



UNIVERSITI PUTRA MALAYSIA

**MEDIATOR CONNECTOR FOR COMPOSING LOOSELY COUPLED
SOFTWARE COMPONENTS**

Hamid Sanatnama

FSKTM 2009 6



**MEDIATOR CONNECTOR FOR COMPOSING LOOSELY COUPLED
SOFTWARE COMPONENTS**

By

Hamid Sanatnama

**Thesis Submitted to the School of Graduate Studies, University Putra Malaysia in
Fulfilment of the Requirement for the Degree of Doctor of Philosophy**

June 2009



DEDICATION

To my dearest wife and my lovely children for the endless support, encouragement and patience, and also those who made this study possible.



Abstract of thesis presented to the Senate of University Putra Malaysia in fulfillment of the requirement for the degree of Doctor of Philosophy

**MEDIATOR CONNECTOR FOR COMPOSING LOOSELY COUPLED
SOFTWARE COMPONENTS**

By

HAMID SANATNAMA

June 2009

Chairman: Associate Professor Abdul Azim Abdul Ghani, PhD

Faculty: Computer Science and Information Technology

Component-Based Software Development (CBSD) is an approach that has many benefits, such as improving application developer productivity, reducing costs and complexity by reusing of existing codes. Programming within this approach is like assembly (i.e. composing software out of prefabricated components) rather than development, which reduces skill requirements, and allows expertise focuses on domain problems. The foundation of any CBSD methodology is its underlying component model, which defines what components are, how they can be constructed, and specifies the standards and conventions that are needed to enable composition of independently developed component. The current component models do not support composition in both design and deployment phase. They also focus on the specification and packaging of components but provide almost no support for the easy composition of components. Component in these models uses either direct or indirect message passing as connection schemes, which leads to tightly coupling (i.e. components mix computation with



control). It is conclude that this research has proposed an effective way for component composition which provides loosely coupling between composed components. For system maintenance and evolution, this decoupling should make it simpler to manage changes in the components, and also changes in the connector separately. This research has resulted in the proposed of mediator connector which is similar to a communication hub. It initiates method calls and manages the returns, and also provides total loosely coupling between components and also itself. Mediator connector is a framework and can be reused without any modification. The components composition using mediator connector belongs to the deployment phase. Our approach is based on interactions between components as a subset of behavior in a system. In order to minimize coupling between components and mediator connector, we have designed and developed an XML-based language, called Component Interaction Markup Language (CIML), where components as well as their interactions are described in a CIML document which are used by mediator connector. For evaluation of mediator connector in order to measure the loosely coupling it provides, four case studies have been tested. To measure coupling we applied Coupling Between Object (CBO) software metric. The result shows that mediator connector provides totally loosely coupling between software components composing in the deployment phase.



Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

**PENGHUBUNG PENENGAH UNTUK MENGGOMPOS KOMPONEN
PERISIAN KEBERGANTUNGAN LONGGAR**

Oleh

Hamid Sanatnama

June 2009

Pengerusi: Profesor Madya Abdul Azim Abd. Ghani, PhD

Fakulti: Sains Komputer dan Teknologi Maklumat

Pembangunan Perisian Berasaskan Komponen (PPBK) adalah satu pendekatan yang mempunyai banyak faedah, seperti mempertingkatkan produktiviti pembangun aplikasi, mengurangkan kos dan kompleksiti dengan mengguna semula kod yang ada. Pengaturcaraan dalam kaedah ini adalah seperti pemasangan (iaitu mencantumkan perisian menggunakan komponen yang telah terhasil) tidak seperti pembangunan, yang mana mengurangkan keperluan terhadap kemahiran, dan membenarkan pakar untuk memfokus kepada permasalahan utama. Asas kepada mana-mana metodologi CBSD adalah model komponen asasnya, yang mana mentakrifkan apakah dia komponen, bagaimana untuk membinanya, dan menspesifikasi piawaian dan peraturan yang diperlukan untuk membolehkan pembentukan komposisi komponen yang dibangunkan berasingan. Model komponen semasa tidak menyokong komposisi dalam kedua-dua fasa reka bentuk dan penyerahan. Mereka memfokus kepada spesifikasi dan pembungkusan komponen tetapi hampir tidak menyokong untuk mempermudah komposisi bagi komponen. Komponen dalam model ini menggunakan penyerahan mesej sama ada

