IMPROVING MODELLING FAULT TOLERANCE BASED ON ASPECT-ORIENTED DESIGN

AHMAD RAHIMI DENJKOLAEI

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Master of Software Engineering

Advanced Informatics School
Universiti Teknologi Malaysia

2013

ABSTRACT

Fault tolerance is a mechanism that is used in the design of systems with high reliability. Software fault tolerance usually is achieved through the diversity and redundancy that it adds additional complexity to the system design, and it focuses on the crosscutting concerns that will affect overall software units. Implementation of the fault tolerance techniques through the approaches such as object oriented programming reduce reusability, maintainability, and degree of the system modularity because crosscutting concerns distribute among objects and increases complexity , therefore to reduce the complexity aspect-oriented introduced. Aspect-oriented is a new thinking approach that separate crosscutting concerns from the components. Aspect oriented approach can be used in the high complex systems and implementing fault tolerance. Some works and research are performed in this filed but for fault tolerance techniques such as recovery blocks there is not any aspect-oriented model or design patterns. The main purpose of this study is modelling fault-tolerant technique based on aspect-oriented approach. Hence a highly used fault tolerance technique such as recovery blocks is selected for study and investigated to model by aspect-oriented. Therefore, crosscutting concern has been identified and is modelled aspect-oriented approach. Then a design model that is called the "aspect-oriented design model recovery blocks" is introduced in order to improve the reusability, maintainability and system modularity. The proposed model was evaluated with a case study by some metrics such as separation of concerns, level of dependability between components and size of program and their advantages and disadvantages has been described against object-oriented approach. As the result showed, the aspect-oriented model can decrease complexity by improving crosscutting concerns distributions and therefore improve system modularity that increases reusability and maintainability.

ABSTRAK

Kesalahan terkawal adalah satu mekanisme yang digunakan didalam sistem dengan keboleh percayaan tinggi. Kesalahan terkawal dalam satu perisian biasanya dicapai melalui pengembangan dan pengulangan yang menambah kompleksiti rekaan sesuatu sistem dan ia menumpukan kepada kerisauan tentang pengubahansuaian yang akan memberi kesan kepada keseluruhan unit perisian. Implementasi kesalahan terkawal melalui pendekatan seperti pengaturcaraan berasaskan objek mengurangkan keboleh gunaan, keboleh selenggaraan dan darjah modulariti sistem kerana pengubahsuaian dibahagikan antara objek dan meningkatkan kompleksiti, disebabkan itu untuk mengurangkan kompleksiti orientasi berasaskan aspek diperkenalkan. Orientasi berasakan aspek adalah satu kaedah menyelesaikan masalah pengubahsuaian dari komponen perisian. Di sebabkan orientasi berasaskan objek boleh di gunakkan dalam sistem yang kompleks dan mempunyai kesalahan terkawal. Beberapa kajian telah dijalankan dalam bidang bagaimanapun, dalam perkara seperti halangan kembali masih belum ada pendekatan orientasi berasaskan aspek. Jadi kajian ini memilih bidang halangan kembali untuk mengkaji model orientasi berasaskan objek. Disebabkan itu kerisauan tentang pengubahsuaian telah dikenalpasti. Kemudian satu rekabentuk model yand dinamakan *aspect-oriented design model recovery blocks* diperkenalkan untuk meningkatkan keboleh gunaan, keboleh selenggaraan dan modulariti sistem. Model yang dicadangkan di nilai menggunakan kajian kes dan matrik seperti kerisauan terasing. Darjah kebergantungan antara komponen dan saiz program dan kelebihan serta kekurangan berbanding pengaturcaraan berasaskan objek. Berdasarkan keputusan yang ditunjukkan, orientasi berasaskan aspek mengurangkan kadar kompleksiti dengan memperbaiki pengubahsuaian seterusnya meningkatkan modulariti sistem, serta menjadikan keboleh gunaan dan keboleh selenggaraan semakin meningkat.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	ACKNOWLEDGEMENT	**iv**
	ABSTRACT	**v**
	ABSTRAK	**vi**
	TABLE OF CONTENTS	**vii**
	TABLE OF TABLES	**x**
	TABLE OF FIGURES	**xi**
	LIST OF ABBREVIATIONS	**xiv**
	LIST OF APPENDICES	**xv**
1	**INTRODUCTION**	**1**
	1.1 Introduction	1
	1.2 Background of the Problem	2
	1.3 Problem Statement	5
	1.4 Goal	6
	1.5 Objectives	6
	1.6 Scope	6
	1.7 Deliverables	7
	1.8 Summary	8
2	**LITERATURE REVIEW**	**9**
	2.1 Introduction	9
	2.2 Fault tolerance	9
	2.2.1 Dependability and Fault Tolerance	10
	2.2.2 Relationship between Fault, Error and Failure	13

