Towards a Taxonomy of Aspect-Oriented Programming.

Mario Bernard Hankerson
East Tennessee State University

Follow this and additional works at: http://dc.etsu.edu/etd

Recommended Citation

This Thesis - Open Access is brought to you for free and open access by Digital Commons @ East Tennessee State University. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons @ East Tennessee State University. For more information, please contact dcadmin@etsu.edu.
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.
# The AOP Categorization Model

<table>
<thead>
<tr>
<th>Category Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concerns</td>
<td>28</td>
</tr>
<tr>
<td>Adaptive</td>
<td>29</td>
</tr>
<tr>
<td>Limitations</td>
<td>29</td>
</tr>
<tr>
<td>Experiential and Case Study</td>
<td>29</td>
</tr>
<tr>
<td>Performance and Reliability</td>
<td>29</td>
</tr>
<tr>
<td>Distributed Systems</td>
<td>29</td>
</tr>
<tr>
<td>Theory</td>
<td>29</td>
</tr>
<tr>
<td>UML Modeling</td>
<td>29</td>
</tr>
<tr>
<td>Architecture</td>
<td>29</td>
</tr>
<tr>
<td>Software Engineering</td>
<td>29</td>
</tr>
<tr>
<td>Trends and Direction</td>
<td>30</td>
</tr>
<tr>
<td>Aspects</td>
<td>30</td>
</tr>
<tr>
<td>Weaving</td>
<td>30</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>30</td>
</tr>
<tr>
<td>Comparisons and Contrasts</td>
<td>30</td>
</tr>
<tr>
<td>Multidimensional View of the Taxonomy</td>
<td>34</td>
</tr>
<tr>
<td>Primary Dimension</td>
<td>35</td>
</tr>
<tr>
<td>Secondary Dimension</td>
<td>40</td>
</tr>
<tr>
<td>Tertiary Dimension</td>
<td>45</td>
</tr>
<tr>
<td>Quaternary Dimension</td>
<td>48</td>
</tr>
<tr>
<td>AOP Yearly Literature Trends</td>
<td>50</td>
</tr>
<tr>
<td>Conclusion</td>
<td>54</td>
</tr>
<tr>
<td>4. Conclusions and Future Work</td>
<td>55</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>57</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Aop Database Tables Descriptions</td>
<td>26</td>
</tr>
<tr>
<td>2. Categories Table Schema</td>
<td>27</td>
</tr>
<tr>
<td>3. Taxonomy Table Schema</td>
<td>27</td>
</tr>
<tr>
<td>4. Literature Taxonomical Breakdowns</td>
<td>31</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

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

---

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

<table>
<thead>
<tr>
<th>Categories</th>
<th>Taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>Sub Category ID</td>
</tr>
<tr>
<td>Sub Categories</td>
<td>Citation</td>
</tr>
<tr>
<td>Category Name</td>
<td>Year</td>
</tr>
<tr>
<td></td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td>Secondary</td>
</tr>
<tr>
<td></td>
<td>Tertiary</td>
</tr>
<tr>
<td></td>
<td>Quaternary</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Table Name</th>
<th>The type of data described in the table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Categories</td>
<td>General description of category information</td>
</tr>
<tr>
<td>Taxonomy</td>
<td>Bibliography data and categorical references</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column description</th>
<th>Required value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Category ID</td>
<td>The sub-categories ID, range 1a – 15g</td>
<td>Yes</td>
</tr>
<tr>
<td>Sub Categories</td>
<td>The sub-categories name</td>
<td>Yes</td>
</tr>
<tr>
<td>Category Name</td>
<td>Name of the main categories</td>
<td>No</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Column name</th>
<th>Column description</th>
<th>Required Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The article auto-number</td>
<td>Yes</td>
</tr>
<tr>
<td>Citation</td>
<td>The article cite</td>
<td>Yes</td>
</tr>
<tr>
<td>Year</td>
<td>The year article available</td>
<td>Yes</td>
</tr>
<tr>
<td>Primary</td>
<td>The one dimensional perspective, main category</td>
<td>Yes</td>
</tr>
<tr>
<td>Secondary</td>
<td>The second dimensional perspective</td>
<td>No</td>
</tr>
<tr>
<td>Tertiary</td>
<td>The third dimensional perspective</td>
<td>No</td>
</tr>
<tr>
<td>Quaternary</td>
<td>The fourth dimensional perspective</td>
<td>No</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Category</th>
<th>Scope</th>
<th>Subcategories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1.b. Modeling Concerns [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.c. Identifying Concerns [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.e. Cross-cutting Concerns [5]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.c. Tools &amp; Applications [2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.b. Theoretical [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.c. Tools [1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.c. Design Aids &amp; Analysis [9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.e. Simulation &amp; Modeling [9]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.c. Reusability [3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.c. Client-Server (Web) [4]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.e. Programming [2]</td>
</tr>
</tbody>
</table>
7. Theory [66]
   7.a. General [1]
   7.c. Applied [34]**
   7.d. Theoretical [16]

