12-2003

Towards a Taxonomy of Aspect-Oriented Programming.

Mario Bernard Hankerson
East Tennessee State University

Towards a Taxonomy of Aspect-Oriented Programming

A thesis

presented to

the faculty of the Department of Computer and Information Sciences

East Tennessee State University

In partial fulfillment

of the requirements for the degree

Master of Science in Computer Science

---

by

Mario B. Hankerson

December 2003

Dr. Martin L. Barrett, Chair
Steven L. Jenkins
Jeffrey W. A. Roach

Keywords: Aspect-Oriented Programming, AOP, Aspects, Components, Concerns, Crosscutting Concerns, Join Points, Object-Oriented Programming, OOP, Post-Object Programming, POP, Separation of Concerns, Taxonomy, Weaving

ABSTRACT

TOWARDS A TAXONOMY OF Aspect-Oriented Programming

by

Mario B. Hankerson

As programs continue to increase in size, it has become increasingly difficult to separate concerns into well localized modules, which leads to code tangling- crosscutting code spread throughout several modules. Thus, aspect-oriented programming (AOP) offers a solution to creating modules with little or no crosscutting concerns. AOP presents the notion of aspects, and demonstrates how crosscutting concerns can be taken out of modules and placed in a centralized location.

In this paper, a taxonomy of aspect-oriented programming, as well as a basic overview and introduction of AOP, will be presented in order to assist future researchers in getting started on additional research on the topic. To form the taxonomy, over four-hundred research articles were organized into fifteen different primary categories coupled with sub-categories, which show where some of the past research has been focused. In addition, trends of the research were evaluated and paths for future exploration are suggested.

ACKNOWLEDGEMENTS

Most importantly, I would like to thank God for assisting me during this challenging and sometimes daunting process, which has been emotionally and physically draining. Nevertheless, finishing the AOP thesis has proven to be an exhilarating liberation.

I would like to give thanks to my committee members, Dr. Barrett, Steven Jenkins, and Jeffery Roach, for their continued support in helping me to achieve my goals. In addition, I would like to honor a few high caliber friends and intellectual scholars of exceptional acumen, i.e., Dr. Gordon Bailes Jr., Robert Nielsen, and Dr. Phillip Pfeiffer IV, for helping me to realize my competencies and improve on my inadequacies as a computer scientist and person.

To Dr. Barrett: I am grateful for everything you have done for me and I truly appreciate your patience and honesty.

To Steven: I am forever indebted to you for directing me towards my thesis topic, and the days of personal counsel, which enabled me to determine the appropriate way to handle certain life situations.

To Jeffery: Thank you for your continuous support and encouragement during this process.

To Dr. Bailes: I can not adequately express how much your data structures course changed my attitude about academia and the importance of due diligence. Thank you so very much for helping me understand Sir Isaac Newton’s three laws of motion, specifically his third law in the context of computer science and its correlation to intolerable indolent behaviors as a student.

To Robert: A man of unquestionable ethics, morals, and values, whom I admire and strive to emulate. I am grateful to you Mr. Nielsen for offering countless tidbits of ageless wisdom, and dialogue about life and philosophical issues, which kept me grounded in reality during this tedious process. In addition, you are a great friend, particularly for allowing me to mount my "soap box" and discuss all the frustration and successes encountered throughout the many phases of this research. Lastly, the time spent working with you as an assistant system administrator has taught me invaluable information about professionalism.

To Dr. Pfeiffer: The dedication you possess and demonstrate toward your work has profoundly influenced me. You remain the single most pedagogical scholar I attempt to simulate throughout my academic experiences and personal life. Furthermore, you have helped me to realize and maximize my full potential as a computer scientist. Finally, I would like to say thanks for challenging me to become a better person, programmer, and student.

To My Family: Thanks for believing in me and continually encouraging me to aspire to new heights academically, personally, and professionally.

To Chasity: This research process was truly made bearable and easier to handle due to your constant encouragement and abilities for lifting my spirits. Additionally, the times we have spent together will always be cherished, for you have helped me to accept, discover, and realize who I am in this dynamic world. Lastly, thanks for helping me to broaden my horizons, and become a more complete individual. The circle is almost complete and hopefully it will not be undone- lest time and efforts would be for naught.

There are not enough words to express my heartfelt thanks and appreciation for all that everyone has done on my behalf, so I will simply say- I love and respect you all from the depths of my inner being. Moreover, all of the persons acknowledged have assisted me in becoming a

better researcher and person in some capacity by helping me understand that tough times are not that tough, and that problems are really hidden opportunities.

CONTENTS

Page

ABSTRACT 2

ACKNOWLEDGEMENTS 3

LIST OF TABLES 9

LIST OF FIGURES 10

Chapter

1. INTRODUCTION 11

From OOP to AOP 12

Building Towards an AOP Taxonomy 14

Background 14

Overview 15

2. WHAT IS ASPECT-ORIENTED PROGRAMMING 16

Primary Concepts of Aspect-Oriented Programming 17

Concerns 17

Crosscutting Concerns 18

Separation of Concerns 18

Aspects and Components 19

Join Points and Weaving 20

Summary 21

3. TAXONOMY 23

The Taxonomy Process 23

Method 24

The AOP Categorization Model 27
Category Descriptions 28
Concerns 28
Adaptive 29
Limitations 29
Experiential and Case Study 29
Performance and Reliability 29
Distributed Systems 29
Theory 29
UML Modeling 29
Architecture 29
Software Engineering 29
Trends and Direction 30
Aspects 30
Weaving 30
Miscellaneous 30
Comparisons and Contrasts 30
Multidimensional View of the Taxonomy 34
Primary Dimension 35
Secondary Dimension 40
Tertiary Dimension 45
Quaternary Dimension 48
AOP Yearly Literature Trends 50
Conclusion 54
4. Conclusions and Future Work 55
REFERENCES 57

APPENDICES 61

Appendix A: Literature used in Taxonomy 62

Appendix B: Primary Dimension Articles and Categories 100

Appendix C: Secondary Dimension Articles and Categories 104

Appendix D: Tertiary Dimension Articles and Categories 107

Appendix E: Quaternary Dimension Articles and Categories 109

Appendix F: Some Recommended AOP Literature 110

VITA 115

LIST OF TABLES

Table	Page
1. Aop Database Tables Descriptions	26
2. Categories Table Schema	27
3. Taxonomy Table Schema	27
4. Literature Taxonomical Breakdowns	31

LIST OF FIGURES

FIGURE Page

1. Aop Database Schema 26
2. Primary Dimension Totals 35
3. Primary Dimension Breakdown (1A - 5D) 36
4. Primary Dimension Breakdown Continued (6A – 9C) 37
5. Primary Dimension Breakdown Continued (10A – 13D) 38
6. Primary Dimension Breakdown Continued (14A – 15G) 39
7. Secondary Dimension Totals 40
8. Secondary Dimension Breakdown (1A – 5D) 41
9. Secondary Dimension Breakdown Continued (6A – 9C) 42
10. Secondary Dimension Breakdown Continued (10A – 13D) 43
11. Secondary Dimension Breakdown Continued (14A - 15G) 44
12. Tertiary Dimension Totals 45
13. Tertiary Dimension Breakdown (1B – 11C) 46
14. Tertiary Dimension Breakdown Continued (12A – 15D) 47
15. Quaternary Dimension Totals 48
16. Quaternary Dimension Breakdown (1D – 15D) 49
17. Aop Year Breakdown 50
18. Aop Leading Category per Year Breakdown 51
18. Aop Leading Sub-category per Year Breakdown 53

CHAPTER 1

INTRODUCTION

Software artifacts are inherently complex; hence, there is no silver bullet for designing and implementing software systems [Brooks 95]. However, the intertwined nature of software design and programming paradigms has led to an evolutionary tenet in the computer science domain, Aspect-Oriented programming[1] (AOP), which potentially contradicts Brooks' assertion that there is no silver bullet for building software. That is, AOP has become an extension of Object-Oriented programming (OOP) by capitalizing on OOP's advantages while improving the disadvantages. AOP captures the structure of crosscutting concerns explicitly in a modular way and with linguistic tool support. AOP creates a better understanding of software through a high level of abstraction; prevents tangling of code, which allows for easier development and maintenance; and increases potential for reuse for both components and aspects.

Thus, the AOP paradigm constitutes the focus area for this research that includes a categorization of AOP literature, which was undertaken to identify and discuss the research areas of AOP. Consequently, a taxonomy was created to demonstrate what issues are being discussed, the direction previous research has taken, and where AOP research may go in the future.

The creation and development of programming paradigms, and subsequent programming languages, in conjunction with software development has become an area of considerable interest in computer science. Programmers have strived to formalize methods for constructing correct, efficient and easily modified programs; thus, languages evolved in order to support these new requirements [Highley 99]. Consequently, computer scientists realized that program organization and software component reusability were necessary given the growing complexity

[1] Read [Kiczales 97] for a perspective of Aspect-Oriented Programming

of software. Hence, object-orientation was created from a desire to have language constructs for modeling real world objects and inspiring software code organization and reuse [Capretz 03]. Continually evolving languages have enabled developers to create more complex programs for more advanced, faster machines. In addition, program design and maintenance have become key issues with the inception of software engineering and its principles [Parnas 01].

From OOP to AOP

Object oriented programming was expected to ignite the industrial revolution in software development [Cox 86], in that OOP addresses many concerns and issues. Furthermore, OOP facilitated the writing of complex applications, such as graphical user interfaces, operating systems, and distributed applications, while simultaneously maintaining understandable source code [Elrad 01b], and its use of modularity, encapsulation, and inheritance. Nonetheless, OOP still has several limitations. For example, OO technology has difficulty in appropriately separating concerns and applying domain-specific knowledge, crosscutting concerns [Elrad 01b]. That is, concerns cannot be easily contained into individual modules. Similarly, while object-orientation does promote software reuse, practical experience has demonstrated that it does not do this as efficiently as OO pioneers originally thought. This is because of code tangling and code spread over several units, making it more difficult for adequate software maintainability [Soares 02]. Code tangling destroys modularity and reduces software quality [Constantinides 00].

OOP was originally envisioned as a method for enhancing maintainability and extensibility by confining modification to affect as few classes/modules as possible, keeping systems highly structured and non-ambiguous. However, as programs continue to increase in size and complexity, it becomes rather difficult to cleanly separate concerns into well localized modules, leading to cross-cutting concerns and code tangling. Nonetheless, one must understand object-oriented concepts in order to recognize the supplementary role of AOP in reference to OOP as a software engineering methodology.

Post-object programming (POP) mechanisms that seek to enhance current OOP paradigms are currently an area of peak research interest in the computer science domain [Elrad 01b]. Aspect-oriented programming (AOP) is a POP technology receiving researcher's attention due to its capabilities of augmenting OOP and its expressiveness. Furthermore, AOP is a technique for achieving clear separation of various concerns at the design and source code levels, which is accomplished by separately specifying various system concerns or areas of interest, then weaving functionality at explicit points in source code to create an executable artifact. Thus, the goal for AOP is to build on OOP by supporting separation of those concerns that OOP handles poorly [Elrad 01a].

For these reasons, aspect-oriented programming evolved as an extension of the object-oriented paradigm to try and solve some of its shortcomings [Soares 02]. AOP maintains object orientation's major goals, such as abstraction, modularity, and code reuse, as well as attempts to avoid the problem of code tangling [Constantinides 00]. In addition, AOP offers a solution to creating modules with little or no crosscutting concerns by introducing aspects, a way of localizing concerns in a centralized area, therefore taking crosscutting concerns out of modules [Kiczales 97].

Building Towards an AOP Taxonomy

Background

The project's first phase began with an exhaustive search to find articles that used a taxonomy approach in research, regardless of domain, in hopes a model would be found that could serve as a guide for organizing the vast amount of material for the current project. Several articles using a taxonomy model were discovered; specifically, the LISA proceedings paper [Anderson 99], which was used as a general outline for this paper. LISA contained 342 papers categorized into 64 separate categories, including 9 major categories with several sub-headings each. However, none of the taxonomical papers, including LISA, reflected a categorical organization of aspect-oriented literature. Instead, LISA's primary focus was on tasks performed by system administrators, thus, included categorical headings reflecting that issue. Hence, the contribution and importance of the current taxonomy to AOP research is vital.

The second phase of the project began with Robert Filman's bibliography of aspect-oriented software development, which he created as an evolving document dedicated to aspect-oriented software development and programming [Filman 02]. The document created by Filman directed the initial gathering of AOP related literature. Subsequently, Internet searches were conducted to find articles that pertained to AOP, which included technical reports as well as literature submitted to conferences and journals. Additionally, categories were created to partition the collected research literature into concentrated areas for easier analysis. Finally, a database was created as a method for organizing and conveniently locating material collected for the taxonomy, which could become a viable research tool (see figure 1, and tables 1 - 3).

Overview

This paper can be broken down into four major parts. First, a brief introduction concerning object-oriented programming is made in order to explain and describe the relationship and transition between OOP and Aspect-oriented programming. Second, an overview of AOP is conducted to give a basic framework for the development of the technique and its meaning and importance. Third, the paper details the methodology for creating the taxonomy and addresses the categorizations created to organize the 494 articles in order to interpret the research trends throughout the chosen articles. Lastly, concluding remarks and suggestions for future research are presented.

CHAPTER 2

WHAT IS ASPECT ORIENTED PROGRAMMING?

Aspect-oriented programming (AOP) emerged as an experimental framework called D that resulted from Cristina Lopes' thesis work [Lopes 97] and earlier groundwork by Xerox PARC researchers into AOP, which she was involved in, while finishing her Ph.D. In her thesis, Lopes was working specifically with a problem domain that had synchronization and communication as crosscutting concerns that were modeled as aspects through two aspect languages, COOL and RIDL [Lopes]. Thus, AOP appeared as a response to the problem known from the generalized procedural languages. In these languages, program code pieces that execute a clearly separable aspect of a system, such as error handling and synchronization, are scattered and repeated throughout the overall program code and become tangled. AOP was designed to factor out such aspects into separate program units called by the same name, aspects [Dolog 01]. As programmers came up with more defined functions in programs developed with structured languages, it made sense to retain the modules, which combine functions and methods for later use so that the code would not have to be rewritten. This is what led to the birth of object-oriented languages and programming systems, and ultimately aspect-oriented programming [Ludy 02].

AOP is based on the idea that computer systems are better programmed by separately specifying the various concerns and properties or areas of interest of a system and some description of their relationships, and then relying on instruments in the AOP environment to weave or compose them together into a coherent program [Elrad 01b]. Simply stated, the primary goal of Aspect-oriented programming is to support the programmer in cleanly separating components and aspects from each other by providing instruments that make it possible to extract and compose them to render the overall system [Kiczales 97].

AOP does what object-oriented programming cannot do effectively, which is clearly and cleanly modularize functional system code, i.e. source code, by separating concerns into well localized units, called aspects, to eliminate code tangling. Objects do not seem to help as much in dealing with systemic concerns such as synchronization, resource sharing, distribution, memory management, and replication. These concerns decrease modularity because they typically cross-cut a system's class and module structure at the source code level. Thus, the complexity in existing systems seems to stem from the way in which the implementation of these concerns ends up being tangled and intertwined throughout the code [Lopes 99].

Primary Concepts of Aspect-Oriented Programming

In brief, aspect-oriented programming is a technique to design and address crosscutting concerns. The technique is intended to enable a more modular expression of design decisions, referred to as aspects, in the actual code. To better support the expression of crosscutting design decisions, AOP uses a component language to detail the basic functionality of the system, and aspect languages to describe the different crosscutting properties. An aspect weaver is then used to combine the components and the aspects into a system.

Concerns

A concern signifies a particular interest in some topic relating to a particular system of interest [Hilliard 99]. Concerns can range from high-level notions like security and quality of service to low-level notions such as caching and buffering. They can be functional, like features or business rules, or nonfunctional (systemic), such as synchronization and transaction management [Elrad 01b]. [Hilliard 99] has decided to express concerns in the form of questions: "How reliable is this system?"; "What function does the system perform?"; and "How is the system deployed?" [Aksit 01] distinguishes between two types of concerns: problem domain and solution domain concerns. Problem domain concerns represent concerns as they are defined from the client perspective. They specifically focus on the functionality of the system as the client expects it. Solution domain concerns represent the concerns as defined by the solution techniques.

Crosscutting Concerns. One problem to writing effective software, according to Kiczales and other researchers, lies at the heart of crosscutting concerns. That is, they are concerns which cannot be represented easily in modular form. These concerns disrupt the modularity that is desired in object-oriented programming. Moreover, the concerns introduce related or even duplicated code into one or more modules [Kiczales 97] that programmers are forced into writing whenever a crosscutting concern has to be executed [Elrad 01a]. In addition, according to Kiczales, this crosscutting phenomenon is directly responsible for code tangling [Kiczales].

Some examples of crosscutting concerns are performance, synchronization, communication, graphics manipulation, and debugging [Highley 99]. These concerns may be naturally non-separable, which are termed as crosscutting concerns, thus, referring to inevitable scattering of concerns to multiple abstractions [Aksit 01]. Any structural realization of a system will find that some concerns are neatly localized within a specific structural piece, while others cross multiple elements.

Separation of Concerns. The need for dealing with one important issue at a time was long ago named by Dijkstra as the principle of separation of concerns [Dijkstra 76]. Aspect-oriented programming is an approach founded on this concept of separation of concerns [Laddad 02]. Typically, the principle of separation of concerns has been used by software engineers to manage the complexity of software system development [Walker 99]. There are numerous benefits in expressing concerns of a software system in a well localized, single code section. For example, the code can be more easily understood, analyzed, modified, extended, debugged, and reused [Constantinides '00].

Aspects and Components

Aspects are the properties of a system that do not necessarily align with the system's functional components but tend to cut across functional components in the system, being spread throughout the code [Constantinides 00]. A property is an aspect if it can not be cleanly encapsulated in a generalized procedure. Aspects tend not to be units of the system's functional decomposition, but rather to be properties that affect the performance or semantics of the components in systemic ways. Aspects are further defined as system properties that crosscut components in system's implementation [Chavez 01]. That is, an aspect is a new abstraction taken from the existing source code in order to wrap concerns that are scattered all over the program, which allows for separating out concerns into a new well-localized module. Examples of aspects include memory access patterns and synchronization of concurrent objects [Kiczales 97].

A component is a modular unit of functional decomposition, which addresses a specific concern or function of the system. Components are properties of a system, for which the implementation can be cleanly encapsulated in a generalized procedure. Aspects are properties for which the implementation cannot be cleanly encapsulated in a generalized procedure.

Components tend to be units of the system's functional decomposition, such as image filters, bank accounts, and GUI widgets. Similarly, aspects and components crosscut each other in a system's implementation [Kiczales 97].

Join Points and Weaving

An aspect language allows for constructs, the aspects, to separate crosscutting features from existing programming modules, e.g. classes in object-oriented languages. It also facilitates specification of reference points, join points, which identify links between the code encapsulated by the aspects and the classes cross cut by this code [Rashid 02]. Because of the crosscutting, it is believed the intersection between components and aspects to be more important as components and aspects themselves [Bardou 98]. Join points are those elements of the component language semantics that the aspect programs coordinate with and are essential to the function of the aspect weaver [Kiczales 97].

A join point is a location that is affected by a crosscutting concern [Ossher 98], the site where two concerns crosscut one another [Stein 02], and the place where the weaver inserts aspect code. Join points can be present at either the statement level, which implies that the set of possible join points include every statement in the system, or the operation level, which implies that the possible set of join points includes every operation that the system performs [Ossher]. It is a major task for aspect-oriented programmers to specify a set of join points at which two concern models are inter-connected to each other. Thus, it is important for an aspect-oriented modeling language to provide suitable representations for join points [Stein].

Common to all aspect languages is that they cannot be processed by modern compilers for object-oriented languages. Until that status changes, aspects have to be integrated with classes before a compiler can take over to produce the executable program. The process of merging is called weaving, which is the process of combining different pieces of code (aspect code with core functionality code) into one executable module [Aldawud 02]; the tool required is called Aspect Weaver [Bollert 99]. Simply, an aspect weaver is a tool which merges the aspects and classes with respect to the join points; furthermore, the weaver must process the component and aspect languages, co-composing them properly to produce the desired total system operation [Kiczales 97]. The AOP weaver does not modify the source code of classes while weaving aspects; instead, inheritance is used to add aspect-specific code to classes [Bollert 99].

This merging or weaving can occur at two points in time: compile-time (static weaving) and run-time (dynamic weaving). During compile-time, the aspect weaver acts as a pre-processor, weaving the aspect definitions into the class definitions before compilation. In addition, the aspect weaver can act as a post-processor, weaving the aspect definitions into the compiled class code; thus, at run-time, the aspect weaver acts as a run-time interpreter or run-time generator [Rashid 02].

Summary

In conclusion, AOP, introduced in the academic/research community in 1997, has many advantages over the traditional Object-oriented programming technique. Specifically, it was introduced to solve crosscutting problems of OOP. It provides a better understanding of software due to a higher level of abstraction, which provides for functional decomposition of problems into smaller sub-problems until a point of granularity is achieved. AOP allows for simpler development and maintenance of source code since tangling of the code is primarily omitted. More importantly, AOP increases the potential for reuse of both components as well as aspects, including minimal coupling.

CHAPTER 3

TAXONOMY

The premise for organizing current aspect-oriented programming literature into a taxonomy was to catalog and categorize as many available papers associated with AOP as possible, and to assist future research on this emerging programming technique. The taxonomy will hopefully serve as a central database for researchers to begin future studies on areas concerning AOP where research is limited or necessary.

The Taxonomy Process

The process of collecting literature began with Robert Filman's bibliography and branched out into searching multiple avenues for data collection. The search for data related to AOP occurred in several online areas, such as the Association for Computing Machinery (ACM) website, the Institute of Electrical and Electronics Engineers (IEEE), its Computer Society branch website, in addition to internet indexing and search engines, e.g., citeseer, google, and vivisimo[2]. The task of finding AOP related literature in these different locations was accomplished using various search terms and phrases with the keywords "aspect-oriented programming" or "AOP" used as the querying string. All articles that were returned via the search mechanisms were downloaded and viewed to organize into the predefined categories.

The number of articles downloaded for the present research was 564, and approximately half can be directly attributed to Filman's bibliography of 592 citations. The number 564 was determined to be a representative sample of AOP, whereby, generalizations could be made about

[2] The location of the websites used are as follows: http://www.acm.org, http://citeseer.org, http://www.computer.org, http://www.google.com, and http://vivisimo.com

the present state of AOP research. However, after extensive cross-referencing among the literature, 70 papers were deemed ineligible to be used in the current study, due to the fact that they resulted in duplicate copies. Consequently, the final number of original documents yielded 494 (see Appendix A), which are used in the current research.

The Method

The groundwork for creating the predefined taxonomy for AOP based work relied heavily upon the ACM Computing Classification System, which has been adopted by IEEE, and the Curricula Computing model that was a joint venture between ACM and IEEE. Furthermore, the process of categorizing literature began with an assessment of current AOP related conferences and workshops to determine if conference themes, e.g. the workshop on Aspect-Oriented Modeling with UML in Aspect-Oriented Software Development (AOSD), could be used in classifications for the collected literature. The methodical approach of naming and defining the categories allowed for a database to be created serving as a data repository and as audit control.

Initially, the categories were entirely based on the ACM Computing Classification System, and the joint task force of ACM and IEEE's model entitled Curricula for Computing (CC); however, some of the literature could not be confined to the strict boundaries of the ACM's classification mechanism or the joint task force method of ACM and IEEE. The ACM Computing Classification System was designed to categorize concepts of computer science and to create a standard for classifying information in the technological realm. It includes many categories, which are also broken down several levels into various sub-categories; thus, the general model of the current paper was taken from the structure of this categorization framework, even though most of the categories themselves were virtually non-useful. IEEE's model for Curricula for Computing attempts to break down various concepts of computer science in the manner in which the concept should be taught in a classroom atmosphere. Some of the specific categories used in the model were also applied to the current paper, e.g., distributed systems, software design and implementation, and software requirements and specifications.