		2.2.3 Redundancy	14
		2.2.4 Diversity Design	14
		2.2.5 Fault Tolerance Scenario	15
		2.2.6 Software Fault Tolerance	18
	2.3	Selection Process	35
	2.4	Aspect Oriented Design	37
		2.4.1 Separation of Concerns	37
		2.4.2 Aspect oriented Programming	39
		2.4.3 Fault tolerance using aspect oriented	45
	2.5	Evaluation Metrics	47
		2.5.1 Separation of Concern Metrics	48
		2.5.2 Coupling Metrics	48
		2.5.3 Size Metrics	49
	2.6	The Previous Models	49
		2.6.1 Strengths and Weaknesses of the Previous Models	53
	2.7	Summary	55
3	**PROJECT METHODOLOGY**		**56**
	3.1	Introduction	56
	3.2	Tools & techniques	59
	3.3	Models	60
	3.4	Evaluate metric /model	60
	3.5	Summary	60
4	**FAULT TOLERANCE BASED ON ASPECT ORIENTED**		**61**
	4.1	Introduction	61
	4.2	Modelling Recovery Block Based on Aspect-Oriented	61
		4.2.1 Recovery Block Design based on Aspect oriented	65
	4.3	Summery	73
5	**EVALUATION OF THE PROPOSED MODEL**		**74**
	5.1	Introduction	74

5.2 Case Study 74
5.2.1 Sort program with Recovery Block by Object Oriented 75
5.2.2 Sort program with Recovery Block by Aspect Oriented 76
5.2.3 Evaluation 78
5.3 The Result 82
5.4 Summary 83

6 CONCLUSION 84
6.1 Introduction 84
6.2 Achievement of the Research Objectives 84
6.3 The characteristics of the research innovations 86
6.4 Constraints 86
6.5 Conclusions 87
6.6 Future works 88

7 REFERENCES 89

8 APPENDIX 93

TABLE OF TABLES

TABLE NO.	TITLE	PAGE
2.1	Product of Fault Tolerance General Scenario	17
2.2	Comparison Between Fault tolerance Techniques	36
3.1	Research Activity	57
5.1	Comparison Metrics in Object Oriented and Aspect Oriented	82

TABLE OF FIGURES

FIGURE NO.	TITLE	PAGE
1.1	The aspect model (Herrero, Sánchez, 2001)	3
1.2	Fault tolerance framework (Afonso and Silva, 2008)	4
1.3	Fault Management of Design Pattern Base on Aspect Oriented	5
2.1	Dependability Tree(Avizienis and Laprie, 2004)	10
2.2	Relationship between Dependability and Security.	13
2.3	Relationship between Fault, Error and Failure	13
2.4	Single-version fault tolerance techniques (Pullum 2001)	20
2.5	Multiple-version fault tolerance techniques (Pullum, 2001)	22
2.6	Recover Blocks Model (Dubrova, 2008)	23
2.7	Structure and Operation of the recovery block	24
2.8	Pseudo code related to Recovery Block	25
2.9	N-Version Programming technique	25
2.10	Structure and Operation of the N-Version Programming	26
2.11	Pseudo code related to the N-Version Programming	26
2.12	Distributed Recovery Blocks Operation	27
2.13	Pseudo code related to the Distributed Recovery Blocks	28
2.14	N-Self-Checking Programming with Acceptance Test	29
2.15	N-Self-Checking Programming with Comparison Process	29
2.16	Pseudo code related to the N-Self Checking Programming	30

