INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

NOTE TO USERS

Page(s) not included in the original manuscript and are unavailable from the author or university. The manuscript was microfilmed as received.

17

This reproduction is the best copy available.

UMI[®]

DECIDING ON A PATTERN:

**TOWARDS A DISTRIBUTED DECISION AID THAT SUPPORTS THE
SELECTION OF AN OBJECT-ORIENTED SOFTWARE PATTERN**

by

Jonathan C. McPhail, B.Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment
of the requirements for the degree of

Master of Science
Information and System Science

School of Computer Science
Carleton University
1125 Colonel By Drive, Ottawa, Ontario

December 2000

© Copyright 2000, Jonathan C. McPhail

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-57788-0

Canada

The undersigned recommend to the Faculty of Graduate Studies and Research
acceptance of the thesis,

**Deciding On A Pattern: Towards a distributed decision aid that supports the
selection of an Object-Oriented Software Pattern**

submitted by

Jonathan C. McPhail, B.Eng.

in partial fulfillment of the requirements for the degree of

Master of Science
Information and System Science

Director, School of Computer Science

Thesis Supervisor

ABSTRACT

Object-oriented software patterns account for knowledge regarding a solution to a programming problem in a context. These patterns are increasingly popular and consequently their numbers are growing. Under these circumstances, the pattern user is challenged to decide on which pattern to incorporate into their design. Articulating a pattern decision analysis approach herein focuses on providing pragmatic support to making this design decision. The approach is applied to three independent tests, each representing a type of pattern decision problem. Reasonable results will confirm the suitability of the proposed approach and based on this confirmation, a design for a distributed pattern decision aid is proposed.

ACKNOWLEDGEMENT

A number of ideas are offered in this thesis applicable to making decisions about object-oriented software patterns. Dr. Dwight Deugo, thesis supervisor, was instrumental to ensuring that these ideas remain relevant to the pattern community of interest. As challenges were encountered, his continued encouragement fostered their resolution.

Appreciation is expressed to Greg Jenson, P.Eng. for his role in reviewing this thesis during final preparations. His comments have assisted in enhancing the readability of the thesis and for providing an objective perspective concerning the content.

It is with utmost gratitude that this thesis is dedicated to Nadine and Camillia. Without their constant patience and support, this journey would not have been feasible.

TABLE OF CONTENTS

Abstract	iii
Acknowledgement.....	iv
Table of Contents.....	v
List of Tables.....	vii
Table of Figures	viii
List of Equations.....	ix
List of Appendices	x
PART 1 THESIS INTRODUCTION	1
CHAPTER 1 Introduction.....	1
1.1 Motivation	1
1.2 Thesis Goal	3
1.3 Objectives	3
1.4 Scope	4
1.5 Contribution.....	4
1.6 Thesis Structure	6
1.7 Notation	7
PART 2 BACKGROUND	8
CHAPTER 2 Making a Decision.....	8
2.1 The Decision in General	8
2.2 Decision Analysis	10
2.3 A Closer Look At An Outranking Method.....	14
2.4 Dealing with Many Criteria.....	24
CHAPTER 3 Patterns	34
3.1 Defining Patterns	34
3.2 Pattern Structure	36
3.3 Forces	38
CHAPTER 4 Pattern Forces as Decision Criteria.....	42
4.1 Influences Among Forces.....	42
4.2 Tangible and Intangible Forces	44
4.3 A Hierarchy of Forces Dominated By Software Quality	45
4.4 Summary of Pattern Decision Analysis Force Hierarchy.....	57
CHAPTER 5 Metrics	59
5.1 The Current State of Object-Oriented Design Metrics.....	59
5.2 Metrics and their Influence on Forces	61
5.3 Measuring Size	62
5.4 Measuring Inheritance	64
5.5 Measuring Coupling	66
5.6 Measuring Cohesion.....	70
5.7 Updated Pattern Decision Analysis Force Hierarchy	72
PART 3 APPROACH	73
CHAPTER 6 Approach to Pattern Decision Analysis	73
6.1 General Approach.....	73