8. UML Modeling [33]
   8.c. Extensions [11]**
   8.e. Verification & Analysis [3]

9. Architecture [23]
   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.f. Design Tools & Languages [18]

11. Trends & Direction [26]
    11.c. Future Domains [9]**
    11.e. Language Constructs [1]

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

14. Miscellaneous [91]
   14.a. General [0]
   14.b. Tools [0]
   14.c. Concerns [18]
   14.e. Distributed Systems [9]
   14.g. Software Engineering [22] **
   14.h. Case Study & Experiential [5]
   14.i. Architecture [1]
   14.l. Aspects [0]
   14.m. UML [7]

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.
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).

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.
Figure 4: Primary Dimension Breakdown Continued (6A – 9C)


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.

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 represents the breakdown of the primary dimension of sub-categories 14a – 15g. The sub-categories are labeled following their respective main category heading: 14.


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**

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)


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.

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.

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.
Figure 11: Secondary Dimension Breakdown Continued (14A – 15G)

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.


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.
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.

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.
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.
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 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**

![AOP Year Breakdown Chart](image)

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.

**Figure 18: AOP Leading Category per Year**

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 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/](http://citeseer.org/) and [http://liinwww.ira.uka.de/bibliography/index.html](http://liinwww.ira.uka.de/bibliography/index.html)
REFERENCES