2.17	Structure and Operation of N-Self Checking Programming	31
2.18	Structure and Performance of Consensus Recovery Blocks	32
2.19	Pseudo code related to the Consensus Recovery Blocks	33
2.20	Structure and Performance of the Acceptance Voting	34
2.21	Pseudo code related to the Acceptance Voting	35
2.22	Structure of aspect oriented	40
2.23	Combine the main C++ code and aspect code	44
2.24	One example in Aspect C++	45
2.25	Fault tolerance as aspect	46
2.26	Aspect code of time redundancy technique	47
2.27	Aspect-Oriented architecture model	50
2.28	Replication as aspect	51
2.29	Fault Management of Design Base on Aspect Oriented	52
2.29	Fault-tolerant framework	53
3.1	Research Framwork	58
3.2	The process of combine the main C++ code and aspect code	59
4.1	Model of Recovery Block without Aspect	62
4.2	Model of Recovery Block with Aspect	63
4.3	product fault tolerance base on aspect-oriented	64
4.4	Design Patterns of aspect-oriented	64
4.5	The model of recovery block base on aspect oriented	67
4.6	point cut code	68
4.7	Advice code	68
4.8	Sequence diagram for the first scenario	69
4.9	Sequence diagram for the second scenario	70

4.10	Sequence diagram for the third scenario	71
5.1	Class diagram with Recovery Block by Object Oriented	76
5.2	Class diagram with Recovery Block by Aspect Oriented	77
5.3	advice code in RBSort	77
5.4	Comparison of Separation of Concern Metrics (CDC)	78
5.5	Comparison of Separation of Concern Metrics (CDO)	79
5.6	The result of Comparison of Coupling Metrics (CBC)	80
5.7	The result of Comparison of Coupling Metrics (DIT)	80
5.8	The result of Comparison of Size Metrics (VS)	81
5.9	The result of Comparison of Size Metrics (NOV)	81

LIST OF ABBREVIATIONS

AO	-	Aspect Oriented
OO	-	Object Oriented
CBC	-	Coupling between Components
DIT	-	Depth of Inheritance
VS	-	Vocabulary Size
NOA	-	Number of Attributes
CDC	-	Concern Diffusion over Components
CDO	-	Concern Diffusion over Operations

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Research Plane	94

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the safety-critical systems modifications and changes are very important and are time-consuming because of the various tests and validation tasks that should be performed after each change. Fault-tolerance is one of the most known methods for designing safety-critical systems. Reality, fault tolerance is the ability of a system to continue performing its intended function despite faults. Fault tolerance is used in the designing of systems with high reliability. Fault tolerance usually is achieved by diversity and redundancy mechanisms. The fault tolerance is a non-functional requirement that usually adds high complexity in the design of safety-critical system when is implemented based on an approaches such as object oriented, because of cross-cutting concerns. Crosscutting concerns consist in software system features having the implementation spread across modules as tangled and scattered code. In many cases, these crosscutting concerns represent design model, invocations to model features. When a design model evolves, this can cause the addition or the change of scattered and tangled code, which contributes to the evolution of the crosscutting concern. A concern is scattered if it is related to multiple target elements, and tangled if both it and at least one other concern are related to the same target element. A crosscutting concern is a concern that is scattered.

1.2 Background of the Problem

Recently, the use of aspect-oriented programming in the field of fault tolerance has become one of the research topics. Fabry (1998) used aspect-oriented programming to define a "replication" aspect in order to improve the reusability and greater transparency of replication in the distributed environment. Also Szentiványi and Nadjm-Tehrani (2004) used aspect-oriented programming to improve performance and maintainability of fault-tolerant servers built with middleware support and migrate some operations of FT-CORBA middleware into application level. In this research, an existing FT-CORBA platform was used and performed some modifications was performed to support the aspect-oriented application extensions. Szentiványi and Nadjm-Tehrani's (2004) results showed that aspect-oriented programming can be used to implement non-functional requirements specially for availability and reliability. This two quality attribute can be improved by object replication mechanisms. Herrero and Sánchez (2001) presented a replication model named JReplica based on aspect-oriented programming. JReplica can separate characteristics of the replication code from functional behaviours of objects. Also, it is possible that programmers define new behaviours to determine fault tolerance requirements. The model presented by Herrero and Sánchez (2001) on the aspect-oriented architecture includes two levels (Figure 1.1):

- Functional Level: In this level Object functionality is defined and two new entities (in, out) in order to communicate objects with its aspects attached to each object.
- Aspect Level: In this level, aspects are defined. Each object can be associated with one or more aspects.