6.2 Approach to Pattern Decision Analysis.....	74
6.3 The Pattern Decision Analysis Approach.....	75
CHAPTER 7 Approach To Testing Pattern Decision Analysis	89
7.1 General Approach to Testing.....	89
7.2 Approach to Measuring Candidates	91
7.3 Approach to Criteria	92
7.4 Approach to Decision Maker Preferences	92
7.5 Approach to Required Thresholds.....	95
7.6 Approach to Test Serials	96
CHAPTER 8 Approach to Pattern Decision Aid Design	97
8.1 General Approach to Analysis and Design	97
8.2 Approach to Capturing Requirements.	99
8.3 Approach to Analysis	100
8.4 Approach to Design.....	101
8.5 Approach to Development.....	102
8.6 Approach to Testing	102
8.7 Approach to Handling Feedback	102
PART 4 RESULTS	104
CHAPTER 9 Test Results	104
9.1 Test Preparation.....	104
9.2 Test 1: Comparison of State Pattern Variations	107
9.3 Test 2: Comparison of Visitor Pattern Alternatives	113
9.4 Test 3: Comparison of Proxy Pattern Alternatives.....	116
9.5 Summary of Findings	118
CHAPTER 10 Pattern Decision Aid Design	119
10.1 Pattern Decision Aid Structure.....	120
10.2 Pattern Decision Aid Domain Structure	122
10.3 Persistence Strategy For The Pattern Decision Aid	123
10.4 Pattern Decision Aid Expected Behaviour	127
10.5 Assessment of the Proposed Design.....	129
PART 5 THESIS CONCLUSION	137
CHAPTER 11 Conclusion	137
11.1 Review of Objectives and Contributions.....	137
11.2 Future Directions	141
CHAPTER 12 Bibliography	144

LIST OF TABLES

Table 1 - Numerical Example 1: Candidate Profiles	20
Table 2 - Numerical Example 1: Pairwise Concordance Indices	21
Table 3 - Numerical Example 1: Pairwise Discordance Indices.....	21
Table 4 - Numerical Example 1: Ordered Results	24
Table 5 - Numerical Example 2: Candidate Profiles	30
Table 6 - Numerical Example 2: Normalized Candidate Profiles	30
Table 7 - Numerical Example 2: Interim Pairwise Concordance Indices.....	31
Table 8 - Numerical Example 2: Interim Candidate Concordance Indices	31
Table 9 - Numerical Example 2: Interim Pairwise Discordance Indices.....	32
Table 10 - Numerical Example 2: Interim Candidate Discordance Indices	32
Table 11 - Numerical Example 2: Interim Rankings.....	32
Table 12 - Numerical Example 2: Final Candidate Concordance Index	33
Table 13 - Numerical Example 2: Final Candidate Discordance Index	33
Table 14 - Numerical Example 2: Ordered Results.....	33
Table 15 - Example 3: Candidate Profiles	85
Table 16 - Example 3: Normalized values.....	85
Table 17 - Example 3: Pairwise Concordance and Discordance Values.....	86
Table 18 - Example 3: Pattern Candidate Concordance Indices.....	87
Table 19 - Example 3: Number of Outranks.....	88
Table 20 - Example 3: Pattern Decision Analysis Output	88
Table 21 - Test Serial Preference Profiles	96
Table 22 - Guide to Assumption / Requirement Catalogue	99
Table 23 - Outline of Class Responsibility Collaboration Tables	101
Table 24 - Sample Scenario Textual Description [9].....	101
Table 25 - Test Serial Performance: Input versus Output.....	108
Table 26 - Strong and Weak Flows for Test 1.a	110
Table 27 - Visitor Pattern Tradeoffs [41]	113
Table 28 - Results of Serial 2.....	114
Table 29 - Pairwise Concordance for any two Candidates	122
Table 30 - forces.dtd Element Descriptions.....	198

TABLE OF FIGURES

Figure 1 - Introduction to Preference Relationships	9
Figure 2 - Directed Graph Representation of Numerical Example.....	22
Figure 3 - Directed Graph of Ranking of Numerical Example Candidates.....	23
Figure 4 - A Force Centric View of Relationships Between Pattern Elements.....	39
Figure 5 - Influences Among Forces	43
Figure 6 - Highest Level of Force Hierarchy.....	49
Figure 7 - Performance	51
Figure 8 - Maintainability	53
Figure 9 - Reliability.....	54
Figure 10 - Utility	56
Figure 11 - Pattern Decision Analysis Force Hierarchy	58
Figure 12 - Metric Influence on Forces	62
Figure 13 - Population Metrics	63
Figure 14 - WMC for State Pattern Classes.....	64
Figure 15 - Inheritance Metrics.....	65
Figure 16 - Illustration of Applying DIT and NOC.....	66
Figure 17 - Coupling Metrics.....	67
Figure 18 - Illustration of Applying CBO, MPC and DAC.....	68
Figure 19 - Illustration of Applying RFC	69
Figure 20 - Illustration of Applying LCOM	71
Figure 21 - Updated Pattern Force Hierarchy.....	72
Figure 22 - Bottom Up Comparison	77
Figure 23 - Determining Cost or Benefit Criteria.....	78
Figure 24 - Updating Candidate Concordance Index	81
Figure 25 - Example 3: Concordance Out-flows and In-flows.....	87
Figure 26 - Highlighting Points of View	94
Figure 27 - General Approach to Analysis and Design	97
Figure 28 - Use Case for Pattern Decision Aid.....	119
Figure 29 - Model View of Pattern Decision Aid.....	120
Figure 30 - Force Type.....	123
Figure 31 - Inter-relationship between Pattern Decision Analysis Objects.....	124
Figure 32 - Untangling the Association Web	125
Figure 33 - XML Force Shadows Applied to Pattern Repository	126
Figure 34 - General Process for Pattern Decision Analysis.....	128
Figure 35 - Illustration of pda Complexity	135
Figure 36 - Overview of UML Notation.....	147
Listing 1 - Time Complexity of Pattern Decision Aid.....	134
Listing 2 - Example of DTD Syntax	195
Listing 3 - Example of Resulting XML Syntax	196
Listing 4 - forces.dtd	197
Listing 5 - XML Sample of Common Pattern Force Shadow	200
Listing 6 - Sample of a Candidate Pattern's Force Shadow.....	201