APPENDIX A

Literature Used in Taxonomy


89


96


# APPENDIX B

Primary Dimension Articles and Categories

<table>
<thead>
<tr>
<th>Sub-category</th>
<th>Article Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.A</td>
<td>62</td>
</tr>
<tr>
<td>1.B</td>
<td>444, 187, 76</td>
</tr>
<tr>
<td>1.C</td>
<td>315, 153, 49</td>
</tr>
<tr>
<td>1.D</td>
<td>486, 475, 392, 360, 349, 331, 309, 188, 177, 129, 115, 32, 7</td>
</tr>
<tr>
<td>1.E</td>
<td>437, 291, 265, 45, 27</td>
</tr>
<tr>
<td>2.C</td>
<td>482, 304</td>
</tr>
<tr>
<td>3.B</td>
<td>455</td>
</tr>
<tr>
<td>3.C</td>
<td>144</td>
</tr>
<tr>
<td>3.D</td>
<td>284, 214, 158, 119, 63</td>
</tr>
<tr>
<td>4.A</td>
<td>470, 469, 386, 328, 311, 261, 125, 56</td>
</tr>
<tr>
<td>4.D</td>
<td>395</td>
</tr>
<tr>
<td>5.A</td>
<td>411, 290</td>
</tr>
<tr>
<td>5.B</td>
<td>412, 233, 232, 55, 13, 12</td>
</tr>
<tr>
<td>5.C</td>
<td>356, 203, 47</td>
</tr>
<tr>
<td>5.D</td>
<td>466, 462, 415, 181, 171</td>
</tr>
<tr>
<td>6.A</td>
<td>208, 160, 157, 34</td>
</tr>
<tr>
<td>6.B</td>
<td>487, 480, 384, 375, 365, 262, 213, 179, 68, 42</td>
</tr>
<tr>
<td>6.C</td>
<td>457, 427, 385, 146</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
</tr>
<tr>
<td>6.D</td>
<td>372, 369, 173, 134, 120, 54</td>
</tr>
<tr>
<td>6.E</td>
<td>255, 180</td>
</tr>
<tr>
<td>6.F</td>
<td>251, 176</td>
</tr>
<tr>
<td>7.A</td>
<td>421</td>
</tr>
<tr>
<td>7.D</td>
<td>472, 433, 342, 334, 303, 279, 253, 223, 204, 202, 166, 127, 104, 98, 52, 28</td>
</tr>
<tr>
<td>8.A</td>
<td>243, 242, 240, 230, 10, 9</td>
</tr>
<tr>
<td>8.B</td>
<td>231, 229</td>
</tr>
<tr>
<td>8.C</td>
<td>484, 446, 428, 429, 402, 364, 302, 289, 217, 196, 192</td>
</tr>
<tr>
<td>8.D</td>
<td>431, 430, 367, 326, 256, 248, 212, 139, 78, 29, 8</td>
</tr>
<tr>
<td>8.E</td>
<td>359, 252, 37</td>
</tr>
<tr>
<td>9.A</td>
<td>416, 271, 250, 244, 161, 126</td>
</tr>
<tr>
<td>9.B</td>
<td>99</td>
</tr>
<tr>
<td>10.B</td>
<td>393, 391, 274, 194, 113, 94, 51, 16</td>
</tr>
<tr>
<td>10.C</td>
<td>491, 488, 435, 434, 394, 373, 348, 323, 286, 228</td>
</tr>
<tr>
<td>10.E</td>
<td>490, 301, 296, 215</td>
</tr>
<tr>
<td>11.A</td>
<td>320, 297, 266, 159, 152</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>11.B</td>
<td>467, 330, 329, 222, 164, 162, 151</td>
</tr>
<tr>
<td>11.C</td>
<td>408, 383, 295, 268, 221, 174, 163, 64, 53</td>
</tr>
<tr>
<td>11.E</td>
<td>201</td>
</tr>
<tr>
<td>12.A</td>
<td>321, 260, 118, 100</td>
</tr>
<tr>
<td>12.B</td>
<td>448, 165, 141, 30</td>
</tr>
<tr>
<td>12.C</td>
<td>305, 136, 70</td>
</tr>
<tr>
<td>12.D</td>
<td>283</td>
</tr>
<tr>
<td>12.E</td>
<td>424, 258, 257, 156</td>
</tr>
<tr>
<td>12.F</td>
<td>388, 23</td>
</tr>
<tr>
<td>13.A</td>
<td>377, 133, 60, 57, 3</td>
</tr>
<tr>
<td>13.B</td>
<td>471, 80</td>
</tr>
<tr>
<td>13.C</td>
<td>150, 24, 19</td>
</tr>
<tr>
<td>13.D</td>
<td>485, 399, 387, 186, 124</td>
</tr>
<tr>
<td>14.C</td>
<td>476, 454, 442, 419, 400, 389, 368, 336, 267, 246, 238, 206, 200, 190, 172, 117, 81, 71</td>
</tr>
<tr>
<td>14.D</td>
<td>451, 413, 404, 363, 355, 350, 211, 123, 72, 59</td>
</tr>
<tr>
<td>14.F</td>
<td>361</td>
</tr>
<tr>
<td>14.I</td>
<td>478</td>
</tr>
<tr>
<td>14.J</td>
<td>61</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>14.M</td>
<td>381, 380, 245, 209, 168, 147, 88</td>
</tr>
<tr>
<td>14.N</td>
<td>477, 463</td>
</tr>
<tr>
<td>15.B</td>
<td>447, 352, 319, 282, 275, 185, 167, 96, 92, 87, 73, 44, 33, 31, 6, 4</td>
</tr>
<tr>
<td>15.C</td>
<td>443, 441, 440, 337</td>
</tr>
<tr>
<td>15.D</td>
<td>418, 353, 345, 318, 270, 131, 90</td>
</tr>
<tr>
<td>15.E</td>
<td>277</td>
</tr>
<tr>
<td>15.F</td>
<td>407</td>
</tr>
<tr>
<td>15.G</td>
<td>93</td>
</tr>
</tbody>
</table>
### APPENDIX C