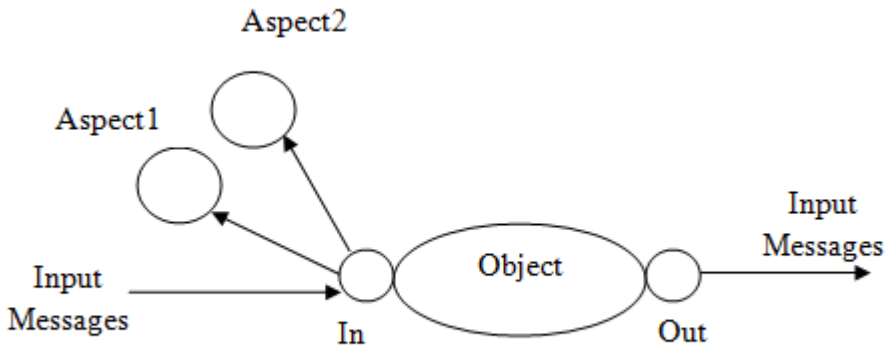

Figure 1.1 The aspect model (Herrero, Sánchez, 2001)

Alexandersson and Öhman (2010) provided one of the best works about the use of aspect-oriented programming to implement fault-tolerant tactics. Alexandersson and Öhman (2010) have been defined a set of fault-tolerant mechanisms which include: recovery cash, time redundant, recovery blocks, runtime checks and control flow checking. Also each one of these mechanisms is surveyed and implemented by AspectC++ language and analyzes recovery blocks mechanism for a specific case study. Alexandersson and Öhman (2010) noticed that time redundant, runtime checks and control flow checking mechanisms can be implemented well in an aspect-oriented programming language.

Afonso and Silva (2008) provided a fault tolerance approach for application programmers of real-time embedded systems in the operating system core by aspect-oriented. Afonso and Silva (2008) introduced a fault tolerance framework and then used it as aspect and implemented using aspect-oriented programming. Figure 1.2 shows the framework.

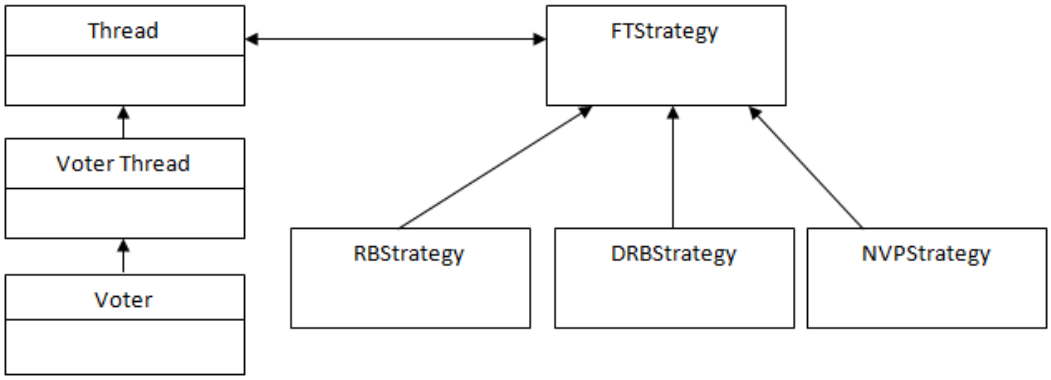

Figure 1.2 Fault tolerance framework (Afonso and Silva, 2008)

A few researches have focused on introducing and implemented aspect-oriented programming patterns. Hameed and Williams (2010) provided a design pattern on the base of aspect-oriented for error detection. Also Chavez (2004) and, Castor Filho and Garcia (2007) introduced an Error handling pattern to manage exceptions (Figure 1.3).