LIST OF EQUATIONS

Equation 1 - The Pairwise Concordance Index.....	17
Equation 2 - The Pairwise Discordance Index.....	17
Equation 3 - Decision Rule: a is qualified to outrank b	18
Equation 4 - Decision Rule: a strongly outranks b	18
Equation 5 - Normalizing Benefit Criteria [34].....	26
Equation 6 - Normalizing Cost Criteria [34].....	27
Equation 7 - Candidate Concordance Index.....	27
Equation 8 - Amended Pairwise Discordance Index.....	28
Equation 9 - Candidate Discordance Index.....	28
Equation 10 - Weighted Methods Per Class.....	63
Equation 11 - Depth in Inheritance Tree.....	65
Equation 12 - Number of Children.....	65
Equation 13 - Coupling between Object Classes.....	67
Equation 14 - Response for a Class.....	69
Equation 15 - Lack of Cohesion in Methods[16].....	70
Equation 16 - Pattern Pairwise Concordance Index.....	80
Equation 17 - Pattern Pairwise Discordance Index.....	80
Equation 18 - Pattern Decision Analysis Candidate Concordance Indices.....	82
Equation 19 - Cost to Traverse the Force Hierarchy.....	131
Equation 20 - Operations to Visit a Force Node.....	131
Equation 21 - Cost to Compare a Pair at a General Force Node.....	133

LIST OF APPENDICES

APPENDIX 1	UML NOTATION	147
APPENDIX 2	STATE PATTERNS TEST	148
A2.1	State Patterns Overview.....	148
A2.2	Pattern Title: State Object.....	152
A2.3	Pattern Title: State Member (Isolated State)	153
A2.4	Pattern Title: State Member (Among Select States).....	154
A2.5	Pattern Title: State Member (Common)	155
A2.6	Pattern Title: Pure State	157
A2.7	Pattern Title: Exposed State	158
A2.8	Pattern Title: State Driven Transitions	159
A2.9	Pattern Title: Owner Driven Transitions	161
A2.10	Pattern Title: Default State	163
A2.11	Pattern Title: Strategy	165
A2.12	Testing Results: State Variations.....	166
A2.13	Testing Results: State Variations with Strategy	169
APPENDIX 3	VISITOR PATTERNS TEST	172
A3.1	Visitor Pattern Overview	172
A3.2	Pattern Title: Visitor	174
A3.3	Pattern Title: Acyclic Visitor.....	176
A3.4	Pattern Title: Default Visitor	178
A3.5	Pattern Title: Extrinsic Visitor.....	180
A3.6	Testing Results	182
APPENDIX 4	PROXY PATTERNS TEST.....	184
A4.1	Summary of Pattern Information	184
A4.2	Pattern Title: Proxy.....	186
A4.3	Pattern Title: Decorator	187
A4.4	Pattern Title: Null Object	188
A4.5	Pattern Title: Distributed Proxy.....	189
A4.6	Pattern Title: Broker	191
A4.7	Testing Results	193
APPENDIX 5	XML AND PATTERN DECISION AID PERSISTENCE	195
A5.1	Guide to Document Type Definition (dtd) Syntax	195
A5.2	Guide to XML Syntax	196
A5.3	Using XML as Part of the Pattern Decision Aid Persistence Strategy	196
APPENDIX 6	PATTERN DECISION AID DESIGN REPORT	202
A6.1	System Objective.....	202
A6.2	Report Structure.....	202
A6.3	Requirements Catalogue.....	203
A6.4	Use Case Diagram	206
A6.5	Domain Structure.....	208
A6.6	Pattern Decision Aid Behaviour	212