The primary presumption was that all the literature would immediately fall into one of the established categories; however, this was not always the case. That is, the ACM and IEEE predefined categories did not take into account areas specific to the AOP framework, e.g., weaving, nor the need to add or take away sub-categories to better express AOP. For purposes of the current paper, modern computing concepts and terminology were used to compose additional categories that express the uniqueness of the literature used in the taxonomy.

A relational database was created to store the article categorizations for audit and tracking purposes, and to serve as a central repository for the data. The database consists of two tables with several expressive columns. The categories table includes the categories name, sub-category, and sub-category id. In addition, the taxonomy table stores an articles id, citation, year, primary category, and/or linked category information as a means of cross-referencing articles to eliminate confusion and duplicity, and to illustrate multi-dimensional perspectives for the taxonomy (see figure 1, and tables 1 – 3).

Figure 1: AOP Database Schema

Categories	
PK	**Sub Category ID**
	Sub Categories Category Name

Taxonomy	
PK	**ID**
	Citation **Year** **Primary** Secondary Tertiary Quaternary

Figure 1, the database schema is a visual aid consisting of two tables, categories and taxonomy. The primary key for the categories table is “Sub Category ID”, and the taxonomy tables’ primary key is “ID”. In addition, the taxonomy table has a foreign key, “Primary” connected to the categories table primary key that defines the foreign keys’ relationship between the two tables. The tables both have required columns that can not have null values. For example, “Sub Category ID” and “Sub Categories must have information, while “Category Name” does not necessarily need a value. Similarly, the “ID”, “Citation”, “Year”, and “Primary” columns in the taxonomy table can not contain null values, but columns “Secondary”, “Tertiary”, and “Quaternary” may have null data.

Table 1: AOP Database Tables Descriptions

Table Name	**The type of data described in the table**
Categories	General description of category information
Taxonomy	Bibliography data and categorical references

Table 1 gives a general description of the tables in the database along with their names. The categories table stores information about the organization of the research literature, and the taxonomy table has the bibliographic and multi-dimensional views data.

Table 2: Categories Table Schema

Column name	Column description	Required value
	Table Name: Categories	
Sub Category ID	The sub-categories ID, range 1a – 15g	Yes
Sub Categories	The sub-categories name	Yes
Category Name	Name of the main categories	No

Table 2 shows the column names and gives a brief description of their functions. In addition, it illustrates if a value is required or not for each column.

Table 3: Taxonomy Table Schema

Column name	Column description	Required Field
	Table Name: Taxonomy	
ID	The article auto-number	Yes
Citation	The article cite	Yes
Year	The year article available	Yes
Primary	The one dimensional perspective, main category	Yes
Secondary	The second dimensional perspective	No
Tertiary	The third dimensional perspective	No
Quaternary	The fourth dimensional perspective	No

Table 3 represents the schema for the taxonomy table. It describes the column names, and gives an idea about their purpose. The table also shows if a column must have a value or not, for example, “Year” can not be null, thus requiring some data.

The AOP Categorization Model

A rudimentary approach for constructing taxonomies is to group related entities by the task each targets. This was done in the current project for all the papers. In order to achieve a list of categories that would encapsulate the various AOP topics, the literature was gathered and the focus of the articles examined, which provided a running list of categories to be included in the taxonomy. Fifteen categories for Aspect-oriented programming were created and defined as a result of examining the current literature trends. The categories are representative of some current research interests in respect to the AOP paradigm but are by no means inclusive of all research conducted on AOP. Some of the literature fell into multiple categories; however, all of the articles were stored into one main category that expressed the primary literatures focus of that specific article. In addition, for the purpose of organizing those articles which contained information relating to categories other than the primary, a second (secondary), third (tertiary), and fourth (quaternary) category were devised to denote the additional dimension of the categorization model.

Category Descriptions

The categories and their descriptions were derived primarily from the current research trends of Aspect-oriented programming as well as from the websites of the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers Computer Society branch. However, the categories were not confined by strict definitions per se; but instead, they were compiled to capture variance among the literature within the context of a designated category. Additionally, the definitions serve as a template, which can be refined to be more inclusive or exclusive depending on desired category expressiveness. Each category and its respective explanation are briefly discussed below.

Concerns. A concern is a domain that defines the manner in which the original problem should be decomposed. Specifically, a concern is a particular goal, concept, or area of interest. Concerns range from business and performance issues to debugging, authentication, and security. This category also includes information about crosscutting concerns, identifying concerns, modeling concerns, separation of concerns, and use of actors. See [Orleans '01]

Adaptive. Adaptive is the automatic adjusting to changing hardware and software environments. See [Constantinides '01]

Limitations. The limitations category includes the body of literature that focuses on constraints and weaknesses of the AOP paradigm. See [Fabry '01]

Experiential and Case Study. Some of the AOP research deals with various techniques of learning better ways to implement AOP. This also includes literature which discusses various situations where AOP has been tested and evaluated in order to examine the final outcome. See [Avdicausevic '01]

Performance and Reliability. The literature in this category aims to curtail possible system degradation and maintenance by optimizing software and hardware elements to be more robust. This category encompasses information about synchronization, error detection and error handling, reusability, memory management, and security. See [Holmes '97]

Distributed Systems. This category includes the mechanisms and methods network computers use to communicate by passing messages among system components. See [Putrycz '02]

Theory. The literature in this category includes formal methods, theoretical foundations, and pragmatic approaches to espouse the AOP paradigm. See [Achermann '00]

UML Modeling. This category is comprised of literature that uses UML concepts and design notation to illustrate system design. See [Pawlak '02]

Architecture. The architecture category discusses the use of AOP in designing and possible implementation of hardware and software elements. See [Navasa '01]

Software Engineering. This category consists of literature that relates to the process of engineering software, including all phases of the development lifecycle. Additionally, tools and languages created using AOP methodologies help to comprise this category. The category includes information regarding requirements engineering, process modeling, and software design, tools and testing. See [Nordberg '01]

Trends and Direction. Literature in this category includes research that describes the framework and future of the AOP paradigm from its origination through its present course. See [Brichau '02]

Aspects. The aspects category incorporates literature that focuses on identifying aspects and understanding their relationship to join points and ultimately the weaving process. See [Coady '03]

Weaving. This category is composed of research that focuses on the integration of aspects and concerns and why this is a necessary process when implementing the AOP technique. See [Akkawi '01]

Miscellaneous. This category was created to place literature that does not adequately fall into any one of the pre-defined categories. See [Van Roy '97]

Comparisons and Contrasts. AOP comparisons category incorporates literature that compares and contrasts other programming techniques and paradigms to AOP. See [Ortin '02]

The following table includes the breakdown of the main categories and sub-categories used to organize the 494 papers (see Appendix A) included in the taxonomy. In addition, those categories with two asterisks indicate the largest sub-category of each particular section.

Table 4: Literature Taxonomical Breakdowns

1. Concerns [25]
    - 1.a. General [1]
    - 1.b. Modeling Concerns [3]
    - 1.c. Identifying Concerns [3]
    - 1.d. Separating Concerns [13] **
    - 1.e. Cross-cutting Concerns [5]

2. Adaptive [13]
    - 2.a. General [0]
    - 2.b. Models & Techniques [11] **
    - 2.c. Tools & Applications [2]

3. Limitations [7]
    - 3.a. General [0]
    - 3.b. Theoretical [1]
    - 3.c. Tools [1]
    - 3.d. Design & Environment [5] **

4. Experiential & Case Study [41]
    - 4.a. General [8]
    - 4.b. Tools & Languages [14] **
    - 4.c. Design Aids & Analysis [9]
    - 4.d. Testing, Fault Tolerance & Reliability [1]
    - 4.e. Simulation & Modeling [9]

5. Performance & Reliability [16]
    - 5.a. General [2]
    - 5.b. Synchronization [6] **
    - 5.c. Reusability [3]
    - 5.d. Security [5]

6. Distributed Systems [28]
    - 6.a. General [4]
    - 6.b. Networks & Environments [10] **
    - 6.c. Client-Server (Web) [4]
    - 6.d. Applications [6]
    - 6.e. Programming [2]
    - 6.f. Real-Time [2]

7. Theory [66]
    - 7.a. General [1]
    - 7.b. Formal Methods [15]
    - 7.c. Applied [34] **
    - 7.d. Theoretical [16]

8. UML Modeling [33]
    - 8.a. General [6]
    - 8.b. Tools [2]
    - 8.c. Extensions [11] **
    - 8.d. Diagrams, Design Aids & Simulation [11] **
    - 8.e. Verification & Analysis [3]

9. Architecture [23]
    - 9.a. General [6]
    - 9.b. Distributed & Network [1]
    - 9.c. Styles & Models [16] **

10. Software Engineering [62]
    - 10.a. General [0]
    - 10.b. Requirements & Specification [8]
    - 10.c. Processes & Metrics [10]
    - 10.d. Design & Implementation [22] **
    - 10.e. Testing & Debugging [4]
    - 10.f. Design Tools & Languages [18]

11. Trends & Direction [26]
    - 11.a. General [5]
    - 11.b. Analysis [7]
    - 11.c. Future Domains [9] **
    - 11.d. Designs & Frameworks [4]
    - 11.e. Language Constructs [1]

12. Aspects [18]
    - 12.a. General [4] **
    - 12.b. Methods & Tools [4] **
    - 12.c. Analysis [3]
    - 12.d. Fundamentals [1]
    - 12.e. Models [4] **
    - 12.f. Implementations [2]

13. Weaving [15]
    - 13.a. General [5] **
    - 13.b. Join Points [2]
    - 13.c. Tools [3]
    - 13.d. Implementation, Techniques & Models [5] **

14. Miscellaneous [91]
    - 14.a. General [0]
    - 14.b. Tools [0]
    - 14.c. Concerns [18]
    - 14.d. Programming Techniques [10]
    - 14.e. Distributed Systems [9]
    - 14.f. Adaptive [1]
    - 14.g. Software Engineering [22] **
    - 14.h. Case Study & Experiential [5]
    - 14.i. Architecture [1]
    - 14.j. Performance & Reliability [1]
    - 14.k. Theory [15]
    - 14.l. Aspects [0]
    - 14.m. UML [7]
    - 14.n. Weaving [2]

15. Comparisons & Contrasts [30]
    - 15.a. General [0]
    - 15.b. Programming Techniques [16] **
    - 15.c. Modeling Concerns [4]
    - 15.d. Separating Concerns [7]
    - 15.e. Identifying Concerns [1]
    - 15.f. Distributed Applications [1]
    - 15.g. Cross-cutting Concerns [1]

The largest single category of literature collected for this taxonomy is the Miscellaneous category (see Figure 2), which comprised fourteen sub-categories, and included a total of 91 articles. Theory, with four sub-categories, came in second with a total of 66 papers collected. Software Engineering, including six sub-categories, came in third with a total of 62 research articles. The applied sub-category under theory occupies the top position with a total of 34 articles. The design and implementation sub-category under software engineering, and the software engineering sub-category under miscellaneous both occupy the 2 position with a total of

22 research articles. Next, the design tools and languages sub-category under software engineering, and the concerns sub-category under miscellaneous were the third most popular sub-categories with 18 research articles.

Multidimensional View of the Taxonomy

The taxonomy includes a four dimensional view: primary, secondary, tertiary, and quaternary. The multi-dimensional view offers researchers another piece of information by classifying articles into all relevant categories, hence, demonstrating the literatures applicability to multiple research domains. The construction of a multi-dimensional view was accomplished by ranking (1-4) the articles to correspond with the appropriate dimension. This was done because the articles are not narrowly written; instead, they tend to bring in multiple ideas and discuss them at great lengths. Essentially, many articles belong in more than one category, i.e., [Giese ‘00]. In other words, the multidimensional view simply depicts the articles’ ability to fall into more than one category with the primary view being the dominant concept of the article. The following figures illustrate the four-dimensional view of the taxonomy by breaking down the categories according to the number of papers in each category for the primary, secondary, tertiary, and quaternary dimension.

Primary Dimension

Figure 2: Primary Dimension Totals

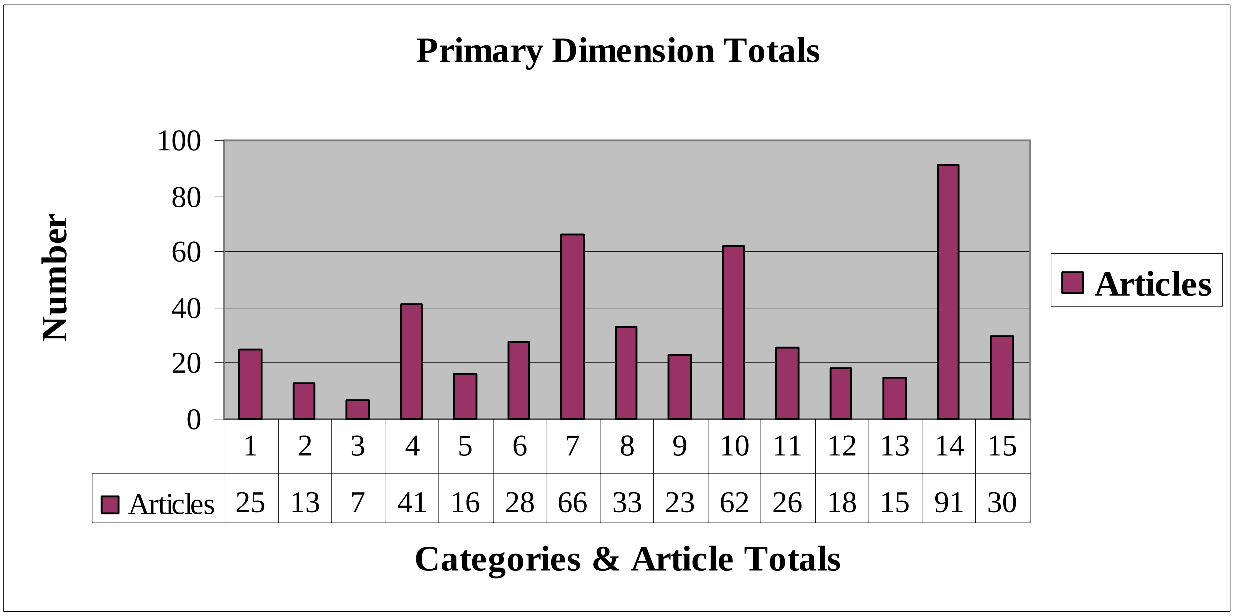


Figure 2 describes the layout of total articles for each of the main fifteen categories in the primary (main) dimension. The primary dimension includes the largest number of articles because each article was categorized into at least one category. The other three dimensions do not necessarily include all of the articles, only the ones which fell into more than one category. The primary categories are numbered 1-15 and include Concerns; Adaptive; Limitations; Experiential/Case Study; Performance/Reliability; Distributed Systems; Theory; UML Modeling; Architecture; Software Engineering; Trends/Direction; Aspects; Weaving; Miscellaneous; and Comparisons/Contrasts. Of the 494 articles indexed in the taxonomy, Category 3, (Limitations), has the least number of articles totaling 7. The primary dimension categorical breakdown into sub-categories and the number of articles per sub-category follows (see figures 3-6).

Figure 3: Primary Dimension Breakdown (1A – 5D)

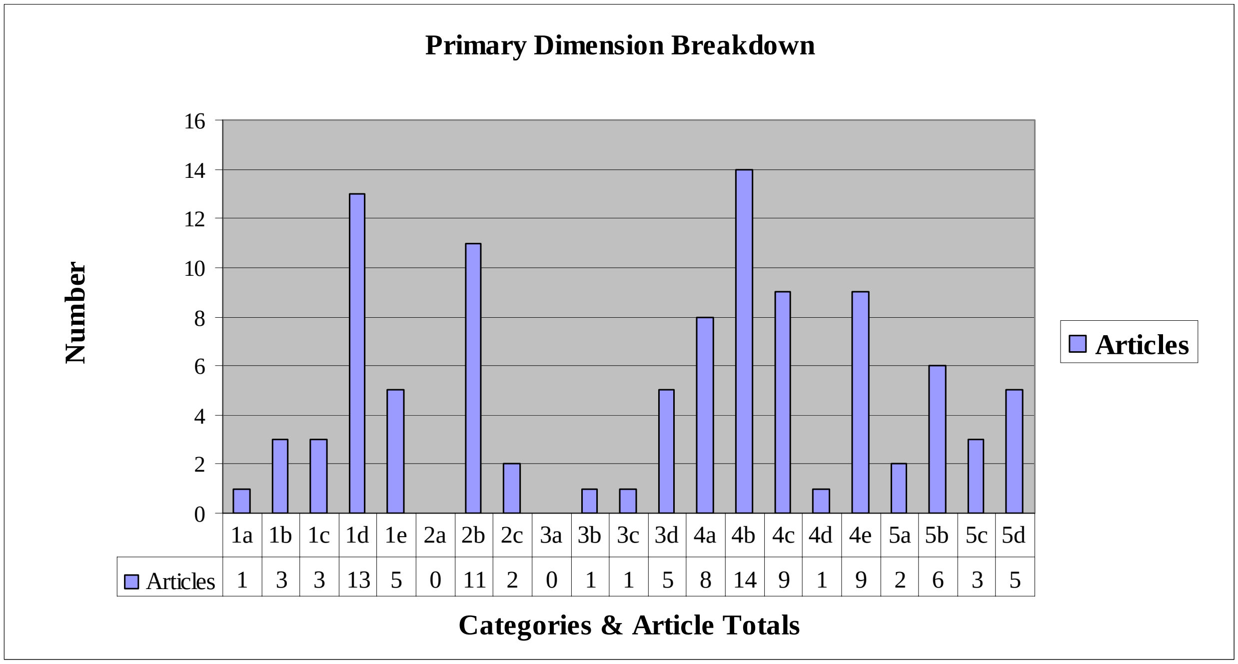


Figure 3 represents the breakdown of the primary dimension of sub-categories 1a – 5d. The sub-categories are labeled following their respective main category heading: 1. Concerns: 1a. General Concerns, 1b. Modeling Concerns, 1c. Identifying Concerns, 1d. Separating Concerns, 1e. Cross-cutting Concerns; 2. Adaptive: 2a. General, 2b. Models & Techniques, 2c. Tools & Applications; 3. Limitations: 3a. General, 3b. Theoretical, 3c. Tools, 3d. Design & Environment; 4. Experiential & Case Study: 4a. General, 4b. Tools & Languages, 4c. Design Aids & Analysis, 4d. Testing, Fault Tolerance & Reliability, 4e. Simulation & Modeling; and 5. Performance & Reliability: 5a. General, 5b. Synchronization, 5c. Reusability, 5d. Security.

The sub-category with the most articles in figure 3 is 4b with 14 and in a close second is 1d with 13. The two sub-categories with the least amount of articles are 2a and 3a with a total of 0 articles.

<u>Figure 4: Primary Dimension Breakdown Continued (6A – 9C)</u>

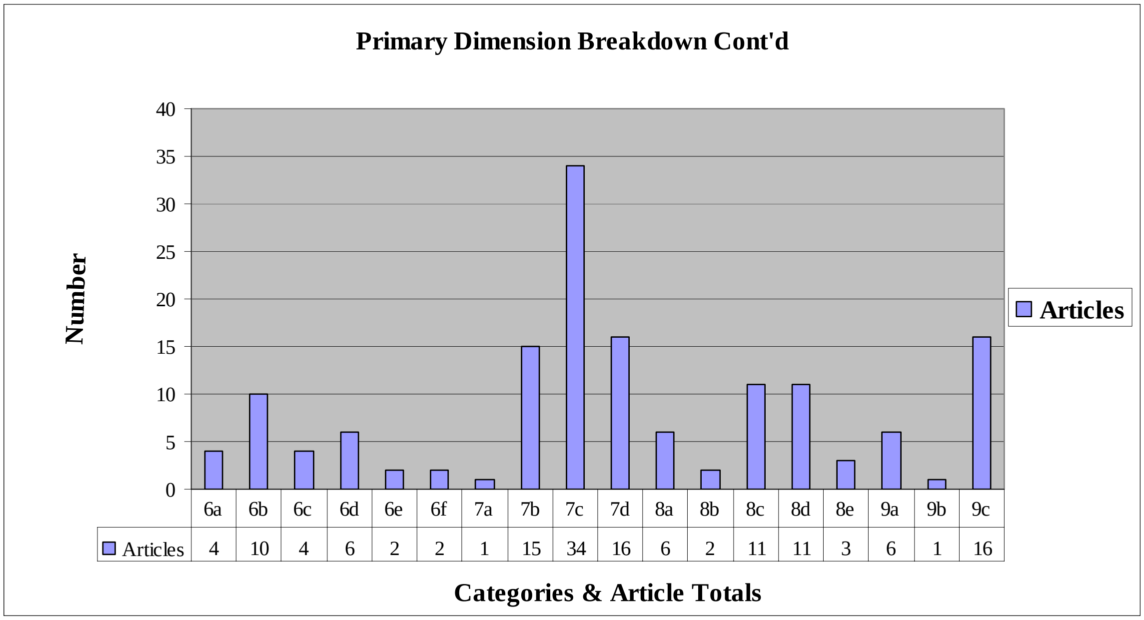


Figure 4 represents the breakdown of the primary dimension of sub-categories 6a - 9c. The sub-categories are labeled following their respective main category heading: 6. Distributed Systems: 6a. General, 6b. Networks & Environments, 6c. Client-Server (Web), 6d. Applications, 6e. Programming, 6f. Real-Time; 7. Theory: 7a. General, 7b. Formal Methods, 7c. Applied, 7d. Theoretical; 8. UML Modeling: 8a. General, 8b. Tools, 8c. Extensions, 8d. Diagrams, Design Aids & Simulation, 8e. Verification & Analysis; and 9. Architecture: 9a. General, 9b. Distributed & Network, 9c. Styles & Models.

The sub-category with the most articles in figure 4 is 7c with 34 and tying for second are categories 7d and 9c with a total of 16 articles. The sub-categories with the least amount of articles are 7a and 9b with 1 article each.