secara langsung atau tidak langsung sebagai skema penghubungnya, yang menyebabkan kebergantungan yang tinggi (iaitu pengiraan komponen berserta kawalan). Kaedah yang dijelaskan di dalam tesis ini adalah hasil daripada penyelidikan untuk mencadangkan kaedah yang lebih berkesan bagi komposisi komponen yang mana menghasilkan kebergantungan yang longgar di antara komponen yang dibangunkan. Bagi penyelenggaraan sistem dan evolusi, penyah-kebergantungan sepatutnya menyebabkan pengurusan perubahan dalam komponen yang lebih mudah, dan juga perubahan dalam penghubung secara berasingan. Penyelidikan ini membuahkan hasil kepada cadangan satu penengah penghubung yang serupa seperti hub komunikasi. Ia memberi nilai awal kepada panggilan 'method' dan menguruskan 'returns', dan juga menyediakan kebergantungan yang sangat longgar antara komponen dan dalamannya. Penengah penghubung adalah sebuah rangka kerja dan boleh diguna semula tanpa sebarang pengubahsuaian. Komposisi komponen menggunakan penengah penghubung dimiliki oleh fasa penyerahan. Pendekatan kami adalah berdasarkan interaksi antara komponen sebagai subset bagi perlakuan dalam sesebuah sistem. Bagi tujuan meminimalkan kebergantungan antara komponen dan penengah penghubung, kami telah merekabentuk dan membangunkan sebuah bahasa berasaskan XML, dipanggil Component Interaction Markup Language (CIML), di mana komponen serta interaksinya dijelaskan di dalam dokumen CIML yang akan digunakan sebagai penengah penghubung. Tesis ini tidak merangkumi komposisi komponen pada fasa reka bentuk, tetapi kami percaya bahawa penengah penghubung dan CIML boleh digunakan walaupun pada komposisi fasa reka bentuk. Bagi tujuan ini kami juga mencadangkan rangka kerja yang menyokong komposisi di kedua-dua fasa reka bentuk dan penyerahan menggunakan penengah penghubung dan CIML. Bagi penilaian penengah penghubung untuk mengukur

kelonggaran kebergantungan yang disediakan, empat kajian telah dilakukan. Untuk mengukur kebergantungan kami menggunakan metrik perisian 'Coupling Between Obejct' (CBO). Keputusan menunjukkan bahawa penengah penghubung menghasilkan kebergantungan yang sangat longgar antara komponen perisian yang dikompos semasa fasa penyerahan.



ACKNOWLEDGEMENTS

This dissertation would not have been possible without the kindness of several individuals. I would like to express my gratitude and deepest appreciation to my supervisor, Assoc. Prof. Dr. Abdul Azim Abdul Ghani for his kindness and significant mentoring all the time and for offering his guidance and insightful comments in the aspects of developing mediator connector using CIML. Assoc. Prof. Mohd. Hasan Selamat also deserve a great deal of thanks not only for his practical guidelines, but also for sponsoring my PhD study through the Graduate Research Assistant (GRA) scheme for his research, and Dr Rodziah Atan for her kindness, and helpful advices specially for starting my thesis writing.

I am also deeply thanking all the staffs working in Faculty of Computer Science and Information Technology in UPM, especially technical staff.

My special thanks go to my dearest wife, Mahnaz Radboo, my children Elnaz and Iman for their understanding and patience throughout the course of my research and the preparation of this dissertation. I would also like to express my uttermost appreciation to Assoc. Prof. Dr. Famarz Sadeghi, and Assoc. Prof. Dr. Hamid Khosravi from Faculty of Mathematic and Computer Science in Shahid Bahonar University of Kerman, who made the opportunity for me to continue my study. And also a great deal of thanks to all others who have helped me in one way or another, specially my best friend Reza Meimandi Parizi.



This thesis submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee are as follows:

Abdul Azim Abdul Ghani, PhD

Associate Professor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Chairman)

Mohd. Hasan Selamat

Associate Professor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Member)

Rodziah Binti Atan

Associate Professor

Faculty of Computer Science and Information Technology

Universiti Putra Malaysia

(Member)

AINI IDERIS, PhD

Professor/Dean

School of Graduate Studies

Universiti Putra Malaysia

Date:



DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UPM or other institutions.

