

ASPECT-ORIENTED MODEL-DRIVEN CODE GENERATION APPROACH  
FOR IMPROVING CODE REUSABILITY AND MAINTAINABILITY

ABID MEHMOOD

A thesis submitted in fulfilment of the  
requirements for the award of the degree of  
Doctor of Philosophy (Computer Science)

Faculty of Computing  
Universiti Teknologi Malaysia

MAY 2014

I declare that this thesis entitled “*Aspect-Oriented Model-Driven Code Generation Approach for Improving Code Reusability and Maintainability*” is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



Signature : .....

Name : ABID MEHMOOD  
.....

Date : 15/05/2014  
.....

To my beloved parents, wife Irum, and our children Shuja and Rameen

## ACKNOWLEDGEMENT

First, I am grateful to Almighty Allah (swt) for always being the source of strength, well-being, and gratification for me throughout my life and particularly through the period of this research work.

I am thankful to Dr. Dayang Jawawi, my supervisor, for her vibrant and invaluable guidance. Dr. Jawawi has continuously supported me during this research, reviewing my work, providing clear advice, and entertaining numerous questions with kind gestures.

I am highly obliged to my parents, brothers and sisters for their encouragement, endless support and compassionate prayers. Their prayers have always been a source of inspiration and encouragement for me. Here, I also acknowledge the support of my dear friend and brother, Shabir Ahmed, who has helped me along in ways he may not even know.

I owe a great debt of gratitude to my wife, Irum, for her love, huge patience, support and motivation. This work would not have been accomplished without her help and support indeed. Besides her, I also express much appreciation to our children Shuja and Rameen, for helping me get through the hard times, and understanding at the moments I was not with them.

## ABSTRACT

Software development teams always need methods that can help in producing high-quality software with reduced development effort and delivery time. Model-Driven Engineering (MDE) as well as Aspect-Oriented Software Development (AOSD) techniques help in reducing the delivery time, and also positively contribute to quality of the produced software. Through the use of AOSD techniques in combination with MDE, an integration of excellent abstraction mechanisms of MDE and capabilities of AOSD with regards to modularity and composition of concerns can be perceived, which is expected to enhance the positive effects of both techniques. To this end, different integration approaches have appeared in literature, but aspect-oriented code generation has advantages over the other approaches. Consequently, a number of aspect-oriented code generation approaches have been offered, but all such approaches lack several features mandatory to materialize a workable integration of aspect technologies in the context of MDE. To address these issues, this research was conducted to present an approach for aspect-oriented model-driven code generation, which focuses on elaborating the conceptual relationship between design models and the implementation code, and exploits the same to obtain aspect-oriented code that is more reusable and maintainable. The key outcomes of this research are the elaboration of the conceptual mappings between elements of visual design and constructs of the code, mapping of the visual models to implementation-specific text-based models, and a technique for generation of aspect-oriented code. The applicability of the proposed approach is shown by the use of case studies, whereas the quality of the approach is measured using reusability and maintainability metrics. A comparison of the proposed approach with existing approaches substantiates its efficacy in terms of reusability and maintainability of code, showing an outperformance of other approaches by the proposed approach against 78% of the employed quality metrics.