<u>Figure 5: Primary Dimension Breakdown Continued (10A – 13D)</u>

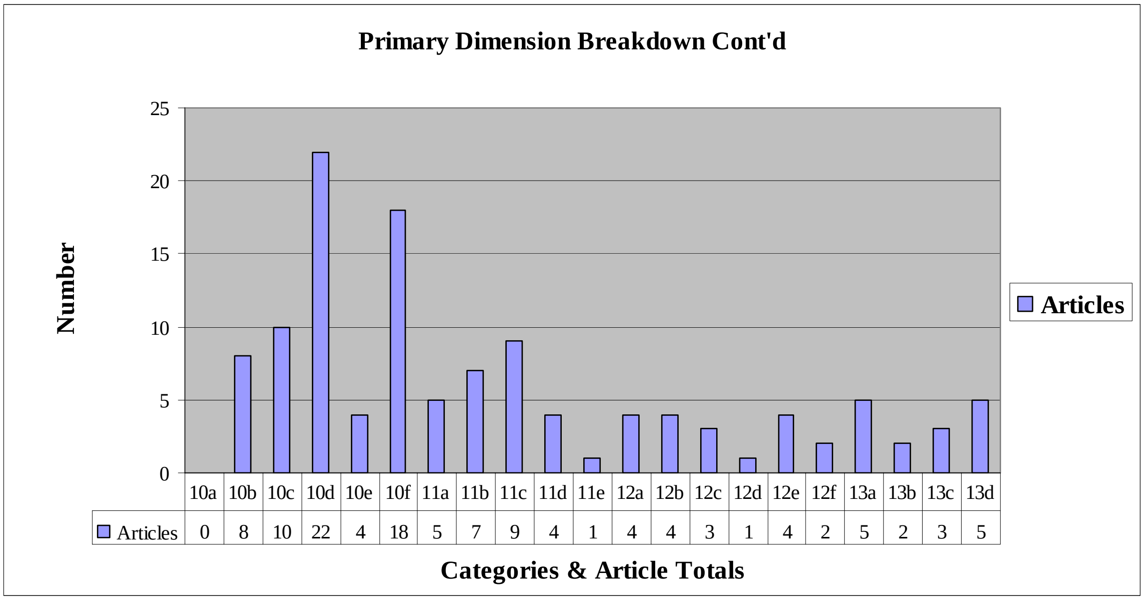
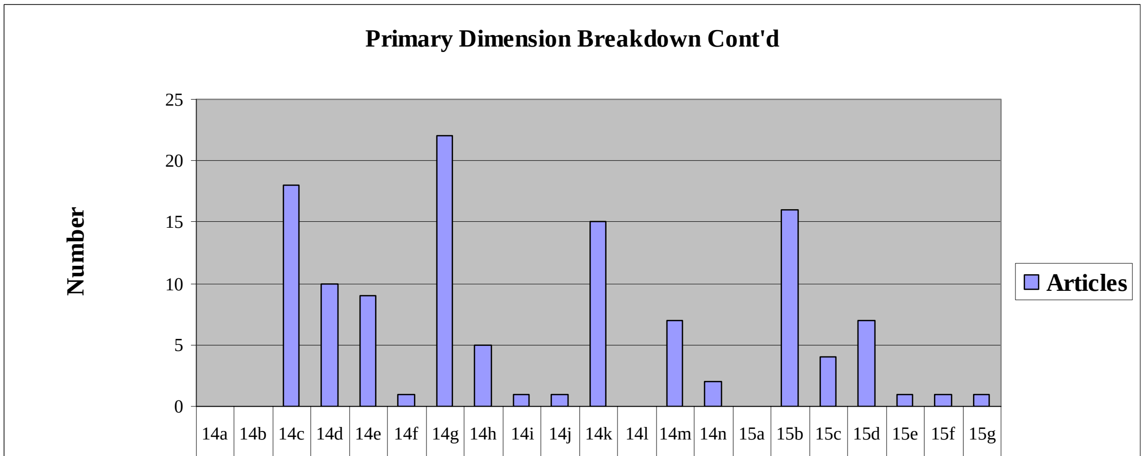


Figure 5 represents the breakdown of the primary dimension of sub-categories 10a – 13d. The sub-categories are labeled following their respective main category heading: 10. Software Engineering: 10a. General, 10b. Requirements & Specification, 10c. Processes & Metrics, 10d. Design & Implementation, 10e. Testing & Debugging, 10f. Design Tools & Languages; 11. Trends & Direction: 11a. General, 11b. Analysis, 11c. Future Domains, 11d. Designs & Frameworks, 11e. Language Constructs; 12. Aspects: 12a. General, 12b. Methods & Tools, 12c. Analysis, 12d. Fundamentals, 12e. Models, 12f. Implementations; and 13. Weaving: 13a. General, 13b. Join Points, 13c. Tools, 13d. Implementation, Techniques & Models.

The sub-category with the most articles in figure 5 is 10d with 22. Sub-category 10f falls in second place with a total 18 articles. The sub-category with the least amount of articles is 10a with 0.

Figure 6: Primary Dimension Breakdown Continued (14A- 15G)



	14a	14b	14c	14d	14e	14f	14g	14h	14i	14j	14k	14l	14m	14n	15a	15b	15c	15d	15e	15f	15g
Articles	0	0	18	10	9	1	22	5	1	1	15	0	7	2	0	16	4	7	1	1	1

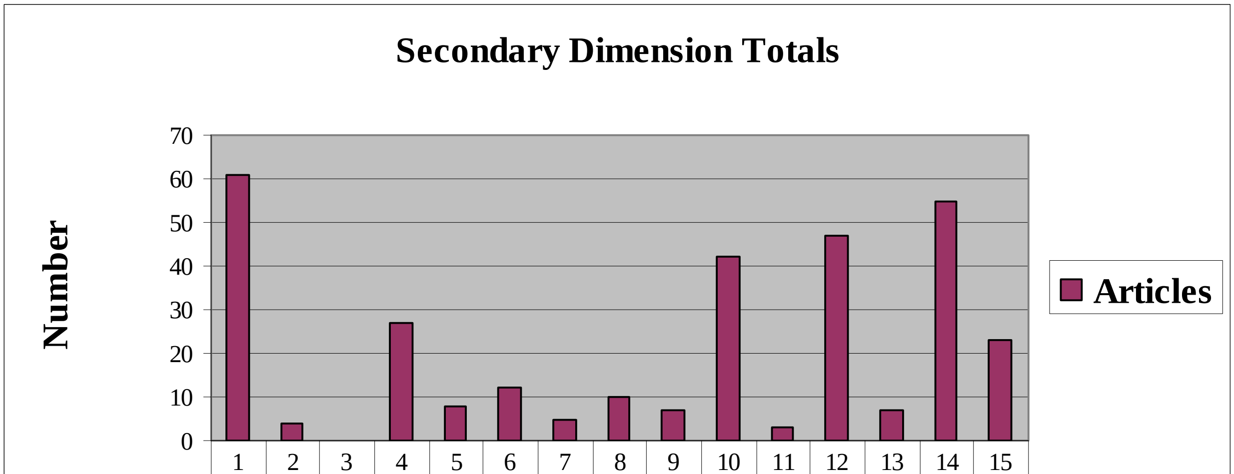
Figure 6 represents the breakdown of the primary dimension of sub-categories 14a – 15g. The sub-categories are labeled following their respective main category heading: 14. Miscellaneous: 14a. General, 14b. Tools, 14c. Concerns, 14d. Programming Techniques, 14e. Distributed Systems, 14f. Adaptive, 14g. Software Engineering, 14h. Case Study & Experiential, 14i. Architecture, 14j. Performance & Reliability, 14k. Theory, 14l. Aspects, 14m. UML, 14n. Weaving; and 15. Comparisons & Contrasts: 15a. General, 15b. Programming Techniques, 15c. Modeling Concerns, 15d. Separating Concerns, 15e. Identifying Concerns, 15f. Distributed Applications, 15g. Cross-cutting Concerns.

The sub-category with the most articles in figure 6 is 14g with 22. Sub-category 14c falls in second place with a total 18 articles. The sub-categories with the least amount of articles are 14a, 14b, 14l, and 15a with 0.

In the primary dimension, sub-category 7c, (Applied Theory) has the largest number of articles, 34. Sub-categories 10d, (Design/Implementation), and 14g, (Software Engineering), both with 22 follow with the second largest amount of articles in the primary dimension. Lastly, the primary dimension consists of 494 articles.

Secondary Dimension

Figure 7: Secondary Dimension Totals



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Articles	61	4	0	27	8	12	5	10	7	42	3	47	7	55	23

Figure 7 describes the layout of total articles for each of the main fifteen categories in the secondary dimension. The secondary categories are numbered 1-15 and include Concerns; Adaptive; Limitations; Experiential/Case Study; Performance/Reliability; Distributed Systems; Theory; UML Modeling; Architecture; Software Engineering; Trends/Direction; Aspects; Weaving; Miscellaneous; and Comparisons/Contrasts. Figure 7 shows the number of articles and the category they belong to in the secondary dimension of the model. Category 1 had the largest number of articles with a total of 61 and category 3 had the least with a total of 0. Figures 8 -11 show the sub-categorical breakdown of the secondary dimension.

Figure 8: Secondary Dimension Breakdown (1A – 5D)

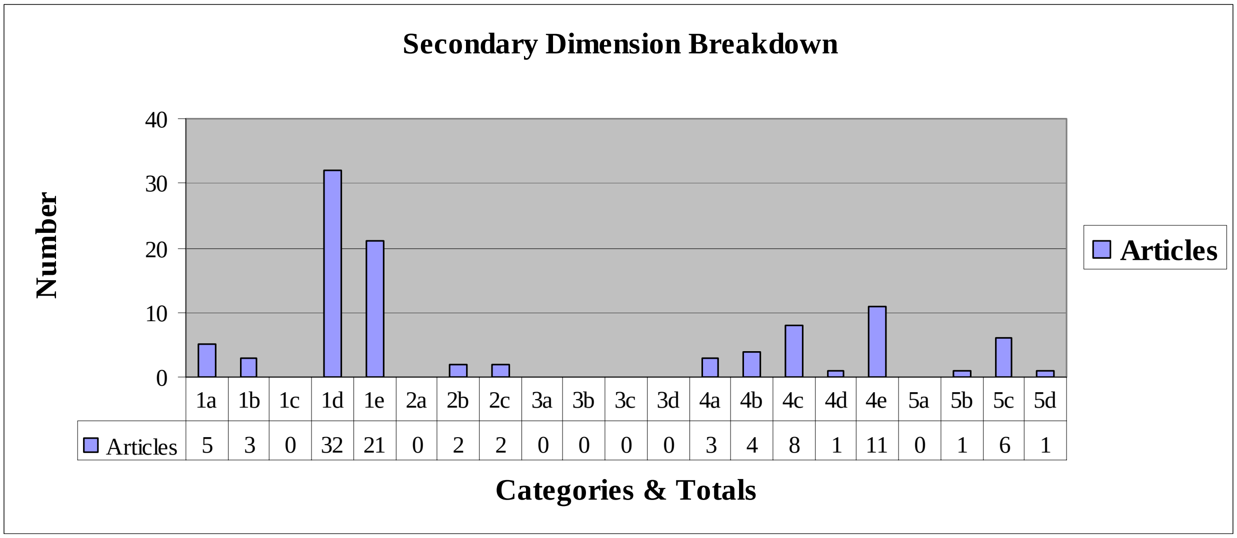


Figure 8 illustrates the second dimension sub-categories 1a - 5d. The sub-categories are labeled following their respective main category heading: 1. Concerns: 1a. General Concerns, 1b. Modeling Concerns, 1c. Identifying Concerns, 1d. Separating Concerns, 1e. Cross-cutting Concerns; 2. Adaptive: 2a. General, 2b. Models & Techniques, 2c. Tools & Applications; 3. Limitations: 3a. General, 3b. Theoretical, 3c. Tools, 3d. Design & Environment; 4. Experiential & Case Study: 4a. General, 4b. Tools & Languages, 4c. Design Aids & Analysis, 4d. Testing, Fault Tolerance & Reliability, 4e. Simulation & Modeling; and 5. Performance & Reliability: 5a. General, 5b. Synchronization, 5c. Reusability, 5d. Security.

The sub-categories with the most articles in figure 8 is 1d with a total of 32 articles and in second place is sub-category 1e with 21 articles. The sub-categories with the least amount of articles are 1c, 2a, 3a, 3b, 3c, 3d, and 5a with 0.

<u>Figure 9: Secondary Dimension Breakdown Continued (6A – 9C)</u>

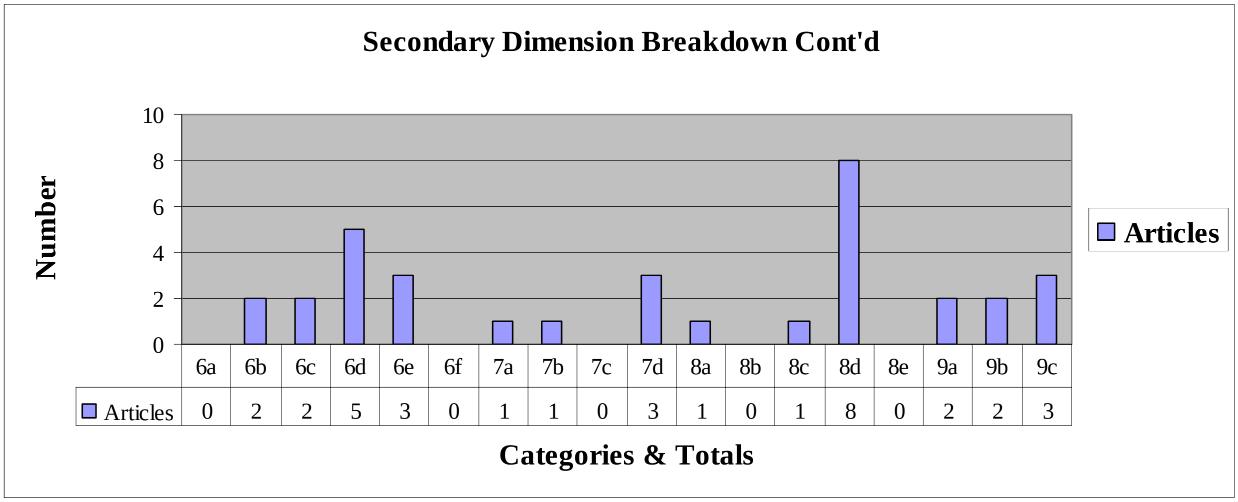


Figure 9 illustrates sub-categories 6a - 9c in the secondary dimension. The sub-categories are labeled following their respective main category heading: 6. Distributed Systems: 6a. General, 6b. Networks & Environments, 6c. Client-Server (Web), 6d. Applications, 6e. Programming, 6f. Real-Time; 7. Theory: 7a. General, 7b. Formal Methods, 7c. Applied, 7d. Theoretical; 8. UML Modeling: 8a. General, 8b. Tools, 8c. Extensions, 8d. Diagrams, Design Aids & Simulation, 8e. Verification & Analysis; and 9. Architecture: 9a. General, 9b. Distributed & Network, 9c. Styles & Models.

The sub-category with the most articles in figure 9 is 8d with 8 articles and coming in second is sub-category 6d with 5 articles. The sub-categories with the least amount of articles are 6a, 6f, 7c, 8b, and 8e with 0.

Figure 10: Secondary Dimension Breakdown Continued (10A – 13D)

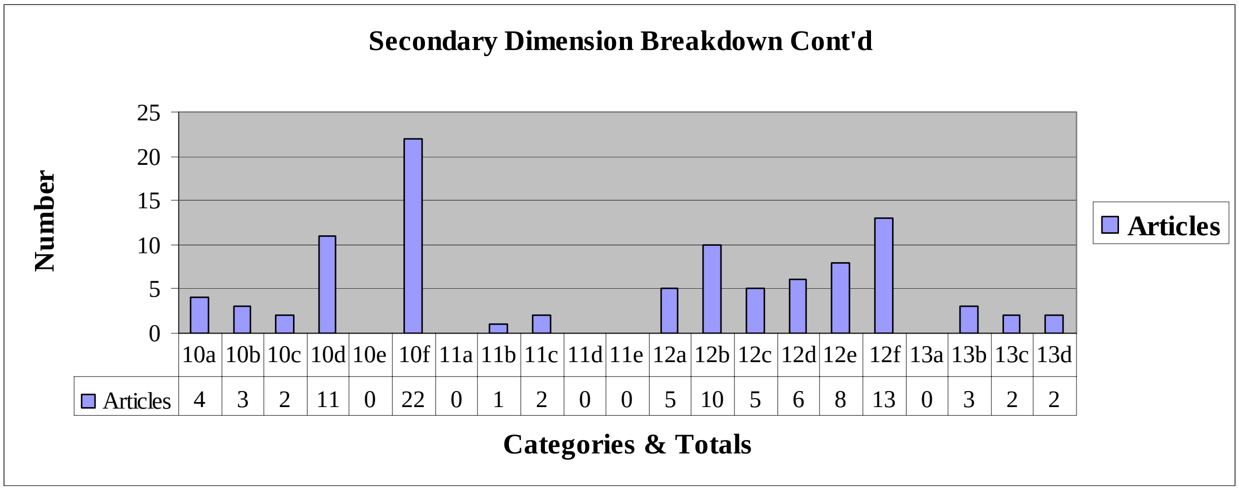


Figure 10 illustrates sub-categories 10a - 13d in the secondary dimension. The sub-categories are labeled following their respective main category heading: 10. Software Engineering: 10a. General, 10b. Requirements & Specification, 10c. Processes & Metrics, 10d. Design & Implementation, 10e. Testing & Debugging, 10f. Design Tools & Languages; 11. Trends & Direction: 11a. General, 11b. Analysis, 11c. Future Domains, 11d. Designs & Frameworks, 11e. Language Constructs; 12. Aspects: 12a. General, 12b. Methods & Tools, 12c. Analysis, 12d. Fundamentals, 12e. Models, 12f. Implementations; and 13. Weaving: 13a. General, 13b. Join Points, 13c. Tools, 13d. Implementation, Techniques & Models.

The sub-category with the most articles in figure 10 is 10f with 22 and coming in second is sub-category 12f with 13 articles. The sub-categories with the least amount of articles are 10e, 11a, 11d, 11e, and 13a with 0.

<u>Figure 11: Secondary Dimension Breakdown Continued (14A – 15G)</u>

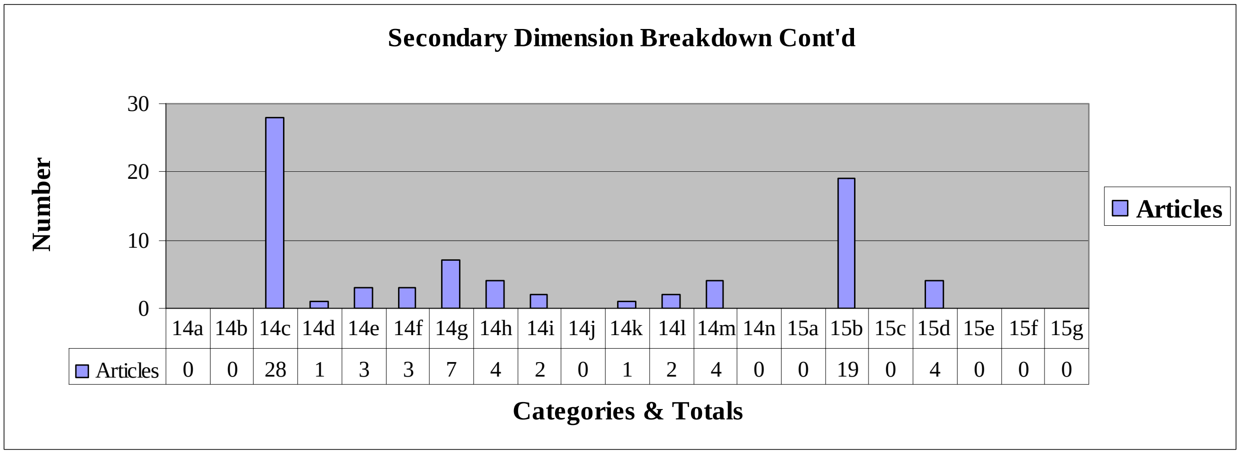


Figure 11 shows the sub-category breakdowns of 14a – 15g in the secondary dimension. The sub-categories are labeled following their respective main category heading: 14. Miscellaneous: 14a. General, 14b. Tools, 14c. Concerns, 14d. Programming Techniques, 14e. Distributed Systems, 14f. Adaptive, 14g. Software Engineering, 14h. Case Study & Experiential, 14i. Architecture, 14j. Performance & Reliability, 14k. Theory, 14l. Aspects, 14m. UML, 14n. Weaving; and 15. Comparisons & Contrasts: 15a. General, 15b. Programming Techniques, 15c. Modeling Concerns, 15d. Separating Concerns, 15e. Identifying Concerns, 15f. Distributed Applications, 15g. Cross-cutting Concerns.

The sub-category with the most articles in figure 11 is 14c with 28 and in second is sub-category 15b with 19 articles. The sub-categories with the least amount of articles are 14a, 14b, 14j, 14n, 15a, 15c, 15e, 15f and 15g with 0.

In the secondary dimension, sub-category 1d, (Separating Concerns) has the largest number of articles with a total of 32. The sub-category 14c (Concerns) has the second largest amount of articles in the secondary dimension with a total 28. Lastly, the secondary dimension consists of a total of 311 articles.

Tertiary Dimension

Figure 12: Tertiary Dimension Totals

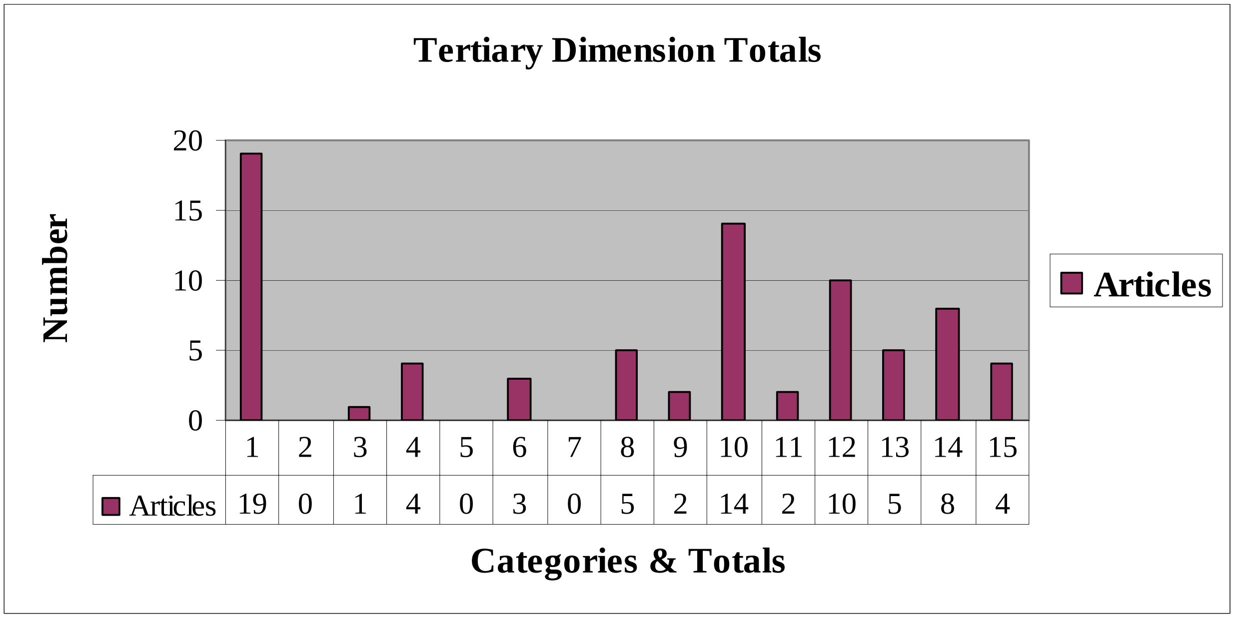
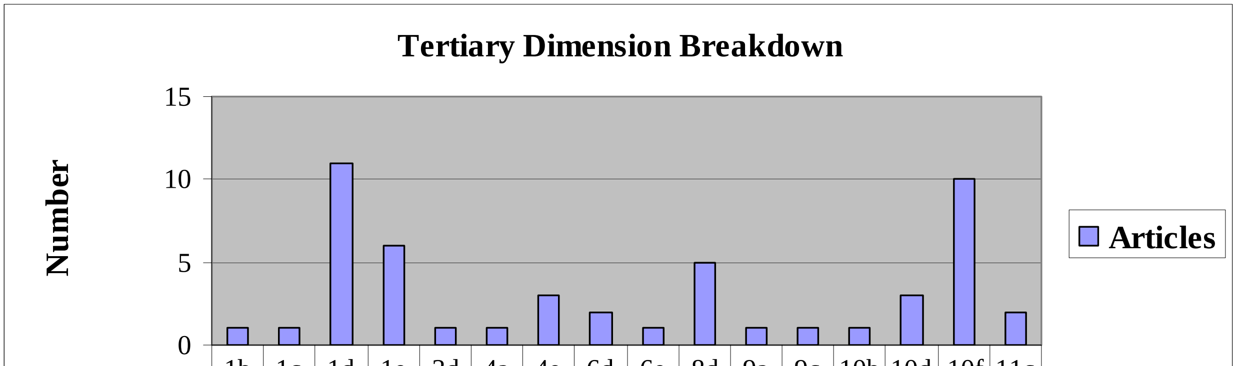


Figure 12 shows the number of articles and the category they belong to in the tertiary dimension of the model. Categories without articles in them do not have any tertiary relationships. The tertiary categories are numbered 1-15 and include Concerns; Adaptive; Limitations; Experiential/Case Study; Performance/Reliability; Distributed Systems; Theory; UML Modeling; Architecture; Software Engineering; Trends/Direction; Aspects; Weaving; Miscellaneous; and Comparisons/Contrasts. Category 1 had the highest number of articles with a total of 19 and categories 2, 5, and 7 had the least with a total of 0. The following figures 13 and 14 show the sub-categorical breakdown of the tertiary dimension.