HAMID SANATNAMA

Date: 30 June 2009



TABLE OF CONTENTS

	Page
ABSTRACT	iii
ABSTRAK	v
ACKNOWLEDGEMENTS	viii
APPROVAL	ix
DECLARATION	xi
LIST OF TABLES	xv
LIST OF FIGURES	xvii
LIST OF ABBREVIATIONS	xxii
CHAPTER	
1 INTRODUCTION	1
1.1 Background	1
1.1.1 Composition	2
1.1.2 Composition at Design Phase	3
1.1.3 Composition at Deployment Phase	4
1.1.4 Composition Scheme	4
1.1.5 Exogenous Connectors	6
1.2 Problem Statement	7
1.3 Objectives	9
1.4 Scope of the Study	9
1.4.1 Evaluation and Verification	10
1.5 Contribution of This Study	10
1.6 Research Methodology	11
1.7 Outlines of this Dissertation	12
2 BACKGROUND IN COMPONENT-BASED SOFTWARE DEVELOPMENT AND COMPOSITION	14
2.1 Introduction	14
2.2 Component-Based Software Development	14
2.3 Composition	15
2.3.1 Scripting Languages	18
2.3.2 Composition Languages	20
2.3.3 Wrapping	22
2.3.4 The Software Bus	22
2.3.5 Connectors	23
2.4 Loosely and Tightly Coupling	25
2.4.1 Design Patterns	26
2.4.2 Creational Patterns	27
2.4.3 Structural Patterns	27
2.4.4 Behavioral Patterns	28
2.4.5 Mediator Design Pattern	30



2.5	Summary	31
3	RELATED WORK	32
3.1	Introduction	32
3.2	Component Models	33
3.2.1	JavaBeans	34
3.2.2	Enterprise Java Beans	36
3.2.3	CORBA	40
3.2.4	COM	48
3.2.5	Koala	57
3.2.6	SOFA	62
3.2.7	Architecture Description Languages (ADLs)	66
3.2.8	Exogenous Connectors	73
3.3	Summary	74
4	RESEARCH METHODOLOGY	77
4.1	Introduction	77
4.2	Literature Study on Software Component Models	78
4.3	Literature Study on Software Composition Techniques	79
4.4	Design and Development of a Connector	80
4.5	Evaluation of Mediator Connector	81
4.6	Summary	82
5	COMPONENT INTERACTION MARKUP LANGUAGE (CIML)	83
5.1	Introduction	83
5.2	Requirements for CIML	84
5.3	CIML Specification	86
5.4	Extensible Markup Language	86
5.4.1	Overview	86
5.4.2	What is XML?	87
5.4.3	XML editor	88
5.4.4	XML Structure Definition	88
5.5	EBNF grammar for CIML	90
5.6	CIML Elements Definition	92
5.6.1	Component Declaration	93
5.6.2	Initialize Declaration	95
5.6.3	Interaction Declaration	97
5.7	Implementation of CIML Specification	99
5.8	The CIML Schema Diagram	99
5.9	The CIML XML Schema Definition	101
5.10	CIML and Mediator Connector	103
5.11	Summary	103



6	PROPOSE MEDIATOR CONNECTOR	105
6.1	Introduction	105
6.2	Mediator Connector	106
6.2.1	Analysis	108
6.2.2	Design	120
6.2.3	Interaction	121
6.2.4	Class Description	126
6.2.5	Behavioral Design	141
6.2.6	Implementation	144
6.3	Summary	151
7	RESULT AND DISCUSSION	153
7.1	Introduction	153
7.2	Measuring Static Coupling	154
7.3	Measuring CBO Using Case Studies	158
7.3.1	Case one: AJcFgraph	159
7.3.2	Case two: A Simple Bank System	167
7.3.3	Case three: MP3 Player	174
7.3.4	Case four: Display Map	181
7.4	Benchmarking	189
7.5	Comparison with Exogenous Connectors	191
7.6	Summary	194
8	CONCLUSION AND FUTURE WORKS	196
8.1	Introduction	196
8.2	The Research Contribution	197
8.3	Future Works	200
	REFERENCES	203
	APENDICES	208
	BIODATA OF STUDENT	233