PART 1 THESIS INTRODUCTION

CHAPTER 1 INTRODUCTION

Based on an individual's experience in programming, an object-oriented software pattern accounts for knowledge regarding a solution to a problem in a context. With the increased popularity of developing reusable object-oriented patterns, a wealth of patterns is emerging from different origins. In some cases, there is the potential for candidate patterns to address similar design issues or problems. Under these circumstances, the designer or team of an object-oriented design is faced with a dilemma, "Which pattern is more appropriate for incorporation into the target design?" It appears that conducting comparisons of patterns to answer this question has yet to be explicitly considered. The general problem examined in this thesis is providing pragmatic support to making this design decision: the *pattern decision problem*.

1.1 Motivation

The concept of articulating patterns, as originally envisaged by Christopher Alexander, is to develop a pattern language through which, a complete structure may be constructed by applying interrelated patterns. Given a single coherent pattern language, Alexander offered a straightforward procedure for choosing relevant patterns to be compressed into the decision maker's particular design [2]. With the adaptation of the pattern to object-oriented software design and given the yearly deliberations to introduce new patterns, the number and diversity of patterns available to the designer continues to grow. Hence, it is no longer practical to scan and choose from among a list of pattern titles as originally suggested [2].

The implication is that the pattern user must scrutinize the contents of each pattern. Pattern writers tend to discuss the benefits and liabilities of applying their pattern, which

are identified as *consequences* in some pattern formats. It may be suggested that by reading this discussion, the pattern user may determine, by subjectively trading off the consequences of those patterns under consideration, to select the pattern with more perceived merit [13]. Once again, with the number of possible candidate patterns vying for the pattern user's attention, this approach can become impractical. The challenge of refining the number of candidate patterns to a practical number for the pattern user's final decision forms the prime motivation to this thesis.

Given that a pattern is to be selected, the timing of that decision further motivates this thesis. It is impractical to support more than one design, in terms of design effort. That is, it is unlikely that a design team will pursue activities on two or more competing designs, each incorporating patterns similar in features, but different in internal structure. In addition, to commit to one design that must later be redone or revisited equates to a reduction of programming productivity. Hence, the motivation herein is to make design decisions that will lead to producing quality object-oriented software and to make these decisions early enough to avoid committing programming resources to an inappropriate design.

This thesis is further motivated by the continued development of a pattern repository (<http://muffin.nexus.carleton.ca/~darrell/repo/>) within the Carleton University, School of Computer Science. Given that core functionality exists for the storage, searching and viewing of design patterns, requirements exist to further assist the user in selecting a pattern. Consequently this thesis has been motivated and influenced by the existing pattern repository with a view to refine the selection of a pattern based on the results of a keyword search.

1.2 Thesis Goal

The goal of this thesis is to develop an approach that facilitates the comparison of candidate Object-Oriented Software Patterns with the intent to establish those patterns more appropriate for use in a target design.

1.3 Objectives

The objectives of this thesis can be classified within four main threads that have been applied throughout associated work. Consequently, the statement of thesis objectives corresponds to the applicable thread. In achieving the stated thesis goal, the following objectives will be addressed:

1. Decision Analysis. Objectives that relate to comparing patterns include:
 - a. Demonstrate an understanding of comparison techniques suitable for comparing object-oriented designs. This includes identifying a decision analysis approach suitable for the pattern decision problem; and
 - b. Develop a *pattern decision analysis* approach suitable for comparing candidate object-oriented software patterns. This endeavour will entail tailoring the identified decision analysis approach to deal with pattern specific criteria and metrics that distinguish one pattern from another.
2. Patterns. Associated objectives specifically related to patterns, given that they are to be compared, include:
 - a. Demonstrate an understanding of object-oriented software patterns, in particular concerning distinguishable features that make patterns comparable; and
 - b. Identify samples of patterns that are different but support similar design requirements.

3. Testing. Design and conduct tests to confirm the suitability of the developed pattern decision analysis approach; and
4. Decision Aid Tool. Propose the design of a distributed tool that supports the confirmed pattern decision analysis approach, entitled a *pattern decision aid*.

1.4 Scope

A number of issues were identified as work proceeded in conjunction with this thesis. Hence, it is stressed that the focus is on establishing a pattern decision analysis approach. Specific challenges pertain to the selection of suitable measures or indicators applicable to the pattern decision problem. Although, a representative set of object-oriented metrics has been identified and used in this thesis, a definitive and comprehensive set of metrics has not been applied. It will be established that the current state of metrics precludes such an assertion. This considered, the design proposed for a pattern decision aid, promotes an open solution, allowing for the definition of suitable metrics to evolve over time.