Figure 13: Tertiary Dimension Breakdown (1B – 11C)



	1b	1c	1d	1e	3d	4a	4e	6d	6e	8d	9a	9c	10b	10d	10f	11c
Articles	1	1	11	6	1	1	3	2	1	5	1	1	1	3	10	2

Figure 13 demonstrates the breakdown of the tertiary relationship of the articles with sub-categories 1b – 11c. The sub-categories are labeled following their respective main category heading: 1. Concerns: 1b. Modeling Concerns, 1c. Identifying Concerns, 1d. Separating Concerns, 1e. Cross-cutting Concerns; and 3. Limitations: 3d. Design & Environment; and 4. Experiential & Case Study: 4a. General, 4e. Simulation & Modeling; and 6. Distributed Systems: 6d. Applications, 6e. Programming; and 8. UML Modeling: 8d. Diagrams, Design Aids & Simulation; and 9. Architecture: 9a. General, 9c. Styles & Models, and 10. Software Engineering: 10b. Requirements & Specification, 10d. Design & Implementation, 10f. Design Tools & Languages; and 11. Trends & Direction: 11c. Future Domains.

The sub-category with the most articles in figure 13 is 1d with 11 and in a close second is sub-category 10f with 10. The sub-categories with the least amount of articles are 1b, 1c, 3d, 4a, 6e, 9a, 9c, and 10b with 1 article each.

Figure 14: Tertiary Breakdown Continued (12A – 15D)

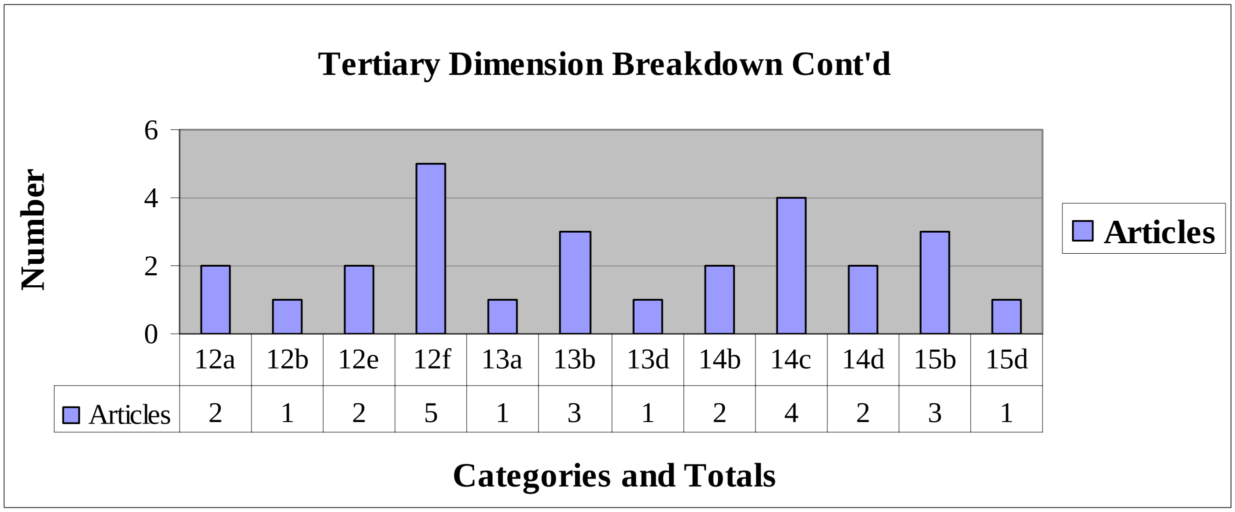


Figure 14 illustrates the breakdown of articles for the tertiary relationship starting with sub-category 12a and ending with sub-category 15d. The sub-categories are labeled following their respective main category heading: 12. Aspects: 12a. General, 12b. Methods & Tools, 12e. Models, 12f. Implementations; and 13. Weaving: 13a. General, 13b. Join Points, 13d. Implementation, Techniques & Models; and 14. Miscellaneous: 14b. Tools, 14d. Programming Techniques; and 15. Comparisons & Contrasts: 15b. Programming Techniques, 15d. Separating Concerns.

The sub-category with the most articles in figure 14 is 12f with 5 and second is sub-categories 13b and 15b with 3. The sub-categories with the least amount of articles are 12b, 13a, 13d, and 15d with 1 article each.

In the tertiary dimension, sub-category 1d, (Separating Concerns) has the largest number of articles with a total of 11. The sub-category 12f (Models) has the second largest amount of articles, 10. Lastly, the tertiary dimension consists of a total of 77 articles.

Quaternary Dimension

Figure 15: Quaternary Dimension Totals

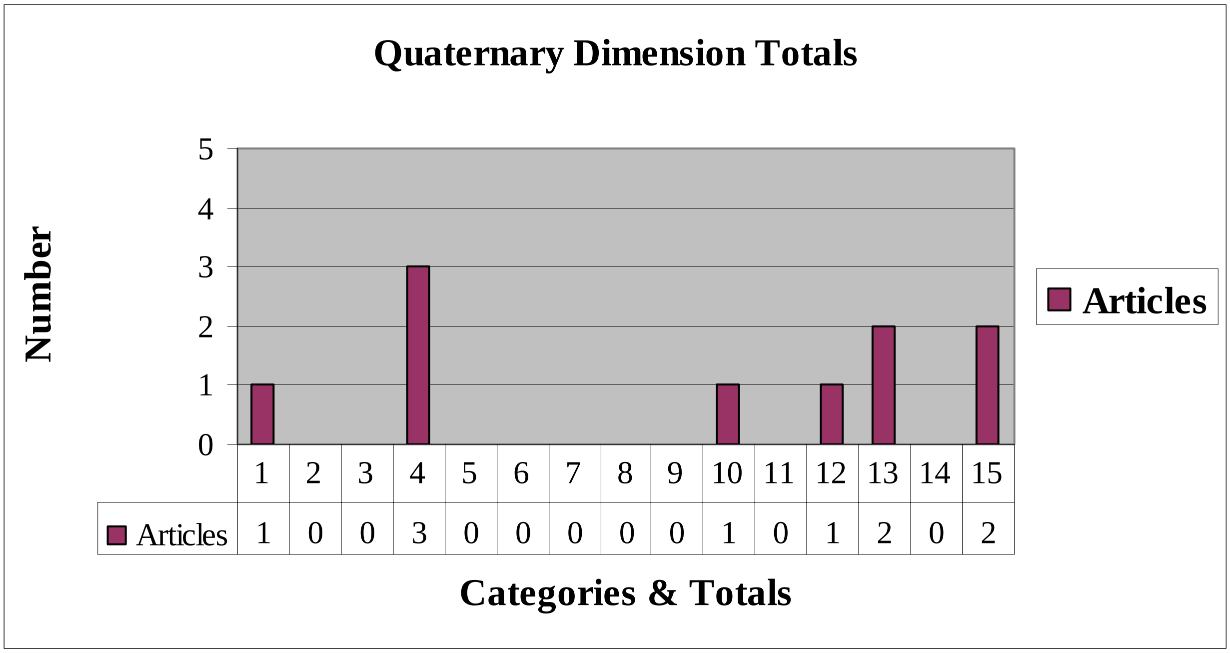


Figure 15 shows the number of articles and the category they belong to in the quaternary dimension of the model. The quaternary categories are numbered 1-15 and include Concerns; Adaptive; Limitations; Experiential/Case Study; Performance/Reliability; Distributed Systems; Theory; UML Modeling; Architecture; Software Engineering; Trends/Direction; Aspects; Weaving; Miscellaneous; and Comparisons/Contrasts. Category 4 had the highest number of articles with a total of 3 and categories 2, 3, 5-9, 11, and 14 had the least with a total of 0. The Figure 16 shows the categorical breakdown of the quaternary dimension.

Figure 16: Quaternary Dimension Breakdown (1D – 15B)

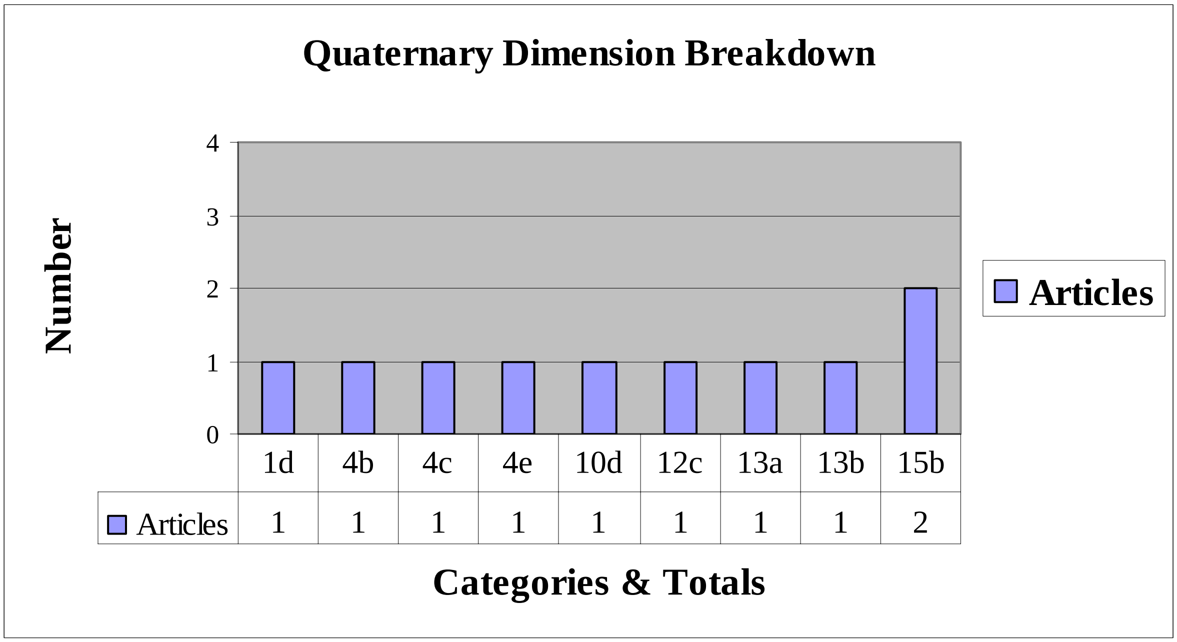


Figure 16 illustrates the breakdown of articles for the quaternary relationship with sub-categories 1d – 15b. Categories not represented in the graph did not have any quaternary relationships. The sub-categories are labeled following their respective main category heading: 1. Concerns: 1d. Separating Concerns; and 4. Experiential & Case Study: 4b. Tools & Languages, 4c. Design Aids & Analysis, 4e. Simulation & Modeling; and 10. Software Engineering: 10d. Design & Implementation; and 13. Weaving: 13a. General, 13b. Join Points; and 15. Comparisons & Contrasts: 15b. Programming Techniques.

All the sub-categories in figure 16 have 1 article except sub-category 15b (Programming Techniques), which is the highest sub-category with a total of 2 articles. The quaternary dimension consists of 10 articles.

AOP Yearly Literature Trends

The AOP collected literature illustrates a data timeline that can assist researchers in finding and seeing the amount of literature produced in a given year, and possibly determine the "hot" or "done" areas (see figures 17-18).

Figure 17: AOP Year Breakdown

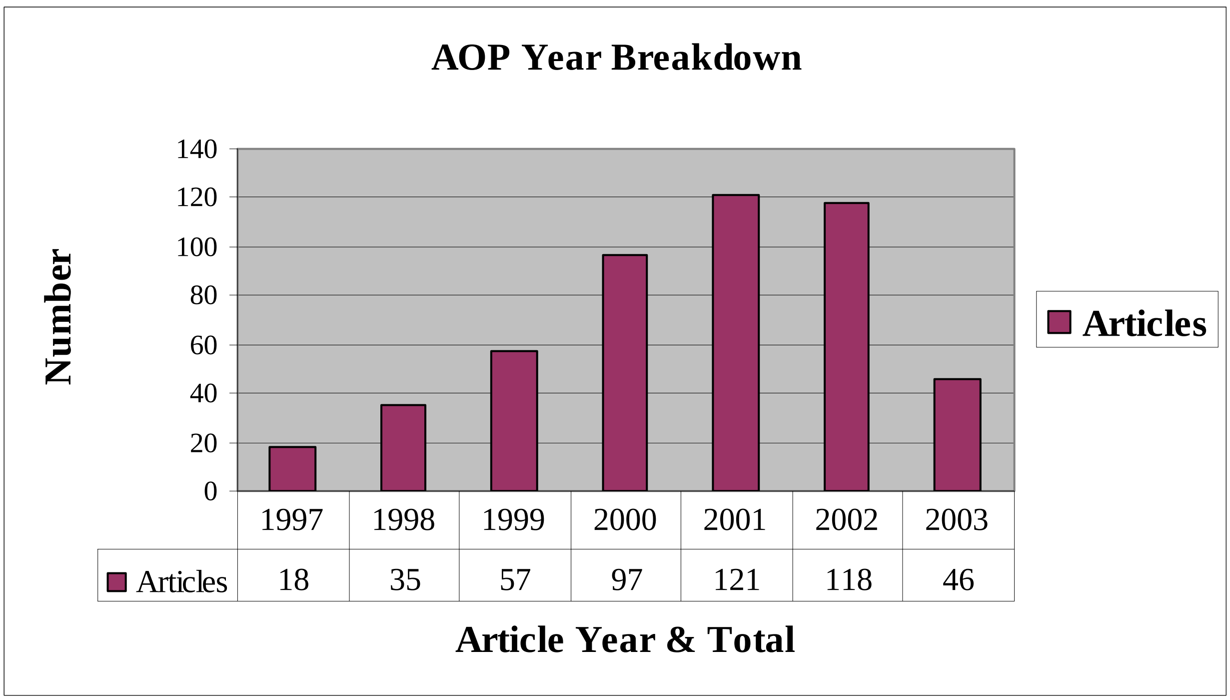


Figure 17 shows that the majority of cataloged literature was done in 2001 with a total of 121 articles and the least amount of written material was in 1997 with a total of 18 articles. However, not indicated in figure 17 are 1993 and 1995 with both years having 1 article each. Furthermore, the years were excluded because AOP was still being developed and not yet widely introduced into research and academic environments until 1997 as a programming paradigm. Thus, figure 17 begins with the year 1997 coinciding with AOP's availability and introduction to academic and research communities. The years 2002 and 2003 are lower than 2001 and seem to indicate a waning interest in AOP, however; the literature used in the taxonomy was not all inclusive. In addition, the year 2003 has significantly fewer articles (46) compared to 2001 (121) and 2002 (118) articles because the literature gathering stage ended for this project during the early part of summer 2003.

<u>Figure 18: AOP Leading Category per Year</u>

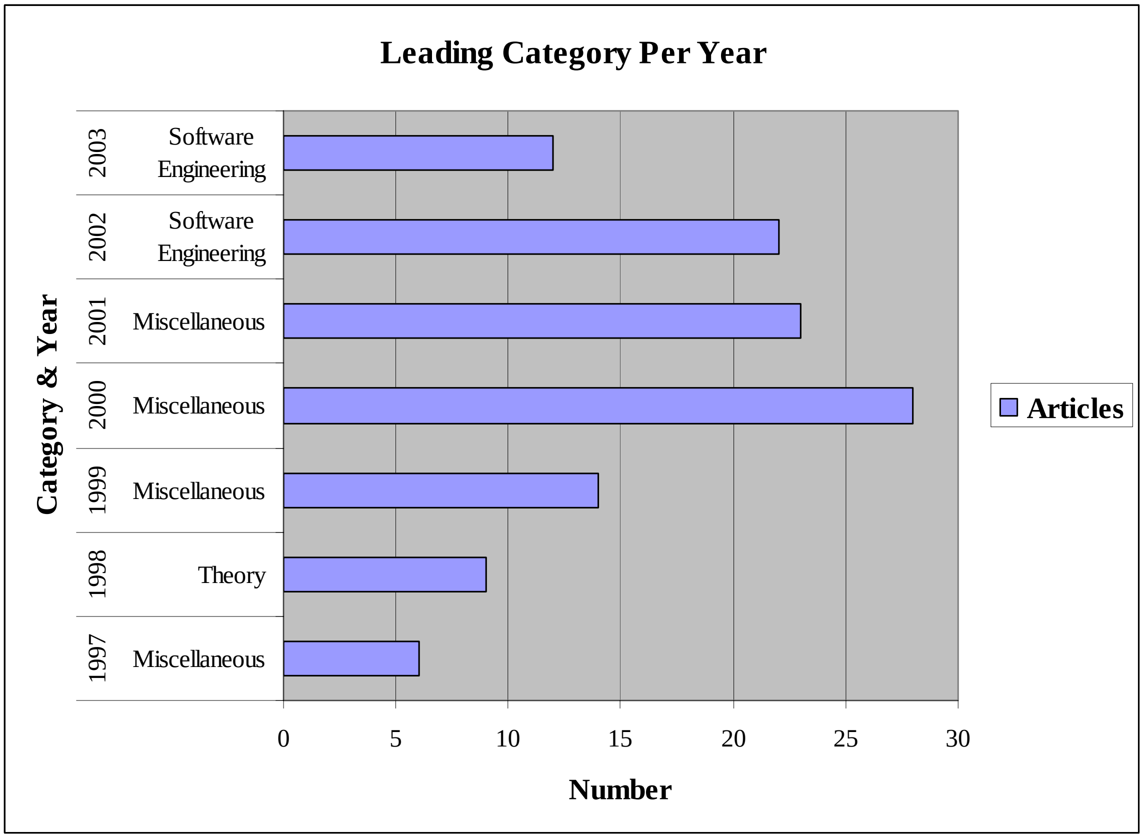


Figure 18 illustrates the leading category per year that received the most focus by researchers in the primary dimension. The Miscellaneous category in 1997 with 6 articles is the lowest leading category per year compared to other years. However, the Miscellaneous category in 2000 with 28 articles is the overall highest leading category per year compared to other years, which indicates the bulk of AOP literature for that year. In addition, the Miscellaneous category was represented the most out of all the categories created in the taxonomy, hence outpacing the other categories during the following years: 1997 (6 articles), 1999 (14 articles), 2000 (28 articles), and 2001 (23 articles). Furthermore, the figure shows that the Miscellaneous category may be a “done” focus area, and that the Theory category may be a “hot” area for researchers to contribute research. However, the Miscellaneous category can be an ever changing category that encompasses new and various topics, thus, it cannot really be considered a “done” area. The Software Engineering category is the second leading category among the other categories, specifically in the years 2002 (22 articles), and 2003 (12 articles) with a combined total of 34 articles. Overall, figure 18 indicates the top category per year for AOP literature and the number of articles that compose the category.

Figure 19: AOP Leading Sub-Category per Year

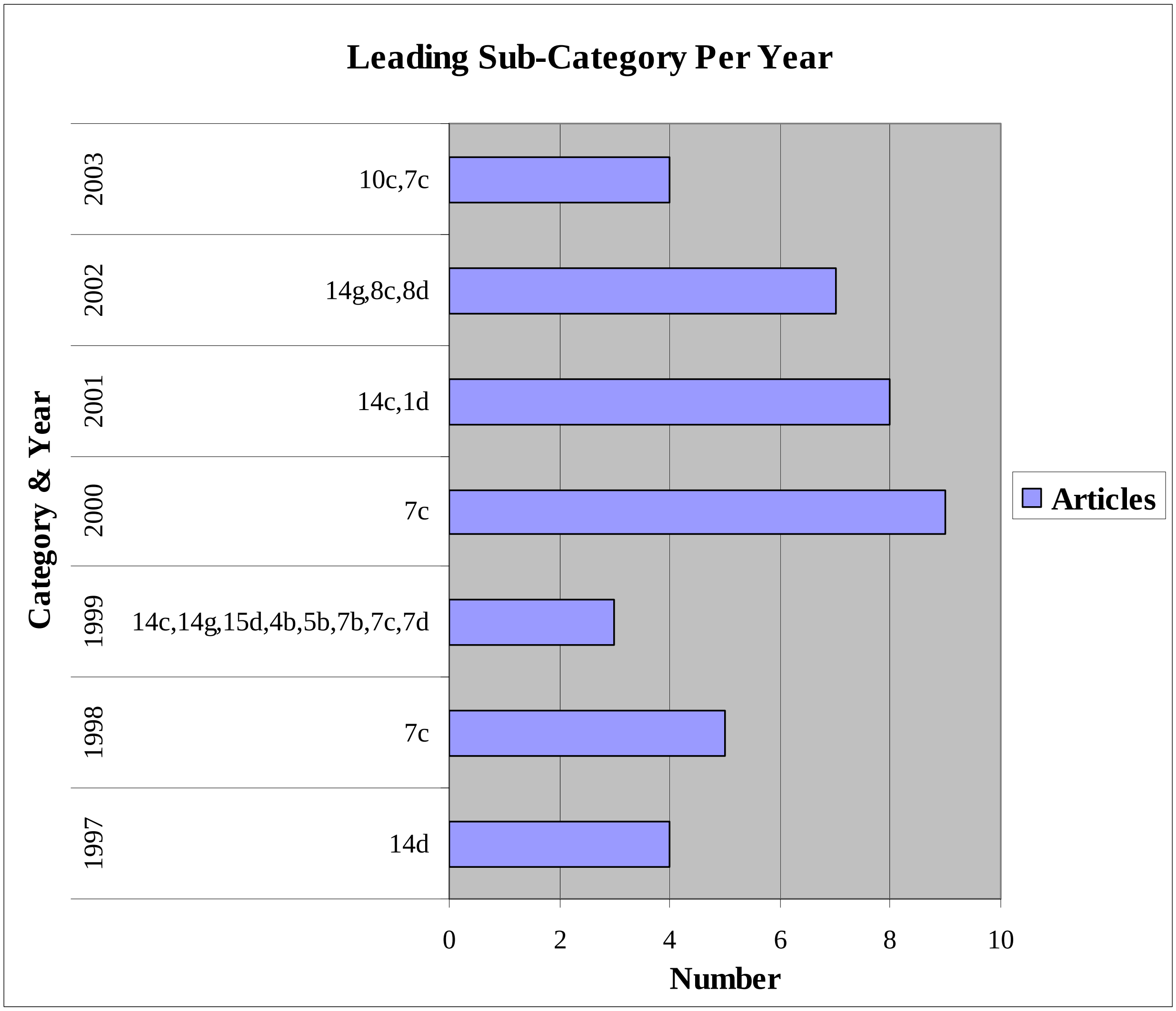


Figure 19 shows the leading sub-categories per year that received the most focus by researchers in the primary dimension. The year 1999 had the most sub-categories (8) represented with each having a total of 3 articles, thus, illustrating the categorical variance of literature contributed to AOP during 1999. The years 1997, 1998, and 2000 have the least amount of sub-categories represented with 1 each. The year 2000 had the most articles in a sub-category with 7c (Applied) under the Theory category having a total of 9 articles. The figure indicates that sub-category 1d (Separating Concerns) under the Concerns category in 2001 and that sub-category 10c (Processes & Metrics) under the Software Engineering category in 2003 may be a “hot” areas. However, the figure also shows that sub-categories 7c (Applied) under the Theory category and 14c (Concerns) under the Miscellaneous category may be “done” areas. Lastly, the inferences drawn from the figure only show the leading sub-categories per year and the number of articles composed in the sub-category.

Conclusion

The taxonomy on AOP is useful because it helps to create a road map to future research needed on aspect-oriented programming. It creates an easy mechanism for viewing the leading papers on the topic as well as the trends of previous research, creating a tool for future researchers. Overall, the research trending analysis gives researchers various perspectives using the multi-dimensional views about how and where AOP is being applied.