LIST OF TABLES

Table	Page
1.1: Taxonomy based on composition (Lau and Wang, 2005)	7
1.2: Existing component models and their composition schemes	8
2.1: Some types of scripting languages in different domains	17
3.1: Example ADLs	68
3.2: Summary of related work	76
4.1: Criteria based on number of components and total CBO	81
5.1: XML Markup Types	87
5.2: Use of XML in Other Scientific and Business Fields	89
5.3: CIML Syntax	91
5.4: CIML elements (tags)	93
5.5: Element descriptions for components declaration	93
5.6: Element descriptions for initial method calls	95
5.7: Element description for an interaction	97
6.1: Description of actors in the use case diaram	110
7.1: Measurement values of CBO components in AJcFgraph Builder Tool	165
7.2: Measurement values of CBO components in simple Bank system	172
7.3: Measurement values of CBO components in MP3 player system	179
7.4: Measurement values of CBO components in Display Map system	187
7.5: Measurement values of CBO applied on components in the Bank system	190
7.6 : Measurement values chart of CBO applied on components in DisplaMap system	190





LIST OF FIGURES

Figure	Page
1.1: Direct Message Passing	5
1.2: Indirect Message Passing	6
1.3: Research activities flowchart	12
2.1: An idealized component life cycle (Lau, Wang, 2007)	17
2.2: Design Pattern Space (Gamma et al. 1994)	26
2.3: A bank system using Mediator design patterns	30
3.1: The Enterprise JavaBean Architecture	37
3.2: CORBA Component	42
3.3: CCM Container	43
3.4: A request passing from client to the object implementation (Hamilton, 1997)	44
3.5: CosNaming IDL	46
3.6: Interoperability by ORB-to-ORB communication (Hamilton, 1997)	46
3.7: Using services provided by a COM object through interface pointer	49
3.8: A Clock COM object	50
3.9: Three methods for accessing COM objects (Microsoft, 1995)	51
3.10: Cross-process communication in COM(Microsoft, 1995)	53
3.11: Structure of proxy in COM (Microsoft MSDN, 2007)	54
3.12: Structure of the stub in COM (Microsoft MSDN, 2007)	55
3.13: An interface definition	59
3.14: A component definition	59
3.15: A Koala Compound Component (Ommering, 2000)	60



3.16: A configuration definition	61
3.17: A sample declaration of a SOFA component in CDL	63
3.18: An example of an interface and frame protocols	65
3.19: A simple example of a client server with a single client represented in Acme	69
3.20 A simple specification in Rapide	71
3.21: A simple specification in Wright	72
3.22: Architecture of a bank example using exogenous connector (Lauet al., 2005)	74
4.1: Research Methodology Overview	77
5.1: Components (Objects) in an interaction diagram.	95
5.2: Initializing in an interaction diagram	96
5.3: Interaction description part	98
5.4: XML Schema for CIML components composition structure description	100
5.5: The CIML XML Schema Definition	102
6.1: The Process of building up a system using CIML and mediator connector	108
6.2: Use case diagram of mediator connector	110
6.3: Collaboration for the Parse CIML document	111
6.4: Communication diagram for Parse CIML document	112
6.5: Collaboration diagram for Create Interaction	113
6.6: Communication diagram for Create Interaction	113
6.7: Collaboration diagram for Create ComWrapper	114
6.8: Communication diagram for Create ComWrapper	115
6.9: Collaboration diagram for Create MethodCall	116
6.10: Communication diagram for Create MethodCall	116



6.11: Collaboration diagram for Create Parameter	117
6.12: Communication diagram for Create Parameter	118
6.13: Collaboration diagram for Run Interaction	119
6.14: Communication diagram for Run Interaction	119
6.15: Mediator connector object diagram	121
6.16: An interaction using mediator connector	123
6.17: Creating Line object not using Flyweight design pattern	125
6.18: Creating Line Objects using Flyweight design patterns	125
6.19: ComWrapper Class	127
6.20: Interaction Class	129
6.21: MethodCall Class	131
6.22: Parameter Class	133
6.23: MediatorConnector Class	135
6.24: CIMLParser Class	137
6.25: CIMLContentHandler class	138
6.26: Mediator connector class diagram	140
6.27: Interaction diagram for running an interaction using mediator connector	143
6.28: System architecture using mediator connector	144
6.29: Mediator Connector's Parser API	145
6.30: Constructors of the ComWrapper Class	146
6.31: The execute method of ComWrapper class	146
6.32: Constructor of the Interaction class	147
6.33: getMethodCall method	147