1.5 Contribution

While achieving the thesis goal, the application of knowledge available in many different fields is prevalent. In particular, object-oriented software patterns, object-oriented metrics, and the operations research field of decision analysis are applicable. Applying this knowledge, the thesis makes the following contributions:

1. Pattern Decision Analysis. Contributions associated to patterns, in particular in the context of making a selection decision, include:
 - a. The articulation of *forces* within a pattern structure is intended to support a designer's selection of one particular pattern over another. However, the specification of the forces section has not been explicitly defined. In this

regard this thesis proposes a distinction between forces and highlights the influences that one force or metric may have on another force;

- b. The specification of a set of requirements that pattern writers should consider while preparing their patterns, in order to facilitate the selection of their pattern; and
- c. Exploring the comparison of design elements to determine whether one is better than another has not received extensive attention. This thesis adapts a decision analysis approach to the pattern decision problem.

2. Pattern Decision Aid. Contributions that transition the proposed pattern decision analysis approach to a distributed pattern decision aid include:

- a. Based on the static nature of pattern related information, a persistence strategy for pattern metric data will be useful to any developed decision aid. This thesis introduces an eXtensible Markup Language (XML) structure to support a distributed pattern decision aid;
- b. With a suitable decision analysis method this thesis introduces the design for a pattern decision aid. This design will be suitable for a distributed environment, and sufficiently open to accommodate different perspectives on pattern decision criteria; and
- c. Assessing the time complexity implications of the proposed decision aid's core behaviour accompanies the design.

1.6 Thesis Structure

This thesis is organized into five main parts including:

1. Thesis Introduction. This chapter serves to introduce the thesis by stating relevant overarching items of interest.
2. Background. As highlighted previously, this thesis depends on the work of other fields. Namely, the operations research subject of decision analysis, object-oriented software patterns, and object-oriented design metrics. Hence, this part reviews and discusses associated material relevant to the proposed pattern decision analysis approach. Background material is further organized into four chapters: making a decision, patterns, pattern forces as decision criteria, and metrics.
3. Approach. The associated chapters outline the manner in which the material contained in the background chapters has been applied in conjunction with this thesis. The Chapters have been organized based on the threads established in the thesis objectives, with the first two threads converging within the approach to pattern decision analysis, presented in Chapter 6.
4. Results. The discussion within this part is limited to the results of testing to confirm the proposed approach followed by a general description of the pattern decision aid. Hence, only two chapters are presented, one for testing results and one for pattern decision aid design.
5. Conclusion. Principally this part summarizes the manner in which the thesis has met the stated objectives and confirms the claimed contributions. During the conduct of this thesis many challenges were encountered, some were resolved, but

it is clear that there exists further work. The thesis finishes with a statement of suggested areas of future research and work.

1.7 Notation

Before embarking on the remainder of this thesis, this section provides a brief note concerning notation. With the exceptions of directed graphs, all models are depicted using the Unified Modeling Language (UML) version 1.3. For a synopsis of the typical notation used, refer to Appendix 1. A typical convention adopted, within the depiction of some patterns, is the greying of classes where they do not contribute to the particular configuration being measured.

PART 2 BACKGROUND

CHAPTER 2 MAKING A DECISION

2.1 The Decision in General

Before a designer can attempt to understand and apply a particular pattern to their own context, they must first decide on the appropriate pattern. Given this challenge, important ingredients to any approach are a description of the decision, the decision maker, and the decision procedure.

2.1.1 The Decision

The decision is comprised of a set of candidates. In this thesis, the candidates in question are those pattern alternatives that have been selected, by the decision maker, for further consideration. Candidates can be described by a set of properties that distinguish them from the other candidates [54]. The properties applicable to candidate patterns are examined further in Chapter 4.

2.1.2 The Decision Maker

The decision maker can in turn be described by the manner in which they will react when two candidates, a and b , are presented to them. In principle, the decision maker will exhibit [54]:

1. Preference. The decision maker prefers one at the exclusion of the other. The strength of this preference can be further qualified;
2. Indifference. The decision maker is indifferent between the two candidates; or
3. Incomparability. The decision maker is unable or refuses to compare them.