CHAPTER 4

CONCLUSIONS AND FUTURE WORK

This thesis has shown the distinctly different and overlapping nature of some the aspect-oriented programming (AOP) literature, and accounted for that by implementing four dimensional perspectives, primary, secondary, tertiary, quaternary, which show the varying applicability of the AOP paradigm. However, a discrepancy exists in the literature composed within the fifteen categories and sub-categories used in the classification. That is, some categories and sub-categories contain greater numbers of literature than other categories and sub-categories, e.g., the category Software Engineering comprised of 62 total articles versus the category Limitations with 7 total articles, and the sub-category Design and Implementation comprised of 22 articles under the Software Engineering category versus Theoretical with 1 total article under the Limitations category. Furthermore, articles that focused almost entirely on a certain category, e.g., Theory, or sub-category e.g., Formal Methods were placed into their respective main category and sub-category, which was done after extensive reviews of each article to better designate the appropriate focus area for the literature.

The taxonomy indicates research trends of the catalogued literature, thus showing areas where AOP is being applied and possible future direction. An in-depth look at the Miscellaneous category shows that it has been contributed to the most and may be a “done” area of research, but of course will continue to grow due to the nature of the category. Conversely, several categories are potential “hot areas where research should continue, i.e., Adaptive, and Weaving. A clear caveat must be factored into the taxonomy results when interpreting the trends of the data, which is the collection of AOP articles gathered is not all inclusive of research literature on the paradigm, thus creating a limitation of this thesis. The corollary for the limitation of the research is to continue to add more bibliographic data; however, this was not possible due to project timeline specifications.

The future has many possibilities for the application of AOP and the data incorporated in this thesis. Numerous articles have been categorized from various conferences, journals, and technical reports within the framework of the four dimensional classification model implemented in the project. Researchers can build on the taxonomy presented in the body of work to enhance the AOP domain. The taxonomy will be made publicly available so that researchers can extend and modify the project to improve the categorizations overall. The primary application of this thesis for future work is to enlighten others to the growing trends and "hot areas" of aspect-oriented programming, perhaps urging developers and researchers to build better tools and processes surrounding the AOP paradigm.

In conclusion, a comprehensive analysis of the AOP literature should be completed annually or semi-annually for tracking purposes. Furthermore, the database should be continually updated and modified in accordance with the comprehensive analysis to keep up with the increased pace of computer science literature, and possibly connected with other bibliographic database driven search engines[3]. The reports produced from the taxonomy will help researchers understand which areas of research are being tapped into and will help focus efforts to commercialize aspect-oriented programming. Lastly, the research conducted can be greatly improved if others continue to use the taxonomy so that all AOP articles, both past and present, are classified.

[3] Several bibliography search engines exist such as: http://citeseer.org/ and http://liinwww.ira.uka.de/bibliography/index.html

REFERENCES

[Achermann ‘00] Achermann, F., (2000), “Language support for feature mixing”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[Anderson ‘99] Anderson, E., and Patterson, D., (1999), “A retrospective on twelve years of LISA proceedings, in Proceedings of the 13th Conference on Systems Administration (LISA 1999).

[Akkawi ‘01] Akkawi, F., Bader, A., and Elrad, T., (2001), “Dynamic weaving for building reconfigurable software systems”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘01).

[Aksit ‘01] Aksit, M., Tekinerdogan, B., and Bergmans, L., (2001), “The six concerns for separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP 2001).

[Aldawud ‘02] Aldawud, O., Bader, A., and Elrad, T., (2002), “Weaving with statecharts”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD-2002).

[Avdicausevic ‘01] Avdicausevic, E., Lenic, M., Mernik, M., and Zumer, V., (2001), “Aspect COOL: An experiment in design and implementation of aspect-oriented language”, ACM Sigplan Notices, vol. 36, no. 12.

[Bardou ‘98] Bardou, D., (1998), “Roles, subjects, and aspects: How do they relate?”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP 1998).

[Bollert ‘99] Bollert, K., (1999), “On weaving aspects”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP 1999).

[Brichau ‘02] Brichau, J., Mens, K., and De Volder, K., (2002), “Building composable aspect-specific languages with logic metaprogramming”, GPCE ’02.

[Brooks ‘95] Brooks, Robert Jr., (1995). “No silver bullet- essence and accident in software engineering”, in *The mythical man-month,* Reading, MA: Addison Wesley.

[Capretz ‘03] Capretz, L.F., (2003), “A brief history of the object-oriented approach”, *Software Engineering Notes,* vol 28, no 2.

[Chavez ‘01] Chavez, C.V.F.G., and Lucena, C.J.P., (2001), “Design-level support for aspect-oriented software development”, in Proceedings of OOPSLA 2001.

[Coady ‘03] Coady, Y. and Kiczales, G., (2003), “Back to the future: A retroactive study of aspect evolution in operating system code”, ACM 2003 (AOSD 2003).

[Constantinides ‘00] Constantinides, C.A., Bader, A., and Elrad, T., (2000), “An aspect-oriented design framework.” ACM Computing Surveys, March 2000.

[Constantindes ‘01] Constantinides, C.A., Skotiniotis, T., and Elrad, T., (2001), “Providing dynamic adaptability in an aspect-oriented framework”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[Cox ‘86] Cox, Brad J., (1986). *Object oriented programming: An evolutionary approach.* Reading, MA: Addison-Wesley.

[Dijkstra ‘76] Dijkstra, Edsger W., (1976). *A discipline of programming.* Englewood, Cliffs, NJ: Prentice-Hall Inc.

[Dolog ‘01] Dolog, P., Vranic, V., and Bielikova, M, (2001), “Representing change by aspect”, *ACM Sigplan Notices*, vol. 36, no. 12.

[Elrad ‘01a] Elrad, T., Aksits. M., Kiczales, G., Lieberherr, K., and Ossher, H., (2001), “Discussing aspects of aop”, *Communications of the ACM*, vol. 44, no. 10.

[Elrad ‘01b] Elrad, T., Filman, R.E., and Bader, A., (2001), “Aspect oriented programming”, *Communications of the ACM*, vol. 44, no. 10.

[Fabry ‘01] Fabry, J., Brichau, J., and Mens, T., (2001), “Moving code”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[Filman ‘02] Filman, Robert E., (2002). A bibliography of aspect-oriented software development, version 1.0. RIACS technical report 02.06. August 2002.

[Giese ‘00] Giese, H. and Vilbig, A, (2000), “Towards aspect-oriented design and architecture”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[Highley ‘99] Highley, T.J., Lack, M., and Myers, P., (1999), “Aspect oriented programming: A critical analysis of a new programming paradigm, University of Virginia, Department of Computer Science Technical Report CS-99-29.

[Hilliard ‘99] Hilliard, R., (1999), “Aspects, concerns, subjects, views,..”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA 1999).

[Holmes ‘97] Holmes, D., Noble, J., and Potter, J., (1997), “Aspects of synchronization”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[Kiczales '97] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., and Irwin, J., (1997), "Aspect-oriented programming", in Aksit, M. and Matsuoka, S., editors, 11th European Conference Object-Oriented Programming, vol. 1241 of LNCS, pgs. 220-242. Springer Verlag.

[Laddad '02] Laddad, R., (2002), "I want my aop! Part 1: Separate software concerns with aspect-oriented programming", in Proceedings of Java World Fueling Innovations, 2002.

[Lopes '97] Lopes, C.V., and Kiczales, G., (1997), "D: A language framework for distributed programming:, Xerox, PARC, Palo Alto, CA. Technical report SPL97-010 P9710047.

[Lopes '99] Lopes, C.V., (1999), "Modularization revisited: Aspects in the design and evolution of software systems", IEEE.

[Ludy '02] Ludy, T., (2002), "Aspect-oriented programming can lead to faster, cheaper, easier-to-use solutions", *Control Solutions*.

[Navasa '01] Navasa, A., Perez, M.A., and Murillo, J.M., (2001), "Developing component based systems using AOP concepts", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[Nordberg '01] Nordberg, M.E.III., (2001), "Aspect-oriented dependency inversion", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[Orleans '01] Orleans, D., (2001), "Separating behavioral concerns with predicate dispatch, or, if statement considered harmful", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[Ortin '02] Ortin, F. and Cueva, J.M., (2002), "Implementing a real computational-environment jump in order to develop a runtime-adaptable reflective platform", ACM Sigplan Notices, vol. 37, no. 8.

[Ossher '98] Ossher, H., (1998), "Operation-level composition: A case in (join) point", in Proceedings of ICSE 98.

[Parnas '01] Parnas, David L., (2001). Some software engineering principles. in *Software fundamentals: collected papers by David Parnas*. Edited by Hoffman, Daniel M., and Weiss, David M. New York, New York: Addison-Wesley.

[Pawlak' 02] Pawlak, R., Duchien, L., Florin, G., Legond-Aubry, F., Seinturier, L., and Martelli, L., (2002), "A UML notation for aspect-oriented software design", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '02).

[Putrycz ‘02] Putrycz, E. and Bernard, G., (2002), “Using aspect oriented programming to build a portable load balancing service”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02).

[Rashid ‘02] Rashid, A., (2002), “Weaving aspects in a persistent environment”, *ACM Sigplan Notices*, vol. 37, no. 2.

[Soares ‘02] Soares, S., and Borba, P., (2002), “Progressive implementation with aspect-oriented programming”, in Springer Verlag, editor, The 12th Workshop for PhD Students in Object--Oriented Systems, ECOOP 02.

[Stein ‘02] Stein, D., Hanenberg, S., and Unland, R., (2002), “On representing join points in the uml”, in Proceeding of AOSD 2002.

[Van Roy ‘97] Van Roy, P., Haridi, S., Brand, P., Smolka, G., Mehl, M., and Scheidhauer, R., (1997), “Using mobility to make transparent distribution practical”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[Walker ‘99] Walker, R.J., Baniassad, E.L.A., and Murphy, G.C., (1999), “An initial assessment of aspect-oriented programming”, in Proceedings of 21st Int’l Conference Software Engineering (ICSE ‘99).

APPENDICES

APPENDIX A

Literature Used in Taxonomy

[1] Achermann, F., (2000), “Language support for feature mixing”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[2] Akkawi, F., Bader, A., and Elrad, T., (2001), “Aspect-oriented technology for business applications: A case study in stock trading”, IEEE ‘01.

[3] Akkawi, F., Bader, A., and Elrad, T., (2001), “Dynamic weaving for building reconfigurable software systems”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘01).

[4] Aksit, M. and Bergmans, L., (1998), “Examples of reusing synchronization code in aspect-oriented programming using composition filters”, MCSEAI ’98.

[5] Aksit, M and Tekinerdogan, B., (1998), “Formalizing adaptability aspects”, in Proceedings of ICSE ‘98.

[6] Aksit, M. and Tekinerdogan, B., (1998), “Solving the modeling problems of object-oriented languages by composing multiple aspects using composition filters”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[7] Aksit, M., Tekinerdogan, B., and Bergmans, L., (2001), “The six concerns for separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[8] Aldawud, O., Bader, A., and Elrad, T., (2002), “Weaving with statecharts”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[9] Aldawud, O., Elrad, T., and Bader, A., (2001), “A UML profile for aspect oriented modeling”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[10] Aldawud, O., Elrad, T, and Bader, A., (2003), “Uml profile for aspect-oriented software development”, AOSD ’03.

[11] Aldrich, J., (2000), “Challenge problems for separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[12] Aldrich, J., Chambers, C., Sirer, E.G., and Eggers, S., (1999), “Static analyses for eliminating unnecessary synchronization from java programs.”

[13] Anderson, K., (2000), “An example of using collaborator and adapters to reuse a synchronization pattern”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[14] Andrews, J., (2001), “Process-algebraic foundations of aspect-oriented programming”, in *Reflection, LNCS 2192*.

[15] Andrews, J., (2001), “Using process algebra as a foundation for programming by separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[16] Araujo, J., Moreira, A., Brito, I., and Rashid, A., (2002), “Aspect-oriented requirements with uml”, in Proceedings of Workshop on Aspect Oriented Modeling with UML.

[17] Assmann, U., (1997), “Aop with design patterns as meta-programming operators”, *Technical Report 28*.

[18] Aβmann, U., (2000), “A component model for invasive composition”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[19] Aβmann, U. and Ludwig, A., (1999), “Aspect weaving with graph rewriting”, GCSE ’99.

[20] Atkinson, C. and Kuhne, T., (2000), “Separation of concerns through stratified architectures”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[21] Atkinson, C. and Kuhne, T., (2003), “Aspect-oriented development with stratified frameworks”, IEEE ‘03.

[22] Avdicausevic, E., Lenic, M., Mernik, M., and Zumer, V., (2001), “Aspect COOL: An experiment in design and implementation of aspect-oriented language”, *ACM Sigplan Notices*, vol. 36, no. 12.

[23] Bachmendo, B. and Unland, R., (2001), “Aspect-based workflow evolution”, in Proceedings of the Workshop on Aspect-oriented Programming and Separation of Concerns.

[24] Baker, J., and Hsieh, W., (2002), “Runtime aspect weaving through metaprogramming”, ACM 2002 (AOSD ‘02).

[25] Baniassad, E., Murphy, G.C., and Schwanninger, C., (2001), “Determining the “why” of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[26] Baniassad, E., Murphy, G.C., Schwanninger, C., and Kircher, M., (2000), “Where are programmers faced with concerns?”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[27] Baniassad, E., Murphy, G., Schwanninger, C., and Kircher, M., (2002), “Managing crosscutting concerns during software evolution tasks”, ACM ’02.

[28] Bardou, D., (1998), “Roles, subjects, and aspects: How do they relate?”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[29] Barros, J.P. and Gomes, L., (2002), “Activities as behaviour aspects”, in Proceedings of AOSD ‘02.

[30] Bastani, F.B., Yen, I.L., Sung, K., Linn, J., and Rao, K., (2001), “Reliability of systems of independently developable end-user assessable logical (ideal) programs”, IEEE ‘01.

[31] Batenin, A. and O’Neill, E., (2002), “Towards unanticipated composition of concerns in hyperspaces”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[32] Batory, D., (2000), “Refinements and separation of concerns”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[33] Bayer, J., (2000), “Towards engineering product lines using concerns”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[34] Becker, C. and Geihs, K., (1998), “Quality of service-aspects of distributed programs”, in Proceedings of Workshop on Aspect Oriented Programming (ICSE ‘98).

[35] Becker, C. and Geihs, K., (2001), “Quality of service and object-oriented middleware-Multiple concerns and their separation”, IEEE ‘01.

[36] Becker, U., (1998), “D2AL: A design-based aspect language for distribution control”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[37] Beier, G. and Kern, M., (2002), “Aspects in uml models from a code generation perspective”, in Proceedings of AOSD ‘02.

[38] Bergenti, F. and Poggi, A., (2000), “Aspect views as a means to promote reuse in aspect-oriented languages”, in Proceedings of ADC ‘00.

[39] Berger, L., (2000), “Junction point aspect: A solution to simplify implementation of aspect languages and dynamic management of aspect programs”, ECOOP ’00.

[40] Berger, L., Dery, A.M., and Fornarino, M., (1998), “Interactions between objects: An aspect of object-oriented languages”, in Proceedings of Int’l Workshop on Aspect Oriented Programming (ICSE ‘98).

[41] Bergmans, L. and Aksit, M., (1999), “Analyzing multi-dimensional programming in AOP and composition filters”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[42] Bergmans, L.M.J. and Aksit, M., (2000), “Aspects & crosscutting in layered middleware systems”, in Proceedings of Workshop in Reflective Middleware ’00.

[43] Bergmans, L. and Aksit, M., (2000), “Composing software from multiple concerns: A model and composition anomalies”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[44] Bergmans, L. and Aksit, M., (2001), “Composing crosscutting concerns using composition filters”, *Communications of the ACM*, vol. 44, no. 10.

[45] Bergmans, L. and Aksit, M., (2001), “How to deal with encapsulation in aspect-orientation”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[46] Bergmans, L., Tekinerdogan, B., Glandrup, M., and Aksit, M., (2000), “On composing separated concerns: Composability and composition anomalies”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[47] Beugnard, A., (1999), “How to make aspect reusable, a proposition”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[48] Black, A.P. and Jones M.P., (2000), “Perspectives on software”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[49] Black, A.P. and Walpole, J., (2000), “Aspects of information flow”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[50] Blair, L. and Blair, G.S., (1998), “The impact of aspect-oriented programming on formal methods”, *Technical Reports*, MPG-98-08, Lancaster University.

[51] Blair, L. and Blair, G., (1999), “A tool suite to support aspect-oriented specification”, in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[52] Blair, L. and Monga, M., (2003), “Reasoning on aspectJ programmes”, GI-AOSDG ’03.

[53] Blank, G. and Vayngrib, G., (1998), “Aspects of enterprise java beans”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[54] Blay-Fornarino, M., Pinna-Derry, A.M., and Riveill, M., (2002), “Towards dynamic configuration of distributed applications”, in Proceedings of ICDCS ‘02.

[55] Bogda, J. and Holzle, U., (1999), “Removing unnecessary synchronization in java”, ACM ’99, (OOPSLA ’99).

[56] Bollert, K., (1998), “Aspect-oriented programming case study: System management application”, in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP ‘98).

[57] Bollert, K., (1999), “On weaving aspects”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[58] Borba, P. and Soares, S., (2002), “Refactoring and code generation tools for aspectJ”, OOPSLA ’02.

[59] Bouraqadi, N., (2000), “Concern oriented programming using reflection”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[60] Bouraqadi-Saadanii, M.M.N. and Ledoux, T., (2001), “How to weave?”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[61] Braga, A.M., Darab, R., and Rubira, C.M.F., (2000), “A meta-object protocol for secure composition of security mechanisms”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[62] Brichau, J., (2000), “Declarative composable aspects”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[63] Brichau, J., De Meuter, W., and De Volder, K., (2000), “Jumping aspects”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[64] Brichau, J., Mens, K., and De Volder, K., (2002), “Building composable aspect-specific languages with logic metaprogramming”, GPCE ’02.

[65] Brito, I., Moreira, A., and Araujo, J., (2002), “A requirements model for quality attributes”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[66] Brodsky, A., Brodsky, D., Chan, I., Coady, Y., Gudmundson, S., Pomkoski, J., and Ong, J.S., (2001), “Coping with evolution: Aspects vs. aspirin”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[67] Buhr, R.J.A., (1998), “A possible design notation for aspect-oriented programming”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[68] Bussard, L., (2000), "Towards a pragmatic composition model of CORBA services based on AspectJ", in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP '00).

[69] Cabri, G., Leonardi, L., and Zambonelli, F., (2002), "Separation of concerns in agent applications by roles", in Proceedings of 2nd Int'l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS '02), Vol. 2.

[70] Campo, M.R. and Diaz-Pace, J.A., (2001), "Analyzing the role of aspects in software design", *Communications of the ACM*, vol. 44, no. 10.

[71] Capouillez, A., Crescenzo, P., and Lahire, P., (2001), "Separation of concerns in OFL", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[72] Carver, L., (2000), "A practical hyperspace application: Lessons from the option processing task", in Proceedings of Workshop on Advanced Separation of Concerns (ICSE '00).

[73] Carver, L., (2000), "Combining selector-guarded blocks: Difficulties from the option-processing task", in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP '00).

[74] Carver, L., (2000), "Using brackets to corral jumping aspects", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[75] Carver, L., (2002), "Composition behaviors for application construction", in Proceedings of Workshop on Identifying, Separating, and Verifying Concerns in the Design (AOSD '02).

[76] Carver, L. and Griswold, W.G., (1999), "Sorting out concerns", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[77] Chavez, C. and Lucena, C.J., (2001), "Design-level support for aspect-oriented software development", in Proceedings of OOPSLA '01.

[78] Chavez, C. and Lucena, C., (2002), "A metamodel for aspect-oriented modeling", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '02).

[79] Chechik, M. and Easterbrook, S., (2001), "Reasoning about compositions of concerns", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[80] Chiba, S., (2001), "What are the best join points?", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[81] Chitchyan, R., Sommerville, I., and Rashid, A., (2002), “An analysis of design approaches for crosscutting concerns”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[82] Choi, J.P., (2000), “Aspect-oriented programming with enterprise javabeans”, IEEE 2000.

[83] Chu-Carroll, M., (2000), “Separation of concerns: An organizational approach”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[84] Chu-Carroll, M., (2000), “Software configuration management as a mechanism for multidimensional separation of concerns”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[85] Chu-Carroll, M., (2001), “Separation of concerns in software configuration management”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[86] Chu-Carroll, M., Wright, J., and Ying, A.T.T., (2003), “Visual separation of concerns through multidimensional program storage”, ACM ‘03 (AOSD ‘03).

[87] Clarke, S., (2000), “Designing reusable patterns of cross-cutting behaviour with composition patterns”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[88] Clarke, S., (2000), “Extending UML metamodel for design composition”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[89] Clarke, S. and Murphy, J., (1998), “Developing a tool to support the application of aspect-oriented programming principles to the design phase”, in Proceedings of Int’l Workshop on Aspect Oriented Programming (ICSE ‘98).

[90] Clarke, S., Harrison, W., Ossher, H., and Tarr, P., (1999), “Separating concerns throughout the development lifecycle”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[91] Clarke, S., Harrison, W., Ossher, H., and Tarr, P., (1999), “The dimension of separating requirements concerns for the duration of the development lifecycle”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[92] Clarke, S. and Walker, R.J., (2001), “Composition patterns: An approach to designing reusable aspects”, IEEE ‘01.

[93] Clarke, S. and Walker, R.J., (2001), “Mapping composition patterns to AspectJ and Hyper/J”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[94] Clarke, S. and Walker, R.J., (2001), “Separating crosscutting concerns across the lifecycle: From composition patterns to aspectJ and hyperJ”, *Technical Report TCD-CS.*

[95] Clarke, S. and Walker, R.J., (2002), “Towards a standard design language for AOSD”, ACM 2002, (AOSD ‘02).

[96] Clemente, P.J. and Hernandez, J., (2003), “Aspect component based software engineering”, AOSD ’03.

[97] Clemente, P.J., Hernandez, J. Murillo, J.M., Perez, M.A., and Sanchez, F., (2002), “Aspect CCM: An aspect-oriented extension of the corba component model”, IEEE ‘02.

[98] Clifton, C. and Leavens, G.T., (2002), “Observers and assistants: A proposal for modular aspect-oriented reasoning”, in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD ‘02).

[99] Coady, Y., Brodsky, A., Brodsky, D., Pomkoski, J., Gudmundson, S., Ong, J.S., and Kiczales, G., (2000), “Can AOP support extensibility in client-server architectures?”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[100] Coady, Y. and Kiczales, G., (2003), “Back to the future: A retroactive study of aspect evolution in operating system code”, ACM 2003 (AOSD 2003).

[101] Coady, Y., Kiczales, G., and Feeley, M., (2000), “Exploring an aspect-oriented approach to operating system code”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[102] Coady, Y., Kiczales, G., Feeley, M., Hutchinson, N., and Ong, J.S., (2001), “Structuring operating system aspects: Using AOP to improve OS structure modularity”, *Communications of the ACM*, vol. 44, no. 10, pp. 79-82.

[103] Coady, Y., Kiczales, G., Feeley, M., Hutchinson, N., and Ong, J.S., (2001), “Structuring system aspects”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE 2001).

[104] Coady, Y., Kiczales, G., Feeley, M., Hutchinson, N., Ong, J.S., and Gudmundson, S., (2001), “Position summary: Aspect-oriented system structure”, IEEE 2001.

[105] Coady, Y., Kiczales, G., Feeley, M., and Smolyn, G., (2001), “Using aspectC to improve the modularity of path-specific customization in operating system code”, ACM 2001 (ESEC/FSE 2001).

[106] Cohen, G.A., (1999), “Recombing concerns: Experience with transformation”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[107] Cohen, G.A., (2000), “A taxonomy of transformation”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[108] Constantinides, C.A., Bader, A., and Elrad, T., (1999), “A framework to address a two-dimensional composition of concerns”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[109] Constantinides, C., Bader, A., and Elrad, T., (1999), “An aspect-oriented design framework for concurrent systems”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP 1999).

