations	13
2.4.1	FeatureC++	13
2.4.2	Feature Implementation	13
2.4.3	Algebraic Hierarchical Equations for Application Design (AHEAD)	14
2.4.4	Model Concepts	14
2.5	Access Control in Feature Oriented Programming	16
2.6	Domain Specific Languages (DSL)	18
2.7	Summary of the Literature Review	19
3	Research Methodology	20
3.1	High Level Architecture	21
3.2	Progress	22
3.3	Evaluation Methodology	22
3.4	Time line	24
4	System Architecture and Implementation	25
4.1	System Overview	26
4.2	System Architecture	26
4.2.1	Architecture of Code Generator & Controller	26
4.2.2	Class Diagram of Code Generator	28
4.2.3	Class Diagram of Controller	29
4.3	System Implementation	30
4.3.1	Implementation of Code Generator	30
4.3.2	Introduction of Domain Specific Language (DSL)	30
4.3.3	Use of Antlr to define the grammar	31
4.3.4	Sample DSL Code	35
4.3.5	Auto generated C++ code	36
4.3.6	Auto generated C++ header	38
4.3.7	Use of StringTemplates to generate the code	38

4.3.8	Challenges faced	39
4.3.9	Key assumptions made	39
4.3.10	Implementation of Controller	39
4.3.11	Summary of implementation	50
5	System Evaluation	51
5.1	Overview	52
5.2	Evaluation of Code Generator	52
5.3	How to provide inputs	52
5.4	Evaluating the outputs	55
5.4.1	How to evaluate the outputs	55
5.4.2	Automated validator	55
5.4.3	Comparison between manually calculated results,automated test results and actual results	56
5.4.4	Evaluation of results	57
6	Conclusion	59
6.1	Contribution	60
6.2	Study limitations	60
6.3	Future work	61
	BIBILOGRAPHY	62
	APPENDIX	67
A	Detailed Test Results	68

LIST OF FIGURES

Figure 1.1	Layered product architecture	2
Figure 2.1	Problem space and solution space [8]	8
Figure 2.2	Domain analysis [3]	9
Figure 2.3	Domain design and specification [3]	9
Figure 2.4	Domain implementation [3]	10
Figure 2.5	Product configuration and generation [3]	10
Figure 2.6	Feature Modeling [3]	11
Figure 2.7	Feature Implementation [3]	12
Figure 2.8	Convergence of each step in FOSD [3]	12
Figure 2.9	Stack of Mixin Layers [9]	13
Figure 2.10	Organization of AHEAD generators. [12]	15
Figure 2.11	Energy Aware Feature Model [26]	19
Figure 3.1	Code generation based on selected features [3]	21
Figure 3.2	Research Time-line	24
Figure 4.1	Architecture of Code Generator	27
Figure 4.2	Code Generator	28
Figure 4.3	Controller	29
Figure 4.4	Entity	40
Figure 4.5	Features	41
Figure 4.6	Feature Container	42
Figure 4.7	Order,Instrument,Trading Parameter Instances	43
Figure 4.8	Reference Data Container	44
Figure 4.9	Order Book	45

Figure 4.10	Order Book Side	45
Figure 4.11	Directional Map	47
Figure 4.12	Order Handler	49
Figure 4.13	Order Injector	49
Figure 4.14	Logger	50
Figure 5.1	Evaluation methodology	55
Figure 5.2	Summary of test results	56
Figure A.1	Automated test result for test case in Table 5.1	68
Figure A.2	Actual result extracted from log file for test case in Table 5.1	68
Figure A.3	Automated test result for test case in Table 5.2	69
Figure A.4	Actual result extracted from log file for test case in Table 5.2	69
Figure A.5	Automated test result for test case in Table 5.3	70
Figure A.6	Actual result extracted from log file for test case in Table 5.3	70
Figure A.7	Automated test result for test case in Table 5.4	71
Figure A.8	Actual result extracted from log file for test case in Table 5.4	71
Figure A.9	Automated test result for test case in Table 5.5	72
Figure A.10	Actual result extracted from log file for test case in Table 5.5	72
Figure A.11	Automated test result for test case in Table 5.6	73
Figure A.12	Actual result extracted from log file for test case in Table 5.6	73
Figure A.13	Automated test result for test case in Table 5.7	74
Figure A.14	Actual result extracted from log file for test case in Table 5.7	74
Figure A.15	Automated test result for test case in Table 5.8	75
Figure A.16	Actual result extracted from log file for test case in Table 5.8	75
Figure A.17	Automated test result for test case in Table 5.9	76
Figure A.18	Actual result extracted from log file for test case in Table 5.9	76
Figure A.19	Automated test result for test case in Table 5.10	77
Figure A.20	Actual result extracted from log file for test case in Table 5.10	77

LIST OF TABLES

Table 2.1	Problems and Solutions given by FeatureC++ [9]	14
Table 2.2	Which members of class could be accessed by a refinement in each language[10]	17
Table A.1	Putting a limit new order	68
Table A.2	Putting limit new order with higher priority	69
Table A.3	Putting sell order for partial match	70
Table A.4	Putting a sell order for full match	71
Table A.5	Putting a sell order for full match and adding remaining quantity to order book	72
Table A.6	Putting a sell order with a higher priority	73
Table A.7	Order amend without losing priority	74
Table A.8	Order amend while losing priority	75
Table A.9	Market order with expiry	76
Table A.10	Order cancellation	77

LIST OF ABBREVIATIONS

Abbreviations	Description
FOSD	Feature Oriented Software Development
FOP	Feature Oriented programming
AOP	Aspect Oriented Programming
AHEAD	Algebraic Hierarchical Equations for
.	Application Design
DSL	Domain Specific Language
VPL	Visual Programming Language