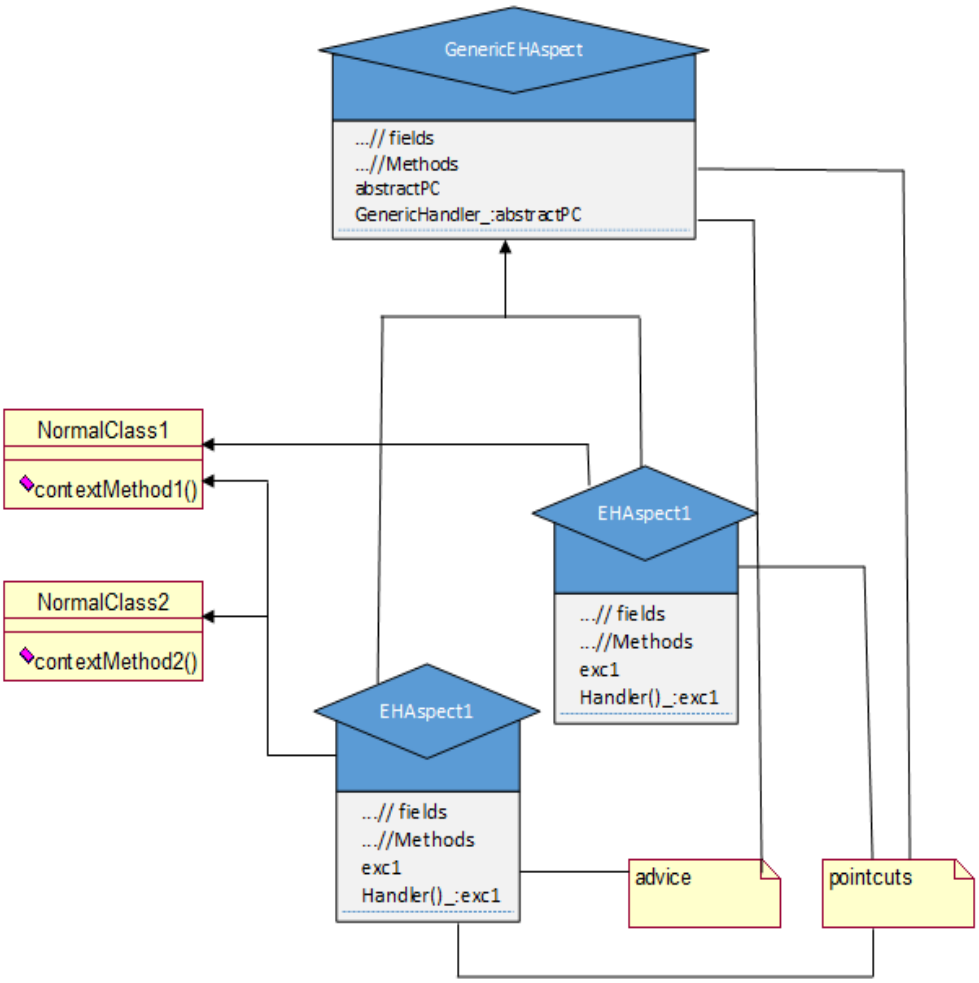

Figure 1.3 Fault tolerance of Design Pattern Base on Aspect Oriented (Hameed and Williams, 2010)

1.3 Problem Statement

Fault tolerance is used in the designing of systems with high reliability. Fault tolerance usually is achieved by diversity and redundancy mechanisms. The fault tolerance is a non-functional requirement that usually adds high complexity in the design of safety-critical system when implemented based on an approach such as object oriented, because of cross-cutting concerns. Cross-cutting usually includes two problems: tangling and scattering. Tangling is a component implementation with more than one requirement and scattering is the one with one requirement.

In this research we believe that aspect-oriented design reduces the complexity and also improves the performance of safety-critical systems because Object oriented approach focuses on the improvement of the code and programming (implementation) while aspect oriented one focuses on the concerns. In this research, the accuracy of the solution and response to improve fault tolerance are evaluated and also it is focused on the reduction of the complexity and improvement of the performance of fault tolerant design by aspect-oriented approach.

1.4 Goal

The main goal of the research is to improve the fault tolerant techniques by using aspect-oriented design in order to reduce the complexity and improving the performance of safety-critical systems.

1.5 Objectives

- To investigate and select techniques of the fault tolerance design.
- To propose fault tolerance model using aspect-oriented concept.
- To evaluate performance and complexity of fault tolerant design by the proposed model.

1.6 Scope

In addition to fault tolerance, there are other method to achieve the high dependability and high reliability that are beyond the scope of this study. Also, fault tolerant design can be implemented in hardware or software and/or different levels of software such as the operating system, middleware and application level. In this study, the scope is application level of fault tolerance will be used and using AspectC++.