[110] Constantinides, C.A., Bader, A., and Elrad, T., (2000), “An aspect-oriented design framework”, *ACM Computing Surveys*, March 2000.

[111] Constantinides, C.A., Bader, A., and Elrad, T., (2000), “Separation of concerns in the design of concurrent software systems”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[112] Constantinides, C.A., Bader, A., and Elrad, T., Fayad, M.E., and Netinant, P., (2000), “Designing an aspect-oriented framework in an object-oriented environment”, ACM ‘00.

[113] Constantinides, C.A. and Elrad, T., (2000), “On the requirements for concurrent software architectures to support advanced separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[114] Constantinides, C.A., and Elrad, T., (2000), “Towards a two-dimensional separation of concerns”, ACM ‘00 (OOPSLA ‘00).

[115] Constantinides, C.A., and Elrad, T., (2001), “Composing concerns with a framework approach”, IEEE ‘01.

[116] Constantinides, C.A., Skotiniotis, T., and Elrad, T., (2001), “Providing dynamic adaptability in an aspect-oriented framework”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[117] Costanza, P., (2000), “Separation of object identity concerns”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[118] Costanza, P., (2000), “Vanishing aspects”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[119] Costanza, P., Kniesel, G., and Austermann, M., (2001), “Independent extensibility for aspect-oriented systems”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[120] Cugola, G., Ghezzi, C., and Monga, M., (1999), “Coding different design paradigms for distributed applications with aspect-oriented programming”, WSDAAL ’99.

[121] Cugola, G., Ghezzi, C., Monga, M., and Picco, G.P., (2000), “Malaj: A proposal to eliminate clashes between aspect-oriented and object-oriented programming”, ICS ’00.

[122] Czarnecki, K. and Eisenecker, U.W., (2000), “Separating the configuration aspect to support architecture evolution”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[123] Czarnecki, K., Eisenecker, U.W., and Steyaert, P., (1997), “Beyond objects: Generative programming”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[124] David, P.C., Ledoux, T., and Bouraqadi-Saadani, N.M.N., (2001), “Two-step weaving with reflection using AspectJ”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[125] de Alwis, B., Gudmundson, S., Smolyn, G., and Kiczales, G., (2000), “Coding issues in AspectJ”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[126] de Bruin, H. and van Vliet, H., (2002), “Top-down composition of software architectures”, ECBS ’02.

[127] De Meuter, W., (1997), “Monads as a theoretical foundation for AOP”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[128] de Moura, A.L., Ururahy, C., Cerque, R., and Rodriguez, N., (2002), “Dynamic support for distributed auto-adaptive applications”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02), vol. 2.

[129] De Paoli, F., (2000), “Multidimensional separation of concerns”, in Proceedings of ICSE ‘00.

[130] De Volder, K., (1998), “Aspect-oriented logic meta programming”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[131] De Volder, K., (2000), “Inheritance with destructive mixins for better separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[132] De Volder, K., (2001), “Code reuse, an essential concern in the design of aspect languages?”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[133] De Win, B., Vanhaute, B., and De Decker, B., (2001), “Towards an open weaving process”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[134] Dempsey, J. and Cahill, V., (1997), “Aspects of system support for distributed computing”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[135] Denaro, G. and Monga, M., (2002), “An experience on verification of aspect properties”, ACM ’02, (IWPSE ‘01).

[136] Deters, M. and Cytron, R.K., (2001), “Introduction of program instrumentation using aspects”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[137] Devanbu, P., Balzer, B., Batory, D., Kiczales, G., Launchbury, J., Parnas, D., and Tarr, P., (2003), “Modularity in the new millennium: A panel summary”, IEEE ‘03.

[138] D’Hondt, M. and D’Hondt, T., (1999), “Is domain knowledge an aspect?”, in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[139] Diaz-Herrera, J.L., Chadha, J., and Pittsley, N., (2002), “Aspect-oriented UML modeling for developing embedded systems product lines”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[140] Dingwal-Smith, A. and Finkelstein, A., (2002), “From requirements to monitors by way of aspects”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[141] Dolog, P., Vranic, V., and Bielikova, M., (2001), “Representing change by aspect”, *ACM Sigplan Notices*, vol. 36, no. 12.

[142] Dominick, L., (1999), “Aspect of lifecycle control in a C++ framework”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[143] Dominick, L., (2000), “Position paper: Instrumentation aspects require symmetric join points”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[144] Dominick, L. and Ostermann, K., (2000), “Supporting extension of components with new paradigms”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[145] Douence, R., Motelet, O., and Sudholt, M., (2001), “A formal definition of crosscuts”, *Lecture Notes in Computer Science*, vol. 2192.

[146] Douence, R., Motelet, O., and Sudholt, M., (2001), “Sophisticated crosscuts for e-commerce”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[147] D’Souza, D., Sane, A., and Birchenough, A., (1999), “First-class extensibility for uml-packaging of profiles, stereotypes, patterns”, in Proceedings of OOPSLA ‘99.

[148] Duclos, F., Estublier, J., and Morat, P., (2002), “Describing and using non functional aspects in component based applications, ACM ‘02 (AOSD ‘02).

[149] Edison, T., Dongarra, J., and Eijkhout, V., (2003), “Applying aspect-orient programming concepts to a component-based programming model”, IEEE ‘03.

[150] Eide, E., Reid, A., Flatt, M., and Lepreau, J., (2001), “Aspect weaving as component knitting: Separating concerns with knit”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[151] Elrad, T., Aksits. M., Kiczales, G., Lieberherr, K., and Ossher, H., (2001), “Discussing aspects of aop”, *Communications of the ACM*, vol. 44, no. 10.

[152] Elrad, T., Filman, R.E., and Bader, A., (2001), “Aspect oriented programming”, *Communications of the ACM*, vol. 44, no. 10.

[153] Ernst, E., (2000), “Separation of concerns and then what”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[154] Ernst, E., (2000), “Syntax based modularization: Invasive or not?”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[155] Ernst, E., (2001), “Loosely coupled class families”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[156] Ernst, E. and Lorenz, D.H., (2003), “Aspects and polymorphism in aspectJ”, ACM ‘03 (AOSD ‘03).

[157] Fabry, J., (1998), “Replication as an aspect”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[158] Fabry, J., Brichau, J., and Mens, T., (2001), “Moving code”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[159] Feng, L., Marcus, A., and Schaffer, K., (2001), “An overview of aspect oriented programming”, in Advanced Programming Languages, Fall ’01 at Kent State University, Department of Computer Science.

[160] Filman, R.E., (1998), “Injecting ilities”, in Proceedings of Int’l Workshop on Aspect Oriented Programming (ICSE ‘98).

[161] Filman, R.E., (2000), “Applying aspect-oriented programming to intelligent synthesis”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[162] Filman, R.E., (2001), “What is aspect-oriented programming, revisited”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[163] Filman, R.E., (2003), “Understanding aop through the study of interpreters”, FOAL ’03.

[164] Filman, R.E. and Friedman, D.P., (2000), “Aspect-oriented programming is quantification and obliviousness”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[165] Filman, R.E. and Havelund, K., (2002), “Realizing aspects by transforming for events”, in Proceedings Automated Software Engineering Workshop on Declarative Meta Programming to Support Software Development.

[166] Filman, R.E. and Havelund, K., (2002), “Source-code instrumentation and quantification of events”, in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD ‘02), pgs. 45-49.

[167] Filman, R.E. and Lee, D.D., (2001), “Redirecting by injector”, IEEE ‘01.

[168] Fontura, M., (1999), “Dimension templates: Multi-dimensional separation of concerns in UML”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[169] Forman, I.R., (2000), “Superimposition: A form of separation of concerns for distributed systems”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[170] Fradet, P. and Sudholt, M., (1998), “AOP: Towards a generic framework using program transformation and analysis”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[171] Fradet, P. and Sudholt, M., (1999), “An aspect language for robust programming”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[172] Frezza, S.T., (2001), "The ubiquitous overlapping, dynamic concerns project vs. product", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[173] Furfaro, A., Nigro, L., and Pupo, F., (2002), "Aspect oriented programming using actors", in Proceedings of 2nd Int'l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS '02).

[174] Gal, A., Franz, M., and Beuche, D., (2003) "Learning from components: Fitting aop for system software", AOSP '03.

[175] Gal, A., Schroeder-Preikschat, W., and Spinczyk, O., (2001), "AspectC++" Language proposal and prototype implementation", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[176] Gal, A., Spinczyk, O., and Schroeder-Preikschat, W., (2002), "On aspect-orientation in distributed real-time dependable systems", IEEE '02.

[177] Garcia, A., Chavez, C., Silva, O., Silva, V., and Lucena C., (2001), "Promoting advanced separation of concerns in intra-agent and inter-agent software engineering", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[178] Garcia, A.F. and de Lucena, C.J.P., (2001), "An aspect-based object-oriented model for multi-agent systems", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[179] Geihs, K. and Becker, C., (2001) "A framework for re-use and maintenance of quality of service mechanisms in distributed object systems", ICSM '01.

[180] Genβler, T. and Lowe, W., (1998), "Correct composition of distributed systems", IEEE '98.

[181] Georg, G., Ray, I., and France, R., (2002), "Using aspects to design a secure system", IEEE '02.

[182] Giese, H., (2001), "Towards ruling component-based distributed systems with role-based modeling and cross-cutting aspects", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[183] Giese, H. and Vilbig, A, (2000), "Towards aspect-oriented design and architecture", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[184] Glandrup, M. and Rensink, A., (2001), "Formal foundations for reasoning about evolution", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[185] Graversen, K.B. and Beye, J., (2002), "Conceptual programming using roles", in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD '02).

[186] Gray, J., (2001), "Using software component generators to construct a meta-weaver framework", IEEE '01.

[187] Gray, J., Bapty. T., and Neema, S., (2000), "Aspectifying constraints in model-integrated computing", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[188] Gray, J., Bapty, T., Neema, S., and Tuck, J., (2001), "Handling crosscutting constraints in domain-specific modeling: Uniting aop with model-integrated computing", ACM '01.

[189] Greefhorst, D., (2000), "Separating concerns in software logistics", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[190] Grier, M., (1999), "Motivation for enabling separation of concerns in software product lines", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[191] Griswold, W.G., Kato, Y., and Yuan, J.J., (1999), "Aspect browser: Tool support for managing dispersed aspects", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[192] Groher, I. and Schulze, S., (2003), "Generating aspect code from uml models", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '03).

[193] Gruenbacher, P., Egyed, A., and Medvidovic, N., (2000), "Dimensions of concerns in requirements negotiation and architecture modeling", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[194] Grundy, J., (1998), "Aspect-oriented requirements engineering for component-based software systems", IEEE '98.

[195] Grundy, J., (2000), "A method and support environment for distributed software component engineering", IEEE '00.

[196] Grundy, J. and Patel, R., (2001), "Developing software components with the uml, enterprise java beans and aspects", IEEE '01.

[197] Gschwind, T. and Oberleitner, J., (2003), "Improving dynamic data analysis with aspect-oriented programming", IEEE '03.

[198] Gudmundson, S. and Kiczales, G., (2001), "Addressing practical software development issues in AspectJ with a pointcut interface", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[199] Gybels, K., and Brichau, J., (2003), "Arranging language features for more robust pattern-based crosscuts", ACM '03 (AOSD '03).

[200] Habra, N., (2001), "Separation of concerns in software engineering education", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE 2001).

[201] Hanenberg, S. and Unland, R., (2001), "Concerning aop and inheritance", Technical Report, tr-ri-01-223.

[202] Hanenberg, S. and Unland, R., (2001), "Grouping objects using aspect-oriented adapters", in Proceedings of Workshop on Aspect-oriented programming & Separation of Concerns.

[203] Hanenberg, S. and Unland, R., (2001), "Using and reusing aspects in AspectJ", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[204] Hanenberg, S. and Unland, R., (2002), "A proposal for classifying tangled code", AOSD '02.

[205] Hanenberg, S., and Unland, R., (2003), "Parametric introductions", ACM '03 (AOSD '03).

[206] Hannemann, J. and Kiczales, G., (2001), "Overcoming the prevalent decomposition in legacy code", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE 2001).

[207] Hannemann, J. and Kiczales, G., (2002), "Design pattern implementation in java and aspectJ", ACM '02 (OOPSLA '02).

[208] Hao, R., Boloni, L., Jun, K., and Marinescu, D.C., (1998), "An aspect-oriented approach to distributed object security", IEEE '98.

[209] Harrison, W., (2001), "Composition and multiple-inheritance in OO design (Where in the madness is the method?)", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[210] Harrison, W. and Ossher, H., (2002), "Member-group relationships among objects", in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD '02).

[211] Harrison, W., Ossher, H., and Tarr, P., (1997), "The beginnings of a graphical environment for subject-oriented programming", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '97).

[212] Harrison, W., Tarr, P., and Ossher, H., (2002), "A position on considerations in UML design of aspects", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '02).

[213] Hauck, F.J., Becker, U., Geier, M., Meier, E., Rastofer, U., and Steckermeier, M., (1998), "AspectIX: A middleware for aspect-oriented programming", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[214] Hernandez, J., Papathomas, M., Murillo, H.M., and Sanchez, F., (1997), "Coordinating concurrent objects: How to deal with the coordination aspect?", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '97).

[215] Herrero, J.L., Sanchez, F., and Toro, M., (2001), "Fault tolerance as an aspect using Jreplica", IEEE '01.

[216] Herrmann, S., (2000), "Dynamic view connectors for separating concerns in software engineering environments", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[217] Herrmann, S., (2002), "Composable designs with UFA", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '02).

[218] Herrmann, S. and Mezini, M., (2000), "On the need for a unified MDSOC model: Experiences from constructing a modular software engineering environment", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[219] Herrmann, S. and Mezini, M., (2000), "PIROL: A case study for multidimensional separation of concerns in software engineering environments", ACM '00 (OOPSLA '00).

[220] Herrmann, S. and Mezini, M., (2001), "Combining composition styles in the evolvable language LAC", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[221] Herrmann, S. and Wloka, J., (2003), "Seven steps from an idea to industrial adoption", AOSD '03.

[222] Highley, T.J., Lack, M., and Myers, P., (1999), "Aspect oriented programming: A critical analysis of a new programming paradigm", University of Virginia, Department of Computer Science Technical Report CS-99-29.

[223] Hilliard, R., (1999), “Aspects, concerns, subjects, views..”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[224] Hirschfeld, R., (2001), “Aspects – AOP with squeak”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[225] Hirschfield, R., (2002), “Advice activation in aspectS”, AOSD ’02.

[226] Hirschfield, R., (2002), “Aspect-oriented programming with aspectS”, NODE ’02.

[227] Hirschfield, R., Lammel, R., and Wagner, M., (2003), “Design patterns and aspects-Modular designs with seamless run-time integration”, in Proceedings of the 3rd German International Workshop on Aspect-oriented Software Development.

[228] Hirschfield, R., Osterbye, K., and Wagner, M., (2003), “System integration using aop”, AOSD ’03.

[229] Ho, W.M., Jezequel, J.M., Pennaneac’h, F., and Plouzeau, N., (2002), “A toolkit for weaving aspect oriented uml designs”, ACM ’02, (AOSD ‘02).

[230] Ho, W.M., Pennaneach, F., Jezequel, M., and Plouzeau, N., (2000), “Aspect-oriented design with the UML”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[231] Ho, W.M., Pennaneach, F., Jezequel, M., and Plouzeau, N., (2000), “UMLAUT: A framework for weaving uml-based aspect-oriented designs”, IEEE ‘00.

[232] Holmes, D., Noble, J., and Potter, J., (1997), “Aspects of synchronization”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[233] Holmes, D., Noble, J., and Potter, J., (1998), “Towards reusable synchronization for object-oriented languages”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[234] Hruby, P., (1999), “Dimensions for the separation of concerns in describing software development processes”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[235] Huang, J., (2000), “Experience using AspectJ to implementation cord”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[236] Hunleth, F. and Cytron, R.K., (2002), “Footprint and feature management using aspect-oriented programming techniques”, ACM ‘02 (LCTES ‘02 and SCOPES ‘02).

[237] Hunleth, F., Cytron, R., and Gill, C., (2001), “Building customizable middleware using aspect oriented programming”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[238] Hursch, W.L. and Lopes, C.V., (1995), “Separation of concerns”, *Northeastern University Technical Report NU-CCS-95-03*.

[239] Ishio, T., Kusumoto, S., and Inoue, K., (2003), “Application of aspect-oriented programming to calculation of program slice”, *IPSJ Journal*, vol. 44, no. 7. {C: 10c}

[240] Jezequel, J.M., Plouzeau, N., Weis, T., and Geihs, K., (2002), “From contracts to aspects in UML designs”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[241] Kaminski, P., (2001), “Applying multi-dimensional separation of concerns to software visualization”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[242] Kande, M.M., Kienzle, J., and Strohmeier, A., (2002), “From aop to uml - A bottom-up approach”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02)

[243] Kande, M.M., Kienzle, J., and Strohmeier, A., (2002), “From aop to uml: Towards an aspect-oriented architectural modeling approach”, in Proceedings of Workshop on Aspect Oriented Modeling.

[244] Kande, M.M. and Strohmeier, A., (2000), “On the role of multi-dimensional separation of concerns in software architecture”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[245] Kande, M.M. and Strohmeier, A., (2001), “Modeling crosscutting concerns using software connectors”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[246] Kaplan, M., (2000), “Dynamic selection: The discriminating developer’s way to compose”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[247] Kasten, E.P., McKinley, P.K., Sadjadi, S.M., and Stirewalt, R.E.K., (2002), “Separating introspection and intercession to support metamorphic distributed systems”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02).

[248] Katara, M., (2002), “Superimposing UML class diagrams”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[249] Katara, M. and Katz, S., (2002), "Towards aspect architectures and remodularization", in Proceedings of The Israeli Workshop on Programming Languages and Development (PLE '02).

[250] Katara, M. and Katz, S., (2003), "Architectural views of aspects", ACM '03 (AOSD '03).

[251] Katara, M. and Mikkonen, T., (2001), "Aspect-oriented specification architectures for distributed real-time systems", IEEE '01.

[252] Katara, M. and Mikkonen, T., (2002), "Refinements and aspects in uml", in Proceedings of AOSD '02.

[253] Katz, S. and Gil, Y., (1999), "Aspects and superimpositions", in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP '99).

[254] Kellomaki, P., (2002), "A formal basis for aspect-oriented specification with superimposition", in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD '02).

[255] Kellomaki, P., (2002), "Composing distributed systems from reusable aspects of behavior", in Proceedings of 2nd Int'l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS '02).

[256] Kendall, E.A., (1998), "Agent roles and aspects", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[257] Kendall, E.A., (1999), "Aspect-oriented programming for role models", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[258] Kendall, E.A., (1999), "Role model designs and implementations with aspect-oriented programming", ACM '99 (OOPSLA '99).

[259] Kendall, L., (2000), "Reengineering for separation of concerns", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[260] Kenens, P., Michiels, S., Matthijs, F., Robben, B., Truyen, E., Vanhaute, B., Joosen, W., and Verbaeten, P., (1998), "An AOP case with static and dynamic aspects", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[261] Kersten, M.A. and Murphy, G.C., (1999), "Atlas: A case study in building a web-based learning environment using aspect-oriented programming", in Proceedings of ACM Conference Object-Oriented Programming, Systems, Languages, and Applications, pgs. 340-352.

[262] Kiciman, E. and Fox, A., (2001), "Separation of concerns in networked service composition", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[263] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W.G., (2001), "An overview of aspectJ", ECOOP '01.

[264] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W.G., (2001), "Getting started with aspectj", *Communications of the ACM*, vol. 44, no. 10.

[265] Kiczales, G., Hugunin, J., Kersten, M., Lamping, J., Lopes, C., and Griswold, W.G., (2000), "Semantics-based crosscutting in AspectJ", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[266] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., and Irwin, J., (1997), "Aspect-oriented programming", in Aksit, M. and Matsuoka, S., editors, 11th European Conference Object-Oriented Programming, ol. 1241 of LNCS, pgs. 220-242. Springer Verlag.

[267] Kimelman, D., (1999), "Multidimensional tree-structured spaces for separation of concerns in software development environments", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[268] Kircher, M., Jain, P., and Corsaro, A., (2002), "XP + aop=better software", XP '02.

[269] Kishi, T. and Noda, N., (2000), "Aspect-oriented analysis for product line architecture", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[270] Kishi, T. and Noda, N., (2001), "Analyzing concerns used in analysis/design techniques", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[271] Kishi, T. and Noda, N., (2002), "Aspect-oriented analysis for architectural design", ACM '02 (IWPSE '01).

[272] Knudsen, J., (1999), "Aspect-oriented programming in BETA using the fragment system", in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP '99).

[273] Kruskal, V., (2000), "A blast from the past: Using P-EDIT for multidimensional editing", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[274] Kuloor, C. and Eberlein, A., (2003), "Aspect-oriented requirements engineering for software product lines", IEEE '03.

[275] Laddaga, R., Robertson, P., and Shrobe, H., (2001), “Aspects of the real-world”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[276] Lai, A. and Murphy, G.C., (1999), “The structure of features in java code: An exploratory investigation”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[277] Lai, A. and Murphy, G.C., (2001), “Capturing concerns with conceptual models”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[278] Lai, A., Murphy, G.C., and Walker, R.J., (2000), “Separating concerns with HyperJ: An experience report”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[279] Lammel, R., (1999), “Declarative aspect-oriented programming”, PEPM ’99.

[280] Lammel, R., (2002), “A semantical approach to method-call interception”, ACM ‘02 (AOSD ‘02).

[281] Lammel, R., Riedewald, G., and Lohmann, W., (1999), “Adaptation of functional object programs”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[282] Lammel, R., Visser, E., and Visser, J., (2003), “Strategic programming meets adaptive programming”, ACM ‘03 (AOSD ‘03).

[283] Lamping, J., (1997), “The interaction of components and aspects”, in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP ‘97).

[284] Lamping, J., (1999), “The role of the base in aspect-oriented programming”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[285] Li, H., Krishnamurthi, S., and Fisler, K., (2002), “Verifying cross-cutting features as open systems”, ACM ‘02 (SIGSOFT ‘02).

[286] Lieberherr, K., Lorenz, D.H., and Wu, P., (2003), “A case for statically executable advice: Checking the law of Demeter with aspectj”, ACM 2003 (AOSD ‘03).

[287] Lieberherr, K. and Orleans, D., (1997), “Preventive program maintenance in Demeter/java”, ACM ’97.

[288] Lieberherr, K., Orleans, D., and Ovlinger, J., (2001), “Aspect-oriented programming with adaptive methods”, *Communications of the ACM*, vol. 44, no. 10.

[289] Lions, J.M., Simoneau, D., Pitette, G., and Moussa, I., (2002), "Extending open tool/uml using metamodeling: An aspect oriented programming case study", in Proceedings of 2nd Int'l Workshop on Aspect Oriented Modeling with UML.

[290] Lippert, M. and Lopes, C., (1993), "A study on exception detection and handling using aspect-oriented programming", *Technical Report*, Xerox PARC.

[291] Lopes, C., Hilsdale, E., Hugunin, J., Kersten, M., and Kiczales, G., (2000), "Illustrations of crosscutting", ECOOP '00.

[292] Lopes, C., Hugunin, J., Kersten, M., Lippert, M., Hilsdale, E., and Kiczales, G., (2000), "Using aspectJ for programming the detection and handling of exceptions", ECOOP '00.

[293] Lopes, C.V. and Kiczales, G., (1998), "Recent developments in AspectJ", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[294] Lorenz, D.H., (1998), "Visitor beans: An aspect-oriented pattern", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[295] Loughran, N. and Rashid, A., (2002), "Mining aspects", in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD '02).

[296] Low, T., (2002), "Designing, modelling and implementing a toolkit for aspect-oriented tracing (TAST)", in Proceedings of Workshop on Aspect Oriented Programming Modeling with UML (AOSD '02).