Throughout, directed graphs will be used to illustrate certain conditions pertaining to the preference relationships between two candidates. In order to illustrate the three

preference relationships and introduce the associated notation, Figure 1 depicts the typical relationships. The edges of the graph represent the preference relationship between two candidates represented as the vertices [44]. Weak versus strong preferences are typically distinguished. The manner in which this is determined will be explained in the following sections. Some particular conditions that arise are depicted in Figure 1. Typical conditions for preference include *strict preference*, where candidate a is strongly preferred to candidate d , as opposed to *large preference*, where, although candidate c is weakly preferred, the strength of the preference for candidate b dominates the relationships between the two candidates. Other conditions of interest are *indifference*, as noted by the equal preference between candidate a versus b , and *incomparability*, as noted by the absence of an established preference between candidates c , and d .

Figure 1 - Introduction to Preference Relationships

2.1.3 The Decision Procedure

Finally, the aim of any decision procedure is to place more emphasis on those candidates that will enhance the decision maker's *total satisfaction* with the recommended results [28]. This overarching objective will tie together the conflicting objectives typical of the

pattern decision problem. Achieving total satisfaction, in the context of a pattern decision, is explored further in Chapter 4, which deals with pattern forces and decision criteria. For the purposes of this thesis the operations research field of decision analysis is examined, in particular concerning two competing approaches.

2.2 Decision Analysis

The intent of decision analysis, that has been adopted here, is not to dictate the selection of a particular solution, but rather organize the candidate solutions while leaving the final selection to the decision maker [48],[54]. This thesis deals with the *comparison* of two or more candidate patterns to determine a ranking of the candidates from most to the least suitable. Although comparison is considered an important application of object-oriented software design measurement, this subject has received little attention [56].

The microcosm of designs collected by object-oriented design patterns provides a suitable proving ground for approaches to decision-making within the context of object-oriented software design. The environment of the decision being made, within this thesis, is very specific. The decision maker and the point within the software life that the decision is being made characterize this environment. In particular:

1. The decision maker is isolated to the pattern user who is likely to be a software designer, investigating a set of particular candidate patterns to be applied within the scope of a larger design.
2. The point in time a decision is expected is during initial design activities where the design begins to take form. It is possible that decisions concerning patterns may take place later in the design, development or maintenance activities, but it is presumed that they will not be as frequent as the initial design.

Considering the environment within which the pattern decision problem arises, approaches to multi-criteria decision-making (MCDM) have been examined. In principle, multi-criteria decision-making is a practical challenge facing decision makers regardless of the function area. Operations research has developed various prescriptive decision analysis methods to support MCDM. In general, two approaches were considered for this thesis:

1. Multiple-Attribute Utility Theory (MAUT); and
2. Outranking Methods.

Both approaches are introduced and assessed for their merit in handling the pattern decision problem. The selection of an approach to be applied in this thesis is based on this assessment.

2.2.1 Multiple-Attribute Utility Theory (MAUT)

The associated approach focuses on the decision maker's attempt to optimize some function aggregating the particular criteria considered. An underlying assumption made by MAUT related approaches, is that each of the candidates under consideration is comparable to the others. That is, given two candidates one will be either strictly preferred or indifference may be exhibited [59]. In order to suitably apply MAUT, knowledge is required concerning:

1. Given g_i criteria, a utility function $U = U(g_1, g_2, \dots, g_n)$, is developed that relates and transforms the criteria to the same scale, and represents properties concerning the decision maker's preferences. Typically, knowledge of ideal conditions is present to further influence comparisons [59]. This function allows the ranking of all candidate solutions from best to worst [54].

2. Substitution rates are required to facilitate the compensation of a candidate when it is losing against one criterion while gaining with respect to another [54],[48].

An aspect of MAUT that detracts from its application to the pattern decision problem is the amount of interaction with the decision maker required. Given that this thesis is partly motivated by the introduction of a distributed decision aid, a significant number of interactions would greatly impact the responsiveness of such a decision aid, given the intended purpose. A further challenge to applying MAUT to this thesis is a description of ideal conditions. Currently object-oriented design metrics continue to evolve, with gradual or limited emergence of supporting empirical evidence [12]. Consequently, the general object-oriented community is not in a position to describe the ideal conditions that a candidate pattern should aspire to attain.

2.2.2 Outranking Methods

Based on the MAUT requirement for decision maker input on the decision intricacies, other approaches were developed that do not demand the building of as rigorous a decision maker preference profile. Among these methods, a key assumption is that there may exist cases where two candidates are not comparable. The decision maker is the origin of these cases in that they may not know how to, may not want to, or cannot compare two candidates [59]. Outranking methods use limited knowledge about the decision maker's preferences and the quality of comparison results to determine that a particular solution a is at least as good as b , while there is not sufficient evidence available to dispute that conclusion [54]. In essence, these methods do require that the decision maker assign weights to particular criteria, as well as certain thresholds.