## ABSTRAK

Pasukan pembangunan perisian sentiasa memerlukan kaedah yang boleh membantu dalam menghasilkan perisian yang berkualiti tinggi dengan mengurangkan usaha pembangunan dan masa penghantaran. Kejuruteraan berpandukan model (MDE) dan teknik Pembangunan Perisian Berorientasikan Aspek (AOSD) membantu dalam mengurangkan masa penghantaran dan juga menyumbangkan kepada kualiti perisian yang dihasilkan. Dengan penggunaan teknik-teknik AOSD yang digabungkan dengan MDE, gabungan ini dijangka boleh memperbaiki kesan positif kedua-dua teknik dengan integrasi mekanisme pengabstrakan MDE yang baik dan kemampuan AOSD dari segi k modularan dan komposisi yang boleh diamati. Untuk ini, pendekatan-pendekatan integrasi yang berbeza telah muncul di dalam literatur, tetapi penjanaan kod berorientasikan aspek mempunyai banyak kelebihan berbanding pendekatan yang lain. Walaupun, pelbagai pendekatan penjanaan kod berorientasikan aspek telah ditawarkan, tetapi kesemua pendekatan tersebut kurang dari segi ciri-ciri mandatori untuk membolehkan integrasi teknologi aspek di dalam konteks MDE dilaksanakan. Bagi menyelesaikan kekurangan ini, kajian ini telah dijalankan untuk menunjukkan satu pendekatan penjanaan kod berorientasikan aspek dan berpandukan model, di mana ia fokus kepada penghuraian hubungan konseptual di antara reka bentuk model dan pelaksanaan kod, dan mengolah kedua-duanya untuk mencapai kod berorientasikan aspek yang mempunyai kadar kebolegunaan semula dan kebolehsenggaraan yang lebih. Penghasilan utama kajian ini adalah penghuraian pemetaan elemen-elemen reka bentuk visual dan pembinaan kod, pemetaan model visual untuk model berasaskan teks dan pelaksanaan-spesifik dan teknik untuk penjanaan kod berorientasikan aspek. Kebolegunaan pendekatan yang dicadangkan ini telah ditunjukkan melalui kajian kes, manakala kualiti pendekatan diukur menggunakan metrik kebolegunaan semula dan kebolehsenggaraan. Perbandingan di antara pendekatan yang dicadangkan dengan pendekatan-pendekatan sedia ada berkaitan keberkesannya dari segi kebolegunaan semula dan kebolehsenggaraan kod, menunjukkan pendekatan yang dicadangkan mengatasi pendekatan-pendekatan lain sebanyak 78% dengan menggunakan metrik pengukuran kualiti.

## TABLE OF CONTENTS

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE</b>
	<b>DECLARATION</b>	ii
	<b>DEDICATION</b>	iii
	<b>ACKNOWLEDGEMENT</b>	iv
	<b>ABSTRACT</b>	v
	<b>ABSTRAK</b>	vi
	<b>TABLE OF CONTENTS</b>	vii
	<b>LIST OF TABLES</b>	xvii
	<b>LIST OF FIGURES</b>	xx
	<b>LIST OF ABBREVIATIONS</b>	xxiv
	<b>LIST OF APPENDICES</b>	xxvi
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Overview	1
	1.2 Research background	3
	1.3 Motivation	9
	1.4 Problem statement	11
	1.5 Research objectives	13
	1.6 Research scope	14
	1.7 Thesis outline	14
<b>2</b>	<b>LITERATURE REVIEW</b>	16
	2.1 Introduction	16

2.2 Integration of Aspect Orientation and MDE	18
2.2.1 Aspect-Oriented Modeling (AOM) notations	19
2.2.1.1 UML Profiles to support AOM	20
2.2.1.2 Meta-model extensions to support AOM	22
2.2.1.3 Infrastructures for the extension process	23
2.2.2 Weave-Then-Generate (WTG) approaches	24
2.2.2.1 Model weavers	24
2.2.2.2 Model unweavers	26
2.2.2.3 Base-aspect interaction enhancement techniques	27
2.2.2.4 Object-oriented code generation approaches	28
2.2.3 Generate-Then-Weave (GTW) approaches	29
2.2.3.1 Transformation-based approaches	30
2.2.3.2 Direct mapping approaches	31
2.2.3.3 Code generation from specification of NFRs	33
2.2.4 Evaluation of object-oriented code generation in WTG setting	34
2.2.4.1 Models	35
2.2.4.2 Design	35
2.2.4.3 Implementation	39
2.2.4.4 Validation	40
2.2.4.5 Extent of Code	40
2.2.4.6 Tools support	41
2.2.4.7 Discussion of results and essential problems in WTG approach	42



2.2.4.8 Discussion of reusability and maintainability problems	44
2.3 Aspect-oriented model-driven code generation – a GTW approach	45
2.3.1 Benefits of aspect-oriented code generation	45
2.3.2 Evaluation of existing aspect-oriented code generation approaches	46
2.3.2.1 Transformation	50
2.3.2.2 Models	51
2.3.2.3 Validation	52
2.3.2.4 Extent of code	53
2.3.2.5 Tool-support	54
2.3.3 Limitations of approaches with respect to integration with MDE	55
2.4 Important concepts in the integration through AO code generation	56
2.4.1 Selection of Aspect-Oriented Modeling (AOM) approach	56
2.4.1.1 General assessment	58
2.4.1.2 A modeling example-driven assessment	78
2.4.1.3 Reusable Aspect Models (RAM) for AO code generation	85
2.4.2 Conceptual mapping of models and integration of behavioral models	87
2.4.2.1 Mapping of AO models	87
2.4.2.2 Implementation of behavioral diagrams	89
2.4.3 Representation approach	92
2.4.3.1 XML as an intermediate representation	93