[297] Ludy, T., (2002), "Aspect-oriented programming can lead to faster, cheaper, easier-to-use solutions", *Control Solutions*, vol. 11.

[298] Lunau, C.P., (1998), "Is composition of metaobjects = aspect oriented programming", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[299] Mahieu, T., Vanhaute, B., De Vlaminck, K., Janssens, G., and Joosen, W., (2000), "Using AOP to build complex data centric component frameworks", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[300] Mahrenholz, D., Spinczyk, O., Gal, A., and Schroder-Preikschat, W., (2002), "An aspect-oriented implementation of interrupt synchronization in the pure operating system family", in Proceedings of ECOOP '02.

[301] Mahrenholz, D., Spinczyk, O., Schroder-Preikschat, W., (2002), "Program instrumentation for debugging and monitoring with aspectC++", IEEE '02.

[302] Massoni, T., Sampaio, A., and Borba, P., (2001), “Progressive implementation of aspects”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[303] Masuhara, H., Kiczales, G., and Dutchyn, C., (2002), “Compilation semantics of aspect-oriented programs”, in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD ‘02).

[304] Matta, I., Eltoweissy, M., and Lieberherr, K., (1998), “From cscw applications to multicast routing: An integrated QoS architecture”, IEEE ICC ’98.

[305] Matthijs, F., Joosen, W., Vanhaute, B., Robben, B., and Verbaeten, P., (1997), “Aspects should not die”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘97).

[306] McDirmid, S., and Hsieh, W.C., (2003), “Aspect-oriented programming with jiazzi”, ACM 2003 (AOSD ‘03).

[307] Mehner, K. and Wagner, A., (1999), “On the role of method families in aspect-oriented programming”, in Proceedings of Int’l Workshop on Aspect Oriented Programming (ECOOP ‘99).

[308] Memmert, J., (2000), “Application development in java: From OOP to SOP”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[309] Memmert, J., (2001), “Advanced separation of concerns: Separation of concerns at the source”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[310] Memmert, J., (2002), “Designing with cosmos”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[311] Mendhekar, A., Kiczales, G., and Lamping, J., (1997). “RG: A case-study for aspect-oriented programming”, *Technical Report* SPL97-009 P9710044, Xerox Palo Alto Research Center.

[312] Mens, K., (2000), “Multiple cross-cutting architectural views”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[313] Mens, K., (2002), “Architectural aspects”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[314] Mens, K., Mens, T., and Wermelinger, M., (2002), “Maintaining software through intentional source-code views”, ACM ‘02 (SEKE ‘02).

[315] Mesquita, C., Barbosa, S., and de Lucena, C., (2002), “Towards the identification of concerns in personalization mechanisms via scenarios”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[316] Mezini, M., and Ostermann, K., (2002), “Integrating independent components with on-demand remodularization”, ACM ‘02 (OOPSLA ’02).

[317] Mikkonen, T., (2002), “On objects, aspects, and specifications addressing their collaboration”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[318] Mili, H., Dargham, J., and Bendelloul, S., (1999), “Separation of concerns and typing: A first stab”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[319] Mili, H., Mcheick, H., and Sadou, S., (2002), “Distributing objects with multiple aspects”, IEEE ‘02.

[320] Miller, S.K., (2001), “Aspect –oriented programming takes aim at software complexity”, *Technology News*, April.

[321] Monga, M., (2000), “Ad-hoc constructs for non-functional aspects”, WSDAAL ’00.

[322] Monga, M., (2000), “Concern-specific aspect-oriented programming with malaj”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE 2000).

[323] Moreira, A., Araujo, J., and Brito, I., (2002), “Crosscutting quality attributes for requirements engineering”, ACM 2002 (SEKE 2002).

[324] Mousavi, M., Russello, G., Chaudron, M., Reniers, M.A., Basten, T., Corsaro, A., Shukla, S., Gupta, R., and Schmidt, D., (2002), “Aspects + GAMMA = AspectGAMMA: A formal framework for aspect-oriented specification”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[325] Mousavi, M., Russello, G., Chaudron, M., Reniers, M.A., Basten, T., Corsaro, A., Shukla, S., Gupta, R., and Schmidt, D., (2002), “Using aspect-gamma in the design of embedded systems”, IEEE ’02.

[326] Mraidha, C. and Benzakki, J., (2003), “A two-aspect approach for a clearer behaviour model”, IEEE ‘03.

[327] Muller, J.K., (1997), "Aspect-design in the building-block method", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '97).

[328] Murphy, G.C., Lai, A., Walker, R.J., and Robillard, M.P., (2001), "Separating features in source code: An exploratory study", in Proceedings of 23rd Int'l Conference Software Engineering, pgs. 275-284. IEEE Computer Society.

[329] Murphy, G.C., Walker, R.J., and Baniassad, E.L.A., (1999), "Evaluating emerging software development technologies: Lessons learned from assessing aspect-oriented programming", *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pgs. 438-455.

[330] Murphy, G.C., Walker, R.J., Banissad, E.L.A., Robillard, M.P., Lai, A., and Kersten, M.A., (2002), "Does aspect-oriented programming work?", *Communications of the ACM*, vol. 44, no.10.

[331] Nakajima, S., (1999), "Separation of concerns in early stage of framework development", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[332] Navasa, A., Perez, M.A., and Murillo, J.M., (2001), "Developing component based systems using AOP concepts", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[333] Navasa, A., Perez, M.A., Murillo, J.M., and Hernandez, J., (2002), "Aspect-oriented software architecture: A structural perspective", in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architectural Design (AOSD '02).

[334] Nebbe, R.D., (1998), "Coordination and composition: The two paradigms underlying AOP?", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[335] Nelson, T., Alencar, P., and Cowan, D., (2000), "Verifying multiple-perspective composition", in Proceedings of Workshop on Advanced Separation of Concerns OOPSLA '00).

[336] Nelson, T., Cowan, D., and Alencar, P., (1999), "Towards a formal model of object-oriented hyperslices", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[337] Nelson, T., Cowan, D., and Alencar, P., (2001), "Formal verification of a bounded buffer with three separate concerns", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[338] Nentwich, C., Emmerich, W., and Finkelstei, A., (2002), "Edit, compile, debug-From hacking to distributed engineering", in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD '02).

[339] Netinant, P., Constantinides, C.A., Elrad, T., and Fayad, M.E., (2000), "Aspect-oriented frameworks: The design of adaptable operating systems", ACM '00 (OOPSLA '00).

[340] Netinant, P., Constantinides, C.A., Elrad, T., Fayad, M.E., and Bader, A., (2002), "Supporting the design of adaptable operating systems using aspect-oriented frameworks."

[341] Netinant, P., Elrad, T., and Fayad, M.E., (2001), "A layered approach: To building open aspect-oriented systems", *Communications of the ACM*, vol. 44, no. 10.

[342] Newman, E., (2001), "Localizing views for separation of concerns", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[343] Noda, N. and Kishi, T., (1998), "On aspect-oriented design: An approach to designing quality attributes", IEEE '98.

[344] Noda, N. and Kishi, T., (1999), "On aspect-oriented design-Applying 'multidimensional separation of concerns' in designing quality attributes", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[345] Noda, N. and Kishi, T., (2001), "Implementing design patterns using advanced separation of concerns", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[346] Nordberg, M.E.III., (2001), "Aspect-oriented dependency inversion", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[347] Noro, M. and Kumazaki, A., (2002), "On aspect-oriented software architecture: It implies a process as well as a product", IEEE '02.

[348] Odgers, B. and Thompson, S., (1999), "Aspect-oriented process engineering", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[349] Orleans, D., (2001), "Separating behavioral concerns with predicate dispatch, or, if statement considered harmful", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[350] Orleans, D., (2002), "Incremental programming with extensible decisions", ACM '02 (AOSD '02).

[351] Ortin, F. and Cueva, J.M., (2002), "Implementing a real computational-environment jump in order to develop a runtime-adaptable reflective platform", *ACM Sigplan Notices,* vol. 37, no. 8.

[352] Ossher, H. and Tarr, P., (1998), "Operation-level composition: A case in (join) point", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[353] Ossher, H. and Tarr, P., (2000), “Multi-dimensional separation of concerns in hyperspace: Position paper”, in Proceedings of Symposium on Software Architectures and Component Technology: The State of the Art in Software Development.

[354] Ossher, H. and Tarr, P., (2000), “On the need for on-demand remodularization”, ECOOP ’00.

[355] Ossher, H. and Tarr, P., (2001), “Hyper/J: Multi-dimensional separation of concerns for java”, IEEE ‘01.

[356] Ossher, H. and Tarr, P., (2001), “Some micro-reuse challenges”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[357] Ossher, H. and Tarr, P., (2001), “Using multidimensional separation of concerns to (re)shape evolving software”, *Communications of the ACM*, vol. 44, no.10.

[358] Ostermann, K. and Mezini, M., (2001), “Object-oriented composition is tangled”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[359] Ovlinger, J., (2003), “From aspect-oriented model to implementation”, AOSD ’03.

[360] Pace, J.A.D., Trilnik, F., and Campo, M.R., (2000), “How to handle interacting concerns?”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[361] Palsberg, J., (1997), “Class-graph inference for adaptive programs”, *Theory & Practice of Object Systems*, vol. 3, no. 2.

[362] Pang, J. and Blair, L., (2002), “An adaptive run time manager for the dynamic integration and interaction resolution of features”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02).

[363] Park, S.H., Park, S.H., and Baik, D.K., (1997), “Platform independent class repository of tmn in personal communication network using entity-aspect oriented programming”, IEEE ‘97 (RTCSA ’97).

[364] Pawlak, R., Duchien, L., Florin, G., Legond-Aubry, F., Seinturier, L., and Martelli, L., (2002), “A UML notation for aspect-oriented software design”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[365] Pawlak, R., Duchien, L., Florin, G., Martelli, L., and Seinturier, L., (2000), “Distributed separation of concerns with aspect components”, IEEE ‘00.

[366] Pawlak, R., Duchien, L., and Seinturier, L., (2001), “Dynamic wrappers: Handling the composition issue with jac”, IEEE ‘01.

[367] Pazzi, L., (1999), “Explicit aspect composition by part-whole state charts”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[368] Pike, S.M., (2001), “Binary trees: A challenge problem for separating concerns”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[369] Pinto, M., Amor, M., Fuentes, L., and Troya, J.M., (2001), “Collaborative virtual environment development: An aspect-oriented approach”, IEEE ‘01.

[370] Pinto, M., Amor, M., Fuentes, L., and Troya, J.M., (2001), “Run-time coordination of components: Design patterns vs. component-aspect based platforms”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[371] Pinto, M., Fuentes, L., Fayad, M.E., and Troya, J.M., (2002), “Separation of coordination in a dynamic aspect oriented framework”, ACM 2002 (AOSD ‘02).

[372] Pinto, M., Fuentes, L., and Troya, J.M., (2001), “Towards an aspect-oriented framework in the design of collaborative virtual environments”, IEEE ‘01.

[373] Piveta, E.K. and Devegil, A.J., (2002), “Aspects in the rational unified process analysis and design workflow”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[374] Piveta, E.K. and Zancanella, L.C., (2001), “Aurelia: Aspect oriented programming using a reflective approach”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[375] Popovici, A., Alonso, G., Gross, T., (2001), “AOP support for mobile systems”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[376] Popovici, A., Gross, T., and Alonso, G., (2001), “Dynamic homogenous aop with prose”, *Technical Report*.

[377] Popovici, A., Gross, T., and Alonso, G., (2002), “Dynamic weaving for aspect-oriented programming”, ACM ‘02 (AOSD ’02).

[378] Prasad, M.D. and Chaudhary, B.D., (2003), “Aop support for C#”, AOSD ’03.

[379] Pratap, R., Cytron, R.K., Sharp, D., and Pla, E., (2003), “Transport layer abstraction in event channels for embedded systems”, ACM ’03, (LCTES ‘03).

[380] Prehofer, C., (2002), “Feature interactions in statechart diagrams or graphical composition of components”, in Proceedings of UML ‘02.

[381] Prehofer, C., (2002), “Graphical composition of components with feature interactions”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[382] Pryor, J. and Bastan, N., (1999), “A reflective architecture for the support of aspect-oriented programming in smalltalk”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP ‘99).

[383] Pulvermuller, E., Speck., A., and Rashid, A., (2000), “Implementing collaboration-based designs using aspect-oriented programming”, IEEE ‘00.

[384] Putrycz, E. and Bernard, G., (2002), “Using aspect oriented programming to build a portable load balancing service”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02).

[385] Quintero, A.M and Valderrama, J.T., (2002), “Separating the navigational aspect”, IEEE ‘02.

[386] Raje, R.R., Zhong, M., and Wang, T., (2001), “Case study: A distributed concurrent system with aspectJ”, ACM ’01.

[387] Rashid, A., (2002), “Weaving aspects in a persistent environment”, *ACM Sigplan Notices*, vol. 37, no. 2.

[388] Rashid, A. and Chitchyan, R., (2003), “Persistence as an aspect”, ACM 2003 (AOSD ‘03).

[389] Rashid, A. and Kotonya, G., (2001), “Risk management in component-based development: A separation of concerns perspective”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[390] Rashid, A. and Loughram, N., (2002), “Relational database support for aspect-oriented programming”, NODE ’02.

[391] Rashid, A., Moreira, A., and Araujo, J., (2003), “Modularisation and composition of aspectual requirements”, ACM ‘03 (AOSD ‘03).

[392] Rashid, A. and Sawyer, P., (2000), “Object database evolution using separation of concerns”, *ACM SIGMOD Record*, vol. 29, no. 4.

[393] Rashid, A., Sawyer, P., Moreira, A., and Araujo, J., (2002), “Early aspects: A model for aspect-oriented requirements engineering”, IEEE ‘02.

[394] Reis, R.Q., Reis, C.A.L., Schlebbe, H., and Nunes, D.J., (2002), "Towards an aspect-oriented approach to improve the reusability of software process models", in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD '02).

[395] Robben, B. and Steyaert, P., (2000), "Aspects on tv", ECOOP '00.

[396] Robillard, M.P. and Murphy, G.C., (1999), "Migrating a static analysis tool to AspectJ", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[397] Robillard, M.P. and Murphy, G.C., (2001), "Analyzing concerns using class member dependencies", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[398] Roman, M., Kon, F., and Campbell, R.H., (1999), "Design and implementation of runtime reflection in communication middleware: The dynamicTAO case", ICDCS '99.

[399] Roudier, Y. and Ichisugi, Y., (1998), "Mixin composition strategies for the modular implementation of aspect weaving", in Proceedings of Int'l Workshop on Aspect Oriented Programming (ICSE '98).

[400] Rouvellou, I., Sutton Jr., S.M., and Tai, S., (2000), "Multidimensional separation of concerns in middleware", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[401] Sanchez, F., Hernandez, J., Murillo, J.M., and Pedarza, E., (1998), "Run-time adaptability of synchronization policies in concurrent object-oriented languages", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[402] Sapir, N., Tyszberowicz, S., and Yehudai, A., (2002), "Extending uml with aspect usage constraints in the analysis and design phases", in Proceedings of AOSD '02.

[403] Sassen, A., Amoros, G., Donth, P., Geihs, K., Jezequel, J., Odent, K., Plouzeau, N., and Weis, T., (2002), "QCCS: A methodology for the development of contract-aware components based on aspect-oriented design", in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD '02).

[404] Savolainen, J., (2000), "Improving product line development with subject-oriented programming", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[405] Savolainen, J., (2000), "Towards multi-dimensional methods", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[406] Sazawal, V., (2001), "Separation of concerns for ubiquitous computing", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[407] Schmidmeier, A., Hanenberg, S., and Unland, R., (2003), "Implementing known concepts in aspectJ", in Proceedings of 3rd International German Workshop on Aspect-oriented Software Development of the SIG Object-oriented software development.

[408] Schmidt, R. and Assmann, U., (1998), "Extending aspect-oriented programming in order to flexibly support workflows", in Proceedings of Int'l Workshop on Aspect Oriented Programming (ICSE '98).

[409] Schonger, S., Pulvermuller, E., and Sarstedt, S., (2002), "Aspect-oriented programming and component weaving: Using xml representations of abstract syntax trees", in Proceedings of Workshop on Aspect-oriented Software Development.

[410] Schult, W. and Polze, A., (2002), "Aspect-oriented programming with c# and .net", IEEE '02.

[411] Schult, W. and Polze, A., (2002), "Speed vs memory usage: An approach to deal with contrary aspects", ACM '02.

[412] Seinturier, L., (1999), "JST: An object synchronization aspect for java", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[413] Seiter, L., Mezini, M., and Lieberherr, K., (1999), "Dyanmic component gluing", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[414] Sereni, D., and de Moor, O., (2003), "Static analysis of aspects", ACM '03.

[415] Shah, V. and Hill, F., (2003), "An aspect-oriented security framework", IEEE '03.

[416] Shomrat, M. and Yehudai, A., (2002), "Obvious or not? Regulating architectural decisions using aspect-oriented programming", ACM '02, (AOSD '02).

[417] Sihman, M. and Katz, S., (2002), "A calculus of superimpositions for distributed systems", ACM '02 (AOSD '02).

[418] Silva, A.R., (1999), "Separation and composition of overlapping and interacting concerns", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[419] Simmonds, I. and Ing, D., (2000), "Clues in the search for ever more valuable separation of concern", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[420] Skipper, M., (1999), “The watson subject compiler and aspectj (A critique of practical objects)”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA ‘99).

[421] Skipper, M., (2000), “A model of composition oriented programming”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[422] Skipper, M., (2001), “Semantics of an object-oriented language with aspects and advice”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[423] Skipper, M. and Drossopoulou, S., (1999), “Formalising composition-oriented programming”, in Proceedings of ECOOP ‘99.

[424] Soares, S. and Borba, P., (2002), “PaDA: A pattern for distribution aspects”, in Proceedings of the Second Latin American Conference on Pattern Languages of Programming-Sugarloaf PLOP ’02.

[425] Soares, S., Laureano, E., and Borba, P., (2002), “Implementing distribution and persistence aspects with aspectJ”, ACM ’02, (OOPSLA ‘02).

[426] Spinczyk, O., Gal, A., and Schroder-Preikschat, W., (2002), “Aspect C++: An aspect-oriented extension to the C++ programming language”, in Proceedings of 40th International Conference on Technology of Object-Oriented Languages and Systems.

[427] Stearns, M. and Piccinelli, G., (2002), “Managing interaction concerns in web-service systems”, in Proceedings of ICDCS ‘02.

[428] Stein, D., Hanenberg, S., and Unland, R., (2002), “A uml-based aspect-oriented design notation for aspectj”, ACM ’02, (AOSD ’02).

[429] Stein, D., Hanenberg, S., and Unland, R., (2002), “Designing aspect-oriented crosscutting in uml”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ‘02).

[430] Stein, D., Hanenberg, S., and Unland, R., (2002), “On representing join points in the uml”, in Proceedings of AOSD ‘02.

[431] Stein, D., Hanenberg, S., and Unland, R., (2003), “Position paper on aspect-oriented modeling: Issues on representing crosscutting features”, in Proceedings of 3rd International Workshop on Aspect Oriented Modeling with UML.

[432] Stojanovic, Z. and Dahanayak, A., (2002), “Components and viewpoints as integrated separations of concerns in system designing”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[433] Storzer, M., (2003), “Analysis of aspectJ programs”, AOSD-GI ’03.

[434] Storzer, M. and Krinke, J., (2003), “Interference analysis for aspectJ”, FOAL ’03.

[435] Storzer, M., Krinke, J., and Breu, S., (2003), “Trace analysis for aspect application”, in Proceedings of Analysis of Aspect-oriented Software (AAOS ’03).

[436] Sullivan, G.T., (2001), “Aspect-oriented programming using reflection”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[437] Sullivan, K., Go, L., and Cai, Y., (2002), “Non-modularity in aspect-oriented languages: Integration as a crosscutting concern for aspectJ”, ACM 2002 (AOSD ‘02).

[438] Sunetnanta, T. and Finkelstein, A., (2001), “Automated consistency checking for multiperspective software specifications”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[439] Sutton, Jr., S.M., (1999), “Multiple dimensions of concern in software testing”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns OOPSLA ‘99).

[440] Sutton, Jr., S.M., (2002), “Early-stage concern modeling”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design (AOSD ‘02).

[441] Sutton, Jr., S.M. and Rouvellou, I., (2000), “Concerns in the design of a software cache”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[442] Sutton, Jr., S.M. and Rouvellou, I., (2001), “Applicability of categorization theory to multidimensional separation of concerns”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[443] Sutton, Jr., S.M. and Rouvellou, I., (2001),”Issues in the design and implementation of a concern-space modeling schema”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[444] Sutton, Jr., S.M. and Tarr, P., (2002),”Aspect-oriented design needs concern modeling”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[445] Suvee, D., Vanderperren, W., and Jonckers, V., (2003), “JasCo: An aspect-oriented approach tailored for component based software development”, ACM 2003 (AOSD ‘03).

[446] Suzuki, J. and Yamamoto, Y., (199), "Extending uml with aspects: Aspect support in the design phase", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[447] Tarr, P., Ossher, H., and Henkel, J., (2001), "Visualization as an aid to compositional software engineering", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[448] Tekinerdogan, B. and Aksit, M., (1998), "Deriving design aspects from canonical models", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[449] Tekinerdogan, B. and Aksit, M., (2000), "Separation and composition of concerns through synthesis-based design", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[450] Thai, J., Pekilis, B., Lau, A., and Seviora, R., (2001), "Aspect-oriented implementation of software health indicators", IEEE '01.

[451] Thorup, K.K., (1997), "Contextual class extensions", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '97).

[452] Tip, F. and Ryder, B., (2000), "Can program slicing be used for separation of concerns?", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[453] Top, S., Jorgensen, B., Thybo, C., and Thusgaard, P., (2001), "Meta-level architectures for fault tolerant control (ftc) in embedded software systems", in Proceedings of ECOOP '01.

[454] Top, S., Norregaard, B., and Angelov, C., (2002), "Separation of fault tolerant control concern in embedded control system", in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD '02).

[455] Tourwe, T., Brichau, J., and Gybels, K., (2003), "On the existence of the aosd-evolution paradox", AOSD '03.

[456] Trilnik, F., Pace, A.D., and Campo, M., (2002), "Smartweaver: An agent-based approach for aspect-oriented development", ACM '02 (ICSE '02).

[457] Truyen, E. and Joosen, W., (2001), "Customization of on-line services with simultaneous client-specific views", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[458] Truyen, E., Jorgensen, B.N., Joosen, W., and Verbaeten, P., (2000), "Aspects for run-time component integration", ECOOP '00.

[459] Tucker, D.B. and Krishnamurthi, S., (2003), "Pointcuts and advice in higher-order languages", ACM '03, (AOSD '03).

[460] Ubayashi, N. and Tamai, T., (2002), "Aspect-oriented programming with model checking", ACM '02.

[461] Ungar, D., (1999), "The limits to factoring", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[462] Vanhaute, B., De Win, B., and De Decker, B., (2001), "Building frameworks in aspectJ", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[463] Vanhaute, B., Truyen, E., Joosen, W., and Verbaeten, P., (1999), "Composing non-orthogonal meta-programs", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns (OOPSLA '99).

[464] VanHilst, M., (1997), "Subcomponent decomposition as a form of aspect oriented programming", in Proceedings of ECOOP '97.

[465] Van Roy, P., Haridi, S., Brand, P., Smolka, G., Mehl, M., and Scheidhauer, R., (1997), "Using mobility to make transparent distribution practical", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '97).

[466] Viega, J., (2000), "Separation of concerns for security", in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE '00).

[467] Viega, J. and Voas, J., (2000), "Can aspect-oriented programming lead to more reliable software?", IEEE '00.

[468] Wagelaar, D. and Bergmans, L., (2002), "Using a concept-based approach to aspect-oriented software design", in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD '02).