6.34: Constructor of the MethodCall class	148
6.35: Constructor of the Parameter Class	149
6.36: Part of constructor code of mediator connector	149
6.37: Part of the code for a system using mediator connector	150
6.38: Invoking startParser method by mediator connector	151
6.39: Constructor of CIMLParser	151
7.1: Measurement framework	155
7.2: AJcFgraph application	160
7.3: AJcFgraph: measurement result before composition	161
7.4: AJcFgraph: class diagram before composition	161
7.5: AJcFgraph: measurement result after composition without MC	162
7.6: AJcFgraph: class diagram not using mediator connector	163
7.7: AJcFgraph: measurement result after composition using MC	164
7.8: AJcFgraph: class diagram using mediator connector	164
7.9: Measurement chart of CBO applied on components in AJcFgraph tool	165
7.10: Bank application	167
7.11: Bank: measurement result before composition	168
7.12: Bank: class diagram before composition	169
7.13: Bank: measurement result after composition without MC	170
7.14: Bank: class diagram without MC	170
7.15: Bank: measurement result after composition using MC	171
7.16: Bank: class diagram using MC	171
7.17: Measurement chart of CBO applied on components in Bank system	172
7.18: MP3Player application	174



7.19: MP3Player: measurement result before composition	175
7.20: Mp3Player: class diagram before composition	176
7.21: MP3Player: measurement result after composition without MC	177
7.22: MP3Player: class diagram without MC	177
7.23: MP3Player: measurement result after composition using MC	178
7.24: MP3Player: class diagram using MC	179
7.25: Measurement chart of CBO applied on components in MP3 Player	180
7.26: Find and show the address assigned to a phone number	182
7.27: DisplayMap: measurement result before composition	183
7.28: DisplayMap: class diagram before composition	183
7.29: DisplayMap: measurement result after composition without using MC	184
7.30: DisplayMap: class diagram after composition without MC	185
7.31: DisPlayMap: measurement result before composition	186
7.32: DisPlayMap: class diagram using MC	186
7.33: Measurement chart of CBO applied on components in Display Map system	187
7.34: Architecture of a bank example using exogenous connectors (Lau et al., 05)	192
7.35: Architecture of a bank example using mediator connector	192
7.36: Outline code of a bank system using exogenous connectors (Lau et al., 05)	193



LIST OF ABBREVIATIONS

API	Application Programming Interface
CIML	Component Interaction Markup Language
CBSD	Component-Based Software Development
GUI	Graphical User Interface
JAXP	Java API for XML Processing
JDOM	Java Document Object Model
JVM	Java Virtual Machine
SAX	Simple API for XML
W3C	World Wide Web Consortium
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSLT	eXtensible Stylesheet Language Transformation



CHAPTER 1

INTRODUCTION

1.1 Background

The design process in most engineering disciplines is based on reuse of existing systems or components. Mechanical or electrical engineers do not normally specify a design where every component has to be manufactured specially. They base their design on components that have been tried and tested in other systems.

Reuse-based software engineering can be compared to software engineering strategy where the development process is geared to reusing existing software. Although the benefits of reuse have been recognized for many years (McIlory, 1968), it is only the past 10 years that there has been an evolution from original software development to reuse-based development.

Many techniques have been developed to support software reuse during the past 20 years (Sommerville, 2004). Reuse is possible at different levels (from simple function to complete application). There are a number of ways to support software reuse:

- Legacy system wrapping
- Design patterns
- Program libraries



- Configurable vertical application
- Components Off The Shelf (COTS) integration
- Application product lines
- Program generators
- Component frameworks
- *Component-based Software Engineering (CBSE)*
- Aspect-oriented software development
- Service-oriented systems

Component-based Software Engineering (CBSE) emerged in the late 1990s as a reuse-based approach to software system development. CBSE is the process of defining, implementing and integrating or composing the loosely coupled and independent developed components into system.

The foundation of any Component-Based Software Development (CBSD) methodology is its underlying component model, which defines what components are, how they can be constructed, how they can be composed or assembled. Components communicate with their environment only through their interfaces, so it is the interface which provides all the information needed.

1.1.1 Composition

Generally *composition* means taking two or more constructs and putting them together in some way. Composition is a constructive operation—its result is a new thing that has