	2.4.3.2 Techniques to transform UML-based models into XML	94
	2.5 Summary	95
<b>3</b>	<b>RESEARCH METHODOLOGY</b>	96
	3.1 Introduction	96
	3.2 Research design	97
	3.3 Research phases	98
	3.4 Research framework	100
	3.5 Research process	102
	3.5.1 Phase 1: Primary studies and data collection	102
	3.5.1.1 Literature analysis	102
	3.5.1.2 Problem formulation	105
	3.5.2 Selection of the AOM and transformation approach	107
	3.5.2.1 Determining AOM approach	107
	3.5.2.2 Determining behavior representation approach	108
	3.5.2.3 Determining text-based model representation approach	108
	3.5.3 Mapping method development	108
	3.5.4 Text-based implementation model development	110
	3.5.5 Code generation technique development	111
	3.5.6 Validation	112
	3.5.6.1 Checking the applicability of the proposed approach using case studies	112
	3.5.6.2 Evaluating the AOMDCG relative to GTW approaches	113
	3.5.6.3 Evaluating the reusability and maintainability through metrics	113
	3.6 Quality metrics	114

3.6.1	Separation of Concerns metrics	115
3.6.1.1	Concern Diffusion over Components (CDC)	115
3.6.1.2	Concern Diffusion over Operations (CDO)	116
3.6.2	Coupling metrics	116
3.6.2.1	Coupling between Components (CBC)	116
3.6.2.2	Depth of Inheritance Tree (DIT)	117
3.6.3	Cohesion metrics	117
3.6.3.1	Lack of Cohesion in Operation (LCOO)	117
3.6.4	Size metrics	118
3.6.4.1	Vocabulary Size (VS)	118
3.6.4.2	Lines of Code (LOC)	118
3.6.4.3	Number of Attributes (NOA)	118
3.6.4.4	Weighted Operations per Component (WOC)	119
3.7	Case studies	119
3.7.1	Remote Service Caller case study	120
3.7.2	Online Book Store System case study	121
3.8	Summary	124
<b>4</b>	<b>METHOD FOR MAPPING ASPECT- ORIENTED MODELS TO ASPECT- ORIENTED CODE</b>	<b>125</b>
4.1	Introduction	125
4.2	Mapping Core properties and Structural units to code	126
4.2.1	General structure, classes, attributes and operations	127
4.2.2	Associations	128
4.2.3	Instantiation and binding directives	130
4.3	Mapping behavior	131
4.3.1	Mapping the basic state diagram	131
4.3.2	Mapping composite states	135

4.3.2.1 Composite states with sequential substates	136
4.3.2.2 Composite states with concurrent substates	137
4.4 Summary	140
<b>5 A TEXT-BASED IMPLEMENTATION MODEL FOR RAM</b>	<b>141</b>
5.1 Introduction	141
5.2 The Conceptual Reference Model for RAM	142
5.2.1 Core	142
5.2.2 StructuralView	144
5.2.3 StateView	145
5.3 The text-based implementation model	146
5.3.1 Text-based model for the Core part	147
5.3.1.1 RAM aspect	147
5.3.1.2 Mandatory instantiation parameters and instantiation directives	149
5.3.2 Text-based model for the StructuralView part	151
5.3.2.1 Interfaces and classes	151
5.3.2.2 Fields	153
5.3.2.3 Constructors and methods	153
5.3.2.4 Relationships	156
5.3.3 Text-based model for the StateView part	158
5.3.3.1 Standard statechart	158
5.3.3.2 States	158
5.3.3.3 Substates	159
5.3.3.4 Transitions	160
5.3.3.5 Aspectual statechart	163
5.4 Summary	164