[469] Walker, R.J., Baniassad, and Murphy, G., (1998), "Assessing aspect-oriented programming and design", in Proceedings of Workshop on Aspect Oriented Programming (ECOOP '98).

[470] Walker, R.J., Baniassad, E.L.A., and Murphy, G.C., (1999), "An initial assessment of aspect-oriented programming", in Proceedings of 21st Int'l Conference Software Engineering (ICSE '99).

[471] Walker, R.J. and Murphy, G.C., (2001), "Joinpoints as ordered events: Towards applying implicit context to aspect-orientation", in Proceedings of Workshop on Advanced Separation of Concerns at the 23rd International Conference on Software Engineering.

[472] Wand, M., Kiczales, G., and Dutchyn, C., (2002), "A semantics for advice and dynamic join points in aspect-oriented programming", in Proceedings of FOAL '02: Foundations of Aspect-Oriented Languages (AOSD '02).

[473] Wegener, H. and Rida, A., (2000), "Reenginnering of metalevel abstractions with data mining methods", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[474] Welch, I. and Stroud, R., (1999), "Load-time application of aspects to java cots software", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[475] Wermelinger, M., Fiadeiro, J.L., Andrade, L., Koutsoukos, G., and Gouveia, J., (2001), "Separation of core concerns: Computation, coordination, and configuration", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[476] Wijnstra, J.G., (2001), "Quality attributes and aspects of a medical product family", IEEE '01.

[477] Willink, E. and Muchnick, V., (1999), "Weaving a way past the c++ one definition rule", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[478] Wohlstadter, E. and Devanbu, P., (2001), "A lazy approach to separating architectural concerns", in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE '01).

[479] Wohlstadter, E., Keen, A., Jackson, S., and Devanbu, P., (2001), "Accommodating evolution in aspectJ", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[480] Wohlstadter, E., Keen, A., Jackson, S., and Devanbu, P., (2003), DADO: Enhancing middleware to support crosscutting features in distributed, heterogeneous systems", IEEE '03.

[481] Xie, X. and Shatz, S.M., (2000), "An approach to using formal methods in aspect orientation", in Proceedings of the Int'l Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '00).

[482] Yang, Z., Cheng, B.H.C., Stirewalt, R.E.K., Sowell, J., Sadjadi, S.M., and McKinley, P.K., (2002), "An aspect-oriented approach to dynamic adaptation", ACM '02 (WOSS '02).

[483] Yokomori, R., Ishio, T., Yamamoto, T., Matsushita, M., Kusumoto, S., and Inoue, K., (2003), "Java program analysis projects in osaka university: Aspect-based slicing system adas and ranked-component search system spars-j", IEEE '03.

[484] Zakaria, A.A., Hosny, H., and Zeid, A., (2002), “A uml extension for modeling aspect-oriented systems”, in Proceedings of AOSD ‘02.

[485] Zdun, U., (2003), “Using structure and dependency tracing patterns for aspect composition”, in Proceedings of 3rd Workshop on Aspect-Oriented Software Development.

[486] Zhang, H., Jarzabek, S., and Swe, S.M., (2001), “X-frame approach for handling variants within concerns”, in Proceedings of Workshop on Advanced Separation of Concerns in Software Engineering (ICSE ‘01).

[487] Zhang, C. and Jacobsen, H.A., (2003), “Quantifying aspects in middleware platforms”, ACM ’03, (AOSD ‘03).

[488] Zhao, J., (2002), “Change impact analysis for aspect-oriented software evolution”, ACM ‘02.

[489] Zhao, J., (2002), “Slicing aspect-oriented software”, IEEE ‘02.

[490] Zhao, J., (2002), “Tool support for unit testing of aspect-oriented software”, OOPSLA ’02.

[491] Zhao, J., (2002), “Towards a metrics suite for aspect-oriented software”, *Technical Report SE-136-25, Information Processing Society of Japan (IPSJ)*.

[492] Zhao, J and Rinard, M., (2003), “Pipa: A behavioral interface specification language for aspectJ”, FASE ’03.

[493] Zhao, J. and Rinard, M., (2003), “System dependence graph construction for aspect-oriented programs”, MIT-LCS-TR-891.

[494] Zinky, J., Shapiro, R., Loyall, J., Pal, P., and Atighetchi, M., (2001), “Separation of concerns for resuse of systemic adaptation in quo 3.0”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

APPENDIX B

Primary Dimension Articles and Categories

Sub-category	Article Number
1.A	62
1.B	444, 187, 76
1.C	315, 153, 49
1.D	486, 475, 392, 360, 349, 331, 309, 188, 177, 129, 115, 32, 7
1.E	437, 291, 265, 45, 27
2.B	494, 401, 362, 357, 351, 340, 339, 288, 287, 116, 2
2.C	482, 304
3.B	455
3.C	144
3.D	284, 214, 158, 119, 63
4.A	470, 469, 386, 328, 311, 261, 125, 56
4.B	483, 459, 425, 420, 396, 276, 236, 235, 220, 218, 143, 50, 40, 22
4.C	456, 450, 414, 294, 292, 259, 207, 41, 17
4.D	395
4.E	300, 299, 237, 142, 138, 135, 102, 101, 66
5.A	411, 290
5.B	412, 233, 232, 55, 13, 12
5.C	356, 203, 47
5.D	466, 462, 415, 181, 171
6.A	208, 160, 157, 34
6.B	487, 480, 384, 375, 365, 262, 213, 179, 68, 42
6.C	457, 427, 385, 146

Sub-category	Article Number
6.D	372, 369, 173, 134, 120, 54
6.E	255, 180
6.F	251, 176
7.A	421
7.B	481, 423, 422, 410, 324, 281, 254, 216, 191, 183, 170, 149, 145, 14, 5
7.C	493, 492, 489, 474, 464, 460, 436, 409, 390, 343, 335, 327, 310, 307, 224, 205, 189, 178, 175, 132, 130, 122, 121, 114, 108, 89, 86, 67, 38, 36, 26, 25, 20, 1
7.D	472, 433, 342, 334, 303, 279, 253, 223, 204, 202, 166, 127, 104, 98, 52, 28
8.A	243, 242, 240, 230, 10, 9
8.B	231, 229
8.C	484, 446, 428, 429, 402, 364, 302, 289, 217, 196, 192
8.D	431, 430, 367, 326, 256, 248, 212, 139, 78, 29, 8
8.E	359, 252, 37
9.A	416, 271, 250, 244, 161, 126
9.B	99
9.C	458, 453, 432, 382, 370, 347, 333, 332, 317, 313, 312, 298, 269, 249, 35, 21
10.B	393, 391, 274, 194, 113, 94, 51, 16
10.C	491, 488, 435, 434, 394, 373, 348, 323, 286, 228
10.D	468, 405, 403, 366, 346, 344, 341, 325, 322, 314, 239, 227, 210, 198, 197, 154, 111, 105, 103, 77, 75, 48
10.E	490, 301, 296, 215
10.F	479, 445, 426, 378, 376, 374, 306, 293, 272, 264, 263, 226, 225, 97, 95, 82, 58, 39
11.A	320, 297, 266, 159, 152

Sub-category	Article Number
11.B	467, 330, 329, 222, 164, 162, 151
11.C	408, 383, 295, 268, 221, 174, 163, 64, 53
11.D	371, 112, 110, 109
11.E	201
12.A	321, 260, 118, 100
12.B	448, 165, 141, 30
12.C	305, 136, 70
12.D	283
12.E	424, 258, 257, 156
12.F	388, 23
13.A	377, 133, 60, 57, 3
13.B	471, 80
13.C	150, 24, 19
13.D	485, 399, 387, 186, 124
14.C	476, 454, 442, 419, 400, 389, 368, 336, 267, 246, 238, 206, 200, 190, 172, 117, 81, 71
14.D	451, 413, 404, 363, 355, 350, 211, 123, 72, 59
14.E	465, 417, 398, 379, 247, 195, 182, 128, 69
14.F	361
14.G	439, 438, 397, 354, 338, 316, 308, 285, 280, 241, 234, 193, 148, 140, 137, 107, 91, 85, 84, 65, 43, 18
14.H	473, 278, 273, 219, 11
14.I	478
14.J	61

Sub-category	Article Number
14.K	461, 452, 449, 406, 358, 199, 184, 169, 155, 106, 83, 79, 74, 46, 15
14.M	381, 380, 245, 209, 168, 147, 88
14.N	477, 463
15.B	447, 352, 319, 282, 275, 185, 167, 96, 92, 87, 73, 44, 33, 31, 6, 4
15.C	443, 441, 440, 337
15.D	418, 353, 345, 318, 270, 131, 90
15.E	277
15.F	407
15.G	93

APPENDIX C

Secondary Dimension Articles and Categories

Sub-category	Article Number
1.A	306, 276, 223, 322, 313
1.B	443, 440, 337
1.D	86, 325, 319, 339, 249, 318, 326, 418, 466, 21, 432, 366, 35, 344, 48, 396, 385, 365, 357, 353, 427, 424, 244, 218, 111, 259, 270, 262, 131, 114, 239, 33
1.E	173, 362, 214, 455, 407, 63, 431, 429, 340, 42, 101, 44, 300, 480, 78, 146, 93, 145, 314, 312, 415
2.B	392, 115
2.C	50, 375
4.A	76, 100, 329
4.B	260, 486, 289, 310
4.C	421, 12, 27, 447, 269, 55, 188, 26
4.D	453
4.E	364, 233, 67, 70, 13, 129, 458, 462, 105, 2, 347
5.B	401
5.C	4, 132, 394, 38, 307, 179
5.D	208
6.B	66, 236
6.C	261, 99
6.D	158, 386, 212, 494, 143
6.E	237, 36, 425
7.A	444

Sub-category	Article Number
7.B	32
7.D	171, 482, 45
8.A	403
8.C	250
8.D	323, 287, 384, 92, 87, 457, 456, 16
9.A	183, 122
9.B	113, 304
9.C	20, 251, 487
10.A	217, 9, 10, 230
10.B	243, 254, 176
10.C	460, 189
10.D	446, 441, 359, 351, 343, 345, 333, 315, 120, 96, 90
10.F	277, 175, 242, 216, 139, 196, 130, 192, 177, 17, 321, 89, 377, 402, 410, 19, 485, 484, 489, 493, 294, 492
11.B	170
11.C	428, 436
12.A	62, 295, 367, 57, 387
12.B	138, 51, 412, 37, 240, 29, 5, 191, 299, 252
12.C	103, 413, 135, 253, 56
12.D	437, 411, 47, 327, 481, 284
12.E	224, 142, 220, 331, 178, 213, 49, 324
12.F	302, 116, 119, 408, 153, 181, 180, 157, 68, 360, 474, 298, 232
13.B	430, 118, 303
13.C	231, 229

Sub-category	Article Number
13.D	472, 8
14.C	155, 85, 91, 11, 439, 278, 182, 15, 84, 106, 449, 452, 273, 168, 59, 69, 43, 245, 354, 46, 355, 413, 241, 219, 397, 83, 406, 79
14.D	246
14.E	169, 363, 308
14.F	247, 128, 419
14.G	200, 389, 211, 209, 195, 451, 267
14.H	398, 140, 72, 404
14.I	473, 193
14.K	147
14.L	148, 74
14.M	65, 358, 234, 438
15.B	97, 144, 121, 28, 75, 41, 108, 94, 423, 1, 342, 409, 248, 422, 210, 205, 464, 420, 309
15.D	154, 150, 328, 124

APPENDIX D

Tertiary Dimension Articles and Categories

Sub-category	Article Number
1.B	441
1.C	323
1.D	345, 342, 189, 216, 287, 420, 310, 240, 421, 90, 113
1.E	243, 26, 96, 94, 87, 16
3.D	466
4.A	427
4.E	214, 176, 480
6.D	418, 482
6.E	13
8.D	486, 261, 17, 339, 340
9.A	179
9.C	129
10.B	324
10.D	384, 183, 487
10.F	447, 364, 36, 114, 119, 474, 33, 38, 86, 409
11.C	139, 78
12.A	223, 307
12.B	365
12.E	42, 218
12.F	28, 432, 66, 396, 425
13.A	367

Sub-category	Article Number
13.B	63, 47, 143
13.D	56
14.B	72, 106
14.C	473, 247, 234, 169
14.D	168, 308
15.B	45, 357, 76
15.D	177

APPENDIX E

Quaternary Dimension Articles and Categories

Sub-category	Article Number
1.D	17
4.B	365
4.C	63
4.E	177
10.D	287
12.C	143
13.A	409
13.B	28
15.B	421, 216

APPENDIX F

Some Recommended AOP Literature

[1] Berger, L., (2000), "Junction point aspect: A solution to simplify implementation of aspect languages and dynamic management of aspect programs", ECOOP '00.

[2] Berger, L., Dery, A.M., and Fornarino, M., (1998), "Interactions between objects: An aspect of object-oriented languages", in Proceedings of Int'l Workshop on Aspect Oriented Programming (ICSE '98).

[3] Bergmans, L. and Aksit, M., (2001), "How to deal with encapsulation in aspect-orientation", in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA '01).

[4] Bollert, K., (1999), "On weaving aspects", in Proceedings of Int'l Workshop on Aspect-Oriented Programming (ECOOP '99).

[5] Brichau, J., Mens, K., and De Volder, K., (2002), "Building composable aspect-specific languages with logic metaprogramming", GPCE '02.

[6] Chavez, C. and Lucena, C.J., (2001), "Design-level support for aspect-oriented software development", in Proceedings of OOPSLA '01.

[7] Chavez, C. and Lucena, C., (2002), "A metamodel for aspect-oriented modeling", in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD '02).

[8] Clarke, S. and Walker, R.J., (2002), "Towards a standard design language for AOSD", ACM 2002, (AOSD '02).

[9] Clifton, C. and Leavens, G.T., (2002), "Observers and assistants: A proposal for modular aspect-oriented reasoning", in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD '02).

[10] Coady, Y., Brodsky, A., Brodsky, D., Pomkoski, J., Gudmundson, S., Ong, J.S., and Kiczales, G., (2000), "Can AOP support extensibility in client-server architectures?", in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP '01).

[11] Coady, Y., Kiczales, G., and Feeley, M., (2000), "Exploring an aspect-oriented approach to operating system code", in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA '00).

[12] Constantinides, C., Bader, A., and Elrad, T., (1999), “An aspect-oriented design framework for concurrent systems”, in Proceedings of Int’l Workshop on Aspect-Oriented Programming (ECOOP 1999).

[13] Constantinides, C.A., Bader, A., and Elrad, T., (2000), “An aspect-oriented design framework”, *ACM Computing Surveys*, March 2000.

[14] Constantinides, C.A., Bader, A., and Elrad, T., Fayad, M.E., and Netinant, P., (2000), “Designing an aspect-oriented framework in an object-oriented environment”, ACM ‘00.

[15] Costanza, P., Kniesel, G., and Austermann, M., (2001), “Independent extensibility for aspect-oriented systems”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[16] Filman, R.E., (2000), “Applying aspect-oriented programming to intelligent synthesis”, in Proceedings of Workshop on Aspects and Dimensions of Concerns (ECOOP ‘00).

[17] Filman, R.E., (2001), “What is aspect-oriented programming, revisited”, in Proceedings of Workshop on Advanced Separation of Concerns (ECOOP ‘01).

[18] Filman, R.E. and Friedman, D.P., (2000), “Aspect-oriented programming is quantification and obliviousness”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[19] Fradet, P. and Sudholt, M., (1998), “AOP: Towards a generic framework using program transformation and analysis”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[20] Furfaro, A., Nigro, L., and Pupo, F., (2002), “Aspect oriented programming using actors”, in Proceedings of 2nd Int’l Workshop on Aspect Oriented Programming for Distributed Computing Systems (ICDCS ‘02).

[21] Georg, G., Ray, I., and France, R., (2002), “Using aspects to design a secure system”, IEEE ‘02.

[22] Giese, H. and Vilbig, A, (2000), “Towards aspect-oriented design and architecture”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[23] Groher, I. and Schulze, S., (2003), “Generating aspect code from uml models”, in Proceedings of Workshop on Aspect-Oriented Modeling with UML (AOSD ’03).

[24] Gschwind, T. and Oberleitner, J., (2003), “Improving dynamic data analysis with aspect-oriented programming”, IEEE ‘03.

[25] Hanenberg, S. and Unland, R., (2001), “Concerning aop and inheritance”, Technical Report, tr-ri-01-223.

[26] Hirschfield, R., Osterbye, K., and Wagner, M., (2003), “System integration using aop”, AOSD ’03.

[27] Ho, W.M., Pennaneach, F., Jezequel, M., and Plouzeau, N., (2000), “Aspect-oriented design with the UML”, in Proceedings of Workshop on Multi-Dimensional Separation of Concerns in Software Engineering (ICSE ‘00).

[28] Kenens, P., Michiels, S., Matthijs, F., Robben, B., Truyen, E., Vanhaute, B., Joosen, W., and Verbaeten, P., (1998), “An AOP case with static and dynamic aspects”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[29] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., and Irwin, J., (1997), “Aspect-oriented programming”, in Aksit, M. and Matsuoka, S., editors, 11th European Conference Object-Oriented Programming, ol. 1241 of LNCS, pgs. 220-242. Springer Verlag.

[30] Kircher, M., Jain, P., and Corsaro, A., (2002), “XP + aop=better software”, XP ’02.

[31] Lamping, J., (1997), “The interaction of components and aspects”, in Proceedings of Workshop on Aspect-Oriented Programming (ECOOP ‘97).

[32] Lieberherr, K., Lorenz, D.H., and Wu, P., (2003), “A case for statically executable advice: Checking the law of Demeter with aspectj”, ACM 2003 (AOSD ‘03).

[33] Lieberherr, K., Orleans, D., and Ovlinger, J., (2001), “Aspect-oriented programming with adaptive methods”, *Communications of the ACM*, vol. 44, no. 10.

[34] Ludy, T., (2002), “Aspect-oriented programming can lead to faster, cheaper, easier-to-use solutions”, *Control Solutions*, vol. 11.

[35] Mahieu, T., Vanhaute, B., De Vlaminck, K., Janssens, G., and Joosen, W., (2000), “Using AOP to build complex data centric component frameworks”, in Proceedings of Workshop on Advanced Separation of Concerns (OOPSLA ‘00).

[36] Masuhara, H., Kiczales, G., and Dutchyn, C., (2002), “Compilation semantics of aspect-oriented programs”, in Proceedings of FOAL 2002: Foundations of Aspect-Oriented Languages (AOSD ‘02).

[37] Miller, S.K., (2001), “Aspect –oriented programming takes aim at software complexity”, *Technology News*, April.

[38] Murphy, G.C., Walker, R.J., and Baniassad, E.L.A., (1999), “Evaluating emerging software development technologies: Lessons learned from assessing aspect-oriented programming”, *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pgs. 438-455.

[39] Murphy, G.C., Walker, R.J., Banissad, E.L.A., Robillard, M.P., Lai, A., and Kersten, M.A., (2002), “Does aspect-oriented programming work?”, *Communications of the ACM*, vol. 44, no.10.

[40] Navasa, A., Perez, M.A., Murillo, J.M., and Hernandez, J., (2002), “Aspect-oriented software architecture: A structural perspective”, in Proceedings of Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architectural Design (AOSD ‘02).

[41] Nebbe, R.D., (1998), “Coordination and composition: The two paradigms underlying AOP?”, in Proceedings of Workshop on Aspect Oriented Programming (ECOOP ‘98).

[42] Netinant, P., Constantinides, C.A., Elrad, T., and Fayad, M.E., (2000), “Aspect-oriented frameworks: The design of adaptable operating systems”, ACM ‘00 (OOPSLA ‘00).

[43] Nordberg, M.E.III., (2001), “Aspect-oriented dependency inversion”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[44] Noro, M. and Kumazaki, A., (2002), “On aspect-oriented software architecture: It implies a process as well as a product”, IEEE ‘02.

[45] Ovlinger, J., (2003), “From aspect-oriented model to implementation”, AOSD ’03.

[46] Popovici, A., Alonso, G., Gross, T., (2001), “AOP support for mobile systems”, in Proceedings of Workshop on Advanced Separation of Concerns in Object-Oriented Systems (OOPSLA ‘01).

[47] Popovici, A., Gross, T., and Alonso, G., (2002), “Dynamic weaving for aspect-oriented programming”, ACM ‘02 (AOSD ’02).

[48] Rashid, A. and Loughram, N., (2002), “Relational database support for aspect-oriented programming”, NODE ’02.

[49] Rashid, A., Sawyer, P., Moreira, A., and Araujo, J., (2002), “Early aspects: A model for aspect-oriented requirements engineering”, IEEE ‘02.

[50] Schmidmeier, A., Hanenberg, S., and Unland, R., (2003), “Implementing known concepts in aspectJ”, in Proceedings of 3rd International German Workshop on Aspect-oriented Software Development of the SIG Object-oriented software development.

[51] Schonger, S., Pulvermuller, E., and Sarstedt, S., (2002), “Aspect-oriented programming and component weaving: Using xml representations of abstract syntax trees”, in Proceedings of Workshop on Aspect-oriented Software Development.

[52] Schult, W. and Polze, A., (2002), “Aspect-oriented programming with c# and .net”, IEEE ‘02.

[53] Shah, V. and Hill, F., (2003), “An aspect-oriented security framework”, IEEE ‘03.

[54] Shomrat, M. and Yehudai, A., (2002), “Obvious or not? Regulating architectural decisions using aspect-oriented programming”, ACM ’02, (AOSD ‘02).

[55] Spinczyk, O., Gal, A., and Schroder-Preikschat, W., (2002), “Aspect C++: An aspect-oriented extension to the C++ programming language”, in Proceedings of 40th International Conference on Technology of Object-Oriented Languages and Systems.

[56] Stein, D., Hanenberg, S., and Unland, R., (2002), “A uml-based aspect-oriented design notation for aspectj”, ACM ’02, (AOSD ’02).

[57] Storzer, M., (2003), “Analysis of aspectJ programs”, AOSD-GI ’03.

[58] Sullivan, K., Go, L., and Cai, Y., (2002), “Non-modularity in aspect-oriented languages: Integration as a crosscutting concern for aspectJ”, ACM 2002 (AOSD ‘02).

[59] Viega, J. and Voas, J., (2000), “Can aspect-oriented programming lead to more reliable software?”, IEEE ‘00.

[60] Wagelaar, D. and Bergmans, L., (2002), “Using a concept-based approach to aspect-oriented software design”, in Proceedings of Workshop on Identifying, Separating and Verifying Concerns in the Design (AOSD ‘02).

[61] Walker, R.J., Baniassad, E.L.A., and Murphy, G.C., (1999), “An initial assessment of aspect-oriented programming”, in Proceedings of 21st Int’l Conference Software Engineering (ICSE ‘99).

[62] Zakaria, A.A., Hosny, H., and Zeid, A., (2002), “A uml extension for modeling aspect-oriented systems”, in Proceedings of AOSD ‘02.

[63] Zhao, J., (2002), “Change impact analysis for aspect-oriented software evolution”, ACM ‘02.

[64] Zhao, J., (2002), “Towards a metrics suite for aspect-oriented software”, *Technical Report SE-136-25, Information Processing Society of Japan (IPSJ).*

VITA

MARIO BERNARD HANKERSON

Personal Data:	Date of Birth: October 23, 1975 Place of Birth: Oglethorpe, Georgia Marital Status: Single
Education:	Macon County High School, Montezuma, Georgia, 1994
	East Tennessee State University, Johnson City, Tennessee; Criminal Justice, B.S., 1998
	East Tennessee State University, Johnson City, Tennessee; Criminal Justice and Criminology, M.A., 2002
	East Tennessee State University, Johnson City, Tennessee; Computer Science, M.S., 2003
Professional Experience:	Technical Support, Sears Payment Systems, Gray, TN, 1999-2000
	Graduate Assistant, East Tennessee State University, College of Arts and Sciences, 1999-2003
	Graduate Assistant, East Tennessee State University, Ronald McNair and Cultural Affairs, 2003
Honors and Awards:	Member- ACM and the IEEE Computer Society