Secondary Dimension Articles and Categories

<table>
<thead>
<tr>
<th>Sub-category</th>
<th>Article Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.A</td>
<td>306, 276, 223, 322, 313</td>
</tr>
<tr>
<td>1.B</td>
<td>443, 440, 337</td>
</tr>
<tr>
<td>2.B</td>
<td>392, 115</td>
</tr>
<tr>
<td>2.C</td>
<td>50, 375</td>
</tr>
<tr>
<td>4.A</td>
<td>76, 100, 329</td>
</tr>
<tr>
<td>4.B</td>
<td>260, 486, 289, 310</td>
</tr>
<tr>
<td>4.C</td>
<td>421, 12, 27, 447, 269, 55, 188, 26</td>
</tr>
<tr>
<td>4.D</td>
<td>453</td>
</tr>
<tr>
<td>5.B</td>
<td>401</td>
</tr>
<tr>
<td>5.C</td>
<td>4, 132, 394, 38, 307, 179</td>
</tr>
<tr>
<td>5.D</td>
<td>208</td>
</tr>
<tr>
<td>6.B</td>
<td>66, 236</td>
</tr>
<tr>
<td>6.C</td>
<td>261, 99</td>
</tr>
<tr>
<td>6.D</td>
<td>158, 386, 212, 494, 143</td>
</tr>
<tr>
<td>6.E</td>
<td>237, 36, 425</td>
</tr>
<tr>
<td>7.A</td>
<td>444</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>7.B</td>
<td>32</td>
</tr>
<tr>
<td>7.D</td>
<td>171, 482, 45</td>
</tr>
<tr>
<td>8.A</td>
<td>403</td>
</tr>
<tr>
<td>8.C</td>
<td>250</td>
</tr>
<tr>
<td>8.D</td>
<td>323, 287, 384, 92, 87, 457, 456, 16</td>
</tr>
<tr>
<td>9.A</td>
<td>183, 122</td>
</tr>
<tr>
<td>9.B</td>
<td>113, 304</td>
</tr>
<tr>
<td>9.C</td>
<td>20, 251, 487</td>
</tr>
<tr>
<td>10.A</td>
<td>217, 9, 10, 230</td>
</tr>
<tr>
<td>10.B</td>
<td>243, 254, 176</td>
</tr>
<tr>
<td>10.C</td>
<td>460, 189</td>
</tr>
<tr>
<td>10.D</td>
<td>446, 441, 359, 351, 343, 345, 333, 315, 120, 96, 90</td>
</tr>
<tr>
<td>10.F</td>
<td>277, 175, 242, 216, 139, 196, 130, 192, 177, 17, 321, 89, 377, 402, 410, 19, 485, 484, 489, 493, 294, 492</td>
</tr>
<tr>
<td>11.B</td>
<td>170</td>
</tr>
<tr>
<td>11.C</td>
<td>428, 436</td>
</tr>
<tr>
<td>12.C</td>
<td>103, 413, 135, 253, 56</td>
</tr>
<tr>
<td>12.D</td>
<td>437, 411, 47, 327, 481, 284</td>
</tr>
<tr>
<td>12.E</td>
<td>224, 142, 220, 331, 178, 213, 49, 324</td>
</tr>
<tr>
<td>13.B</td>
<td>430, 118, 303</td>
</tr>
<tr>
<td>13.C</td>
<td>231, 229</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>13.D</td>
<td>472, 8</td>
</tr>
<tr>
<td>14.D</td>
<td>246</td>
</tr>
<tr>
<td>14.E</td>
<td>169, 363, 308</td>
</tr>
<tr>
<td>14.F</td>
<td>247, 128, 419</td>
</tr>
<tr>
<td>14.G</td>
<td>200, 389, 211, 209, 195, 451, 267</td>
</tr>
<tr>
<td>14.H</td>
<td>398, 140, 72, 404</td>
</tr>
<tr>
<td>14.I</td>
<td>473, 193</td>
</tr>
<tr>
<td>14.K</td>
<td>147</td>
</tr>
<tr>
<td>14.L</td>
<td>148, 74</td>
</tr>
<tr>
<td>14.M</td>
<td>65, 358, 234, 438</td>
</tr>
<tr>
<td>15.D</td>
<td>154, 150, 328, 124</td>
</tr>
</tbody>
</table>
### APPENDIX D