<b>6</b>	<b>ASPECT-ORIENTED CODE GENERATION TECHNIQUE</b>	165
	6.1 Introduction	165
	6.2 Overview of the code generation algorithm	166
	6.3 Code generation for the core concepts	168
	6.4 Code generation for the structural part	172
	6.4.1 Classes	172
	6.4.2 Interfaces	173
	6.4.3 Constructors	174
	6.4.4 Fields	175
	6.4.5 Methods	176
	6.4.6 Relationships	178
	6.4.7 Instantiation and binding directives	178
	6.5 Code generation for the behavioral part	179
	6.5.1 Statechart	180
	6.5.2 Aspectual statechart	181
	6.6 Summary	188
<b>7</b>	<b>IMPLEMENTATION OF APPROACH: DEVELOPING A REMOTE SERVICE CALLER WITH FAILURE HANDLING FUNCTIONALITY</b>	189
	7.1 Introduction	189
	7.2 RAM models for Remote Service Caller	190
	7.3 Text-based representation of RAM models	194
	7.3.1 Network Failure Handler aspect	195
	7.3.1.1 Core properties	195
	7.3.1.2 Structural view	196
	7.3.1.3 State view	196
	7.3.2 Service Controller aspect	197
	7.3.2.1 Core properties	197
	7.3.2.2 Structural view	198

7.3.2.3 State view	199
7.4 Code generation	201
7.4.1 NetworkFailureHandler aspect	202
7.4.1.1 Context Handler	202
7.4.1.2 State Controller	203
7.4.1.3 State classes	203
7.4.2 ServiceController aspect	207
7.4.2.1 Context Handler	207
7.4.2.2 State Controller	208
7.4.2.3 Context Handler for composite state	210
7.4.2.4 State Controller for composite state	211
7.4.2.5 State Classes	212
7.4.3 Instantiation Aspect	213
7.5 Discussion	214
7.6 Summary	216
<b>8 IMPLEMENTATION OF APPROACH: ASPECT-ORIENTED DEVELOPMENT OF AN ONLINE BOOK STORE SYSTEM</b>	217
8.1 Introduction	217
8.2 Reusable Aspect Models for Online Book Store System	218
8.3 Text-based representation of OBSS RAM models	224
8.3.1 Persistence, Currency Conversion and Encryption aspects	225
8.3.2 Order base implementation as a RAM aspect	225
8.4 Code generation	227
8.4.1 Persistence, Currency Conversion and Encryption aspects	228
8.4.2 Order base implementation as RAM aspect	236
8.4.2.1 Order Context Handler	237
8.4.2.2 Order State Controller and associated states	242
8.4.2.3 Composite state controller and associated	

	classes	243
	8.4.2.4 Other classes in the model	247
	8.5 Discussion	249
	8.6 Summary	250
<b>9</b>	<b>QUALITY MEASUREMENT OF THE APPROACH</b>	<b>251</b>
	9.1 Introduction	251
	9.2 Evaluation of AOMDCG relative to GTW approaches	252
	9.2.1 Transformation	252
	9.2.2 Models	254
	9.2.3 Validation	254
	9.2.4 Extent of code	254
	9.2.5 Tool-support	255
	9.3 Measurement of reusability and maintainability	256
	9.3.1 Separation of Concerns (SOC)	257
	9.3.1.1 Concern Diffusion over Components (CDC)	257
	9.3.1.2 Concern Diffusion over Operations (CDO)	259
	9.3.2 Coupling	260
	9.3.2.1 Coupling Between Components (CBC)	260
	9.3.2.2 Depth of Inheritance Tree (DIT)	262
	9.3.3 Cohesion	263
	9.3.3.1 Lack of COhesion in Operations (LCOO)	263
	9.3.4 Size	264
	9.3.4.1 Vocabulary Size (VS)	264
	9.3.4.2 Lines Of Code (LOC)	266
	9.3.4.3 Number Of Attributes (NOA)	267
	9.3.4.4 Weighted Operations per Component (WOC)	268
	9.4 Discussion of results	269
	9.5 Summary	271

<b>10</b>	<b>CONCLUSION</b>	272
	10.1 Summary	272
	10.2 Achievement of research objectives	273
	10.3 Research contributions	274
	10.4 Future work	276
	<b>REFERENCES</b>	278
	Appendices A-E	300-344