2.2.3 Selection of an Approach to Decision Analysis

In selecting which approach to apply in this thesis, three issues were considered. Of the following issues, with accompanying analysis, the first is considered the most important as it can be directly linked to the thesis motivation:

1. Distributed Pattern Decision Aid. In the context of developing a distributed pattern decision aid, the target audience of the tool could be anyone interested in using patterns, from students to designers. This implies that a preference profile would need to be constructed for each user over a distributed environment. If MAUT is applied, heavy initial or multiple interactions for user input will be required. This interaction intensity in turn implies a greater user think time for the various preferences. Alternatively, an outranking approach requires less input implying lower user think time during a session.
2. Object-Oriented Design Ideal Conditions. Furthermore, there is a challenge facing the description of what constitute ideal conditions for object-oriented design in general. An example is an ongoing discussion of the permitted depth of an inheritance hierarchy before it no longer has a positive impact on maintainability [14],[20],[35]. In order to derive a statement of ideal conditions numerous empirical studies are required which are only slowly emerging. Thus, deriving a valid utility function to apply MAUT is currently not practical. The outranking method does not require a statement of ideal conditions; rather the merits of one candidate are compared to the merits of another, but not to ideals.
3. Incomparability. MAUT insists that one candidate will be preferred to another, whereas in outranking methods the presence of incomparable candidates is

allowed. Accommodating this incomparability more closely models the pattern decision problem than MAUT.

On each of these points, the outranking methods are preferred. Coupled with the consideration that the thesis is partly motivated by the first point, it is concluded that pattern decision analysis will apply an outranking method. The following sections present a closer look at the outranking method selected.

2.3 A Closer Look At An Outranking Method

Even though the outranking methods approach has been selected, there exist many alternative methods. While reviewing the alternatives [54], the deciding point in selecting the outranking approach applies again. Based on this point, the **EL**imination **Et** **Choix** **T**raduisant la **RE**alité II (ELECTRE II) method is selected due to the limited amount of user input. Another consideration in favour of ELECTRE II is that it can be extended to account for specific decision problem properties [39]. A particular challenge facing the pattern decision problem is many criteria may be applied to compare different candidates. Given this number, there exists an extension to ELECTRE II, which further reduces the input requirement with a view to enhance the decision reliability [38]. The remainder of this section introduces the ELECTRE II method and an extension, both followed by a numerical example.

2.3.1 ELECTRE II Input

In brief the ELECTRE II method uses decision maker assigned weights to reflect their preference for each criteria. The method translates the preferences into values of *concordance*, the agreement that candidate a outranks candidate b , and *discordance*, the refusal that candidate a outranks candidate b . The results of these comparisons are tested

against related thresholds of concordance and discordance. The candidate solutions are then organized according to these results.

ELECTRE II depends on the presence of certain information to describe the decision problem with the following serving as input to the method:

1. The Candidates. The various alternatives to be evaluated during the decision procedure are described in terms of the multiple criteria relevant to the decision.
2. The Criteria. A candidate, a , is rated against a certain criterion j by a criterion value, $g_j(a)$. Given values for a particular criterion, a preference relation must exist implicitly. That is, a decision maker will prefer a higher criterion value to a candidate with a lower criterion value, or the inverse.
3. The Decision Maker's Preference. The decision maker will prefer certain criteria to others. Hence, this preference among the criteria is accounted for by a set of *preference weights*, p_j , assigned to the respective criteria.
4. Concordance Threshold. Based on the previously defined concordance, the decision maker is prompted for input concerning two thresholds:
 - a. Strong concordance threshold, $\hat{c}_1$, is indicative of a strong preference relationship between candidate a to b . The value of $\hat{c}_1$ is between 0 and 1.
 - b. Weak concordance threshold, $\hat{c}_2$, represents a weak preference relation of a to b . Similarly, the value of this threshold is between 0 and 1.
5. Discordance Threshold. This threshold, denoted $\hat{d}$, is indicative of the incomparability of a to b . Whereas concordance promotes the assertion that

candidate a outranks b , discordance is a refusal of such a conclusion. Hence, this threshold establishes a ceiling to any disagreements that are tolerated before the assertion is refused. The detailed roles of this and concordance thresholds is included in the subsequent description of the ELECTRE II method.

2.3.2 ELECTRE II Method

The explanation of the actual ELECTRE II method will be conveyed by activity stages enroute to producing an ordered list of candidates, grouped according to the analyzed comparison results. These stages include:

1. Comparing the candidates; and
2. Analyzing the comparison results.

Stage 1: Comparing the Candidates.