In this section, the scope of the research is given based on each objective as below:

To investigate and select techniques of the fault tolerance design: In the first objective, it is crucial to have a solid understanding on the concepts of fault tolerance techniques such as Recover blocks, N-Version programming, Distributed recovery blocks, N-self checking programming, Consensus recovery blocks and Acceptance voting, and then select better technique. According to some concepts in Literature Review is selected recovery block because of this technique is the main technique.

To propose fault tolerance model using aspect-oriented concept: Based on the investigation to achieve the first objective, the model based on aspect oriented design is proposed in order to, reduce complexity and improve performance.

To evaluate performance and complexity of fault tolerant design by the proposed model: Based on the investigation to achieve the second objective, it is crucial to have a solid understanding on the some metrics such as, separation of concern metric, coupling metrics and size programme metrics. And use these metrics for compare between object oriented model and aspect oriented model.

1.7 Deliverables

Each phase in the research has contributed to the deliverable documents as below where these reports are provided in the future chapters in this thesis:

- Concepts and techniques report (Chapter 2)
- Proposed models report (Chapter 4)
- Proposed models evaluation report (Chapter5)
- Conclusion (Chapter6)

1.8 Summary

In this chapter the problem was stated and, solutions and objectives were proposed. In this research it is believed that aspect-oriented design reduces the complexity and also improves the performance of safety-critical systems because Object oriented approach focuses on the improvement of the code and programming (implementation) while aspect oriented approach focuses on the concerns. This research, evaluates the accuracy of the solution and response to the improvement of fault tolerance also focuses on the reduction the complexity and improving the performance of fault tolerant design by aspect-oriented approach. Crosscutting concerns consist in software system features having the implementation spread across modules as tangled and scattered code. In many cases, these crosscutting concerns represent design model, invocations to model features. When a design model evolves, this can cause the addition or the change of scattered and tangled code, which contributes to the evolution of the crosscutting concern.

REFERENCES

Laprie, A., and Randell, J.C., and Landwehr Avizienis, B. (2004, March). "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11-33.

Harper, J.H., and Lala, R.E.(1994). "Architectural principles for safety-critical real-time," in *Proceedings of the IEEE*, 82 (1), pp. 25-40.

Storey, N. (1996). "Safety Critical Computer Systems: Addison-Wesley Longman".

Sanchez, J.L., and Toro, F., and Herrero, M. (2001)."Fault tolerance as an aspect using JReplica," in *Proceedings of the Eighth IEEE Workshop on Future trends of Distributed Computing Systems*, Los Alamitos, pp. 201–207.

Lohmann, O., and Spinczyk, D. (2007). "The design and implementation of AspectC++," in *Knowledge Based Systems*, pp. 636-651.

Fabry, J., (1998). "A Framework for Replication of Objects using Aspect-Oriented Programming, " Phd Thesis, University of Brussel.

Nadjm-Tehrani, D., and Szentivanyi, S. (2004). "Aspects for improvement of performance in faulttolerant software," in Proceedings of the 10th IEEE Pacific Rim International Symposiumon Dependable Computing, pp. 283–291.

Alexandersson, R. (2007). "Implementing Fault Tolerance Using Aspect Oriented Programming," *Springer*, pp. LNCS 4746, pp. 57–74.

Alexandersson, R. (2010). "Aspect-Oriented Implementation of Fault Tolerance: An Assessment Assessment of Overhead," *Springer*, SAFECOMP, no. LNCS 6351, pp. 466–479.

Alexandersson, R. (2010)."On Hardware Resource Consumption for Aspect-Oriented Implementation of Fault Tolerance," in *European Dependable Computing Conference.*

Afonso, F. and Silva. C. (2008). “Aspect-oriented fault tolerance for real-time embedded systems,” in *Proceedings of the 2008 AOSD workshop on Aspects, components, and patterns for infrastructure software, ACM*.

Avizienis, A. (1995). "The methodology of n-version programming," *Software fault tolerance* 3: 23-46.

Avizienis, A. and Laprie, J.-C. (2004). "Basic concepts and taxonomy of dependable and secure computing." in *Dependable and Secure Computing, IEEE Transactions* on 1(1): 11-33.

Bass, L. and Clements, P. (2003). “Software architecture in practice,” in *Addison-Wesley Professional*.