**Tertiary Dimension Articles and Categories**

<table>
<thead>
<tr>
<th>Sub-category</th>
<th>Article Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.B</td>
<td>441</td>
</tr>
<tr>
<td>1.C</td>
<td>323</td>
</tr>
<tr>
<td>1.D</td>
<td>345, 342, 189, 216, 287, 420, 310, 240, 421, 90, 113</td>
</tr>
<tr>
<td>1.E</td>
<td>243, 26, 96, 94, 87, 16</td>
</tr>
<tr>
<td>3.D</td>
<td>466</td>
</tr>
<tr>
<td>4.A</td>
<td>427</td>
</tr>
<tr>
<td>4.E</td>
<td>214, 176, 480</td>
</tr>
<tr>
<td>6.D</td>
<td>418, 482</td>
</tr>
<tr>
<td>6.E</td>
<td>13</td>
</tr>
<tr>
<td>8.D</td>
<td>486, 261, 17, 339, 340</td>
</tr>
<tr>
<td>9.A</td>
<td>179</td>
</tr>
<tr>
<td>9.C</td>
<td>129</td>
</tr>
<tr>
<td>10.B</td>
<td>324</td>
</tr>
<tr>
<td>10.D</td>
<td>384, 183, 487</td>
</tr>
<tr>
<td>10.F</td>
<td>447, 364, 36, 114, 119, 474, 33, 38, 86, 409</td>
</tr>
<tr>
<td>11.C</td>
<td>139, 78</td>
</tr>
<tr>
<td>12.A</td>
<td>223, 307</td>
</tr>
<tr>
<td>12.B</td>
<td>365</td>
</tr>
<tr>
<td>12.E</td>
<td>42, 218</td>
</tr>
<tr>
<td>12.F</td>
<td>28, 432, 66, 396, 425</td>
</tr>
<tr>
<td>13.A</td>
<td>367</td>
</tr>
<tr>
<td>Sub-category</td>
<td>Article Number</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>13.B</td>
<td>63, 47, 143</td>
</tr>
<tr>
<td>13.D</td>
<td>56</td>
</tr>
<tr>
<td>14.B</td>
<td>72, 106</td>
</tr>
<tr>
<td>14.C</td>
<td>473, 247, 234, 169</td>
</tr>
<tr>
<td>14.D</td>
<td>168, 308</td>
</tr>
<tr>
<td>15.B</td>
<td>45, 357, 76</td>
</tr>
<tr>
<td>15.D</td>
<td>177</td>
</tr>
</tbody>
</table>
APPENDIX E
Quaternary Dimension Articles and Categories

<table>
<thead>
<tr>
<th>Sub-category</th>
<th>Article Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.D</td>
<td>17</td>
</tr>
<tr>
<td>4.B</td>
<td>365</td>
</tr>
<tr>
<td>4.C</td>
<td>63</td>
</tr>
<tr>
<td>4.E</td>
<td>177</td>
</tr>
<tr>
<td>10.D</td>
<td>287</td>
</tr>
<tr>
<td>12.C</td>
<td>143</td>
</tr>
<tr>
<td>13.A</td>
<td>409</td>
</tr>
<tr>
<td>13.B</td>
<td>28</td>
</tr>
<tr>
<td>15.B</td>
<td>421, 216</td>
</tr>
</tbody>
</table>
APPENDIX F

Some Recommended AOP Literature


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


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

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