Briere, D. and Traverse, P. (1993). “AIRBUS A320/A330/A340 electrical flight controls-a family of fault-tolerant systems,” in *Fault-Tolerant Computing, 1993. FTCS-23. Digest of Papers., The Twenty-Third International Symposium on, IEEE*.

Castor Filho, F. and Garcia, A. (2007). “Error handling as an aspect,” *Proceedings of the 2nd workshop on Best practices in applying aspect-oriented software development*.

Chavez, C. (2004). "A Model-Driven Approach for Aspect-Oriented Design."

Chen, L. and Avizienis, A. (1978). “N-version programming: A fault-tolerance approach to reliability of software operation,” in *Proc. 8th IEEE Int. Symp. on Fault-Tolerant Computing* (FTCS-8).

Dijkstra, E. (1982). “A Personal Perspective. On the role of scientific thought. Selected Writings on Computing,” *Springer-Verlag*.

Dubrova, E. (2008). "Fault tolerant design: An introduction," Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden.

Florio, D. (2009). "Application-layer fault-tolerance protocols,"

Florio, V. D. and Blondia, C. (2008). "A survey of linguistic structures for application-level fault tolerance," *ACM Computing Surveys (CSUR)* 40(2): 6.

Hameed, K. and Williams, R. (2010). "Software Fault Tolerance: An Aspect Oriented Approach." *Electronic Engineering and Computing Technology*: 153-164.

Herrero, J. L. and Sánchez, F. (2001). “Fault tolerance as an aspect using JReplica,” *Distributed Computing Systems, 2001. FTDCS 2001. Proceedings. The Eighth IEEE Workshop on Future Trends of, IEEE*.

Horning, J., Lauer, H. (1974). "A program structure for error detection and recovery," *Operating Systems*: 171-187.

Kim, K. (1995). "The distributed recovery block scheme," *Software fault tolerance* 3: 189-210.

Kim, K. and Welch, H. O. (1989). "Distributed execution of recovery blocks: An approach for uniform treatment of hardware and software faults in real-time applications," *Computers, IEEE Transactions* on 38(5): 626-636.

Kulkarni, S. and Arora, A. (2000). ”Automating the addition of fault-tolerance. Formal Techniques in Real-Time and Fault-Tolerant Systems,” *Springer*.

Lala, J. H. and Harper, R. E. (1994). "Architectural principles for safety-critical real-time applications," *Proceedings of the IEEE* 82(1): 25-40.

Pawlak, R., Seinturier, L. (2005). “Foundations of AOP for J2EE Development,” Apress.

Pullum, L. L. (2001). “Software fault tolerance techniques and implementation,” Artech House Publishers.

Siemwiorek, D. (1991). "Architecture of fault-tolerant computers: An historical perspective," *Proceedings of the IEEE* 79(12): 1710-1734.

Spinczyk, O. and Lohmann, D. (2005). "Advances in AOP with AspectC++," *New Trends in Software Methodologies, Tools and Techniques* (SoMeT’05)(129): 33-53.

Tarr, P., Ossher, H. (1999). “N degrees of separation: multi-dimensional separation of concerns,” in *Proceedings of the 21st international conference on Software engineering*, ACM.

Torres-Pomales, W. (2000). "Software fault tolerance: A tutorial," NASA Technical Report, NASA-2000-tm210616.

Traverse, P. (1988). "AIRBUS and ATR system architecture and specification," Software diversity in computerized control systems(A 88-45951 19-61). Vienna and New York, *Springer-Verlag*, 1988: 95-104.

Wingate, G. A. and Preece, C. (1993). “A software-implemented fault-tolerant technique for microprocessor controllers,” Reliability and Maintainability Symposium, 1993. Proceedings., Annual, *IEEE*.

Xu, J. and Randell, B. (2002). “A generic approach to structuring and implementing complex fault-tolerant software,” Object-Oriented Real-Time Distributed Computing, 2002.(ISORC 2002). Proceedings. Fifth IEEE International Symposium on, *IEEE*.

Xu, J. and Randell, B. (1994). “Toward an object-oriented approach to software fault tolerance,” Fault-Tolerant Parallel and Distributed Systems, 1994., Proceedings of IEEE Workshop on, IEEE.