This stage is conducted in two steps:

1. Calculating the pairwise concordance index; and
2. Calculating the pairwise discordance index.

Step 1.1: Calculating the Pairwise Concordance index.

Given the preference weights and the representation of candidates by the values related to the multiple criteria, a *pairwise concordance index*, $c(a,b)$, is calculated by pairwise comparison of the candidates a to b . During stage 2, this index will be compared to the specified concordance thresholds. The pairwise concordance index expressed in Equation 1 is the sum of preference weights where the criteria value of candidate a , $g_j(a)$, is preferred to that of candidate b . This summation is normalized to an index between 0 and 1 by dividing by the sum of all preference weights. It must be highlighted that the

account for qualitative scales such as high, medium, low. It is assumed, for the purposes of the pattern decision problem, that these scales can be enumerated and subsequent discordance analysis conducted on the applicable values using Equation 2.

Stage 2: Analyzing the Comparison Results.

This stage is conducted in two steps:

1. Assessing comparison results according to decision rules; and
2. Ranking the candidates according to the assessment.

Equation 3 - Decision Rule: a is qualified to outrank b

$$Q(a,b) \left\{ \begin{array}{ll} c(a,b) \geq \hat{c}_2 & \text{rule 3.a} \\ \sum_{j: g_j(a) < g_j(b)} p_j < \sum_{j: g_j(a) > g_j(b)} p_j & \text{rule 3.b} \\ d(a,b) < \hat{d} & \text{rule 3.c} \end{array} \right.$$

Equation 4 - Decision Rule: a strongly outranks b

$$S(a,b) \left\{ \begin{array}{ll} c(a,b) \geq \hat{c}_1 & \text{rule 4.a} \\ \sum_{j: g_j(a) < g_j(b)} p_j < \sum_{j: g_j(a) > g_j(b)} p_j & \text{rule 4.b} \\ d(a,b) < \hat{d} & \text{rule 4.c} \end{array} \right.$$

Step 2.1: Assessing Comparison Results According to Decision Rules.

This step entails deciding the strength of the outranking relations as applied to the comparison results. The specific results from stage1 applicable include strict preference, pairwise concordance and pairwise discordance. The particular decision rules applied are included in Equations 3 and 4. It can be observed that the qualified outranking relation $Q(a,b)$ of Equation 3 is comprised of the union of the strong outranking relation $S(a,b)$ with a weak outranking relation $W(a,b)$. As the strong outranking relation is formed, the

weak outranking relation is comprised of those qualifying comparisons that are not strong.

Step 2.2 Ranking the Candidates According to the Assessment.

In order to produce the final results, two rankings are produced to come to appropriate recommendations for the decision. For the purposes of this description, the out-degree and in-degree of a candidate's node in a directed graph representation of preferences is considered [54]. Hence the two rankings are constructed as follows:

1. Ranking based on out-degree from highest to lowest. The out-degree of strong preference represents the number of candidates that the candidate in question strongly out-ranks. Consider the out-degree of weak preferences to break ties.
2. Ranking based on in-degree from lowest to highest. The in-degree of strong preferences represents the number of candidates that outrank the candidate in question. Considering the in-degree of weak preferences similarly breaks ties.

2.3.3 ELECTRE II Output

Based on these two rankings a single ordering of candidates is produced based on the relative position of a candidate within each ranking. Problem conditions can be detected based on a candidate's position in each of these rankings. For example if a candidate is last in the out-degree ranking and first in the in-degree ranking, this may indicate that the candidate outranks no other, and in turn is not outranked by any other. For the purposes of this thesis, studying and detecting problem conditions has been left as future work to this thesis.

2.3.4 Example 1

A numerical example is presented to reinforce the preceding discussion concerning the original ELECTRE II method. The example is conveyed using the same format as the original description.

ELECTRE II Input

Table 1 describes the decision problem, in accordance with the input requirements stipulated. The criteria values are arbitrary and may represent enumerated qualitative scales, or the values may be actual measured values. Note that the precision between different criteria may be mixed, but within particular criteria it is consistent. In Table 1, criterion 1, g_1 , is representative of a enumerated qualitative scale such as 1 – low, 2 – medium, 3 – high while, g_4 , is representative of a simple Boolean such as 0 – No, 1 – Yes. Furthermore the preference weights can be treated as whole numbers or as a percentage. This precision will be adjusted, as all criteria will transform into ratios during the method.

Table 1 - Numerical Example 1: Candidate Profiles

Candidate	Criteria (g_j)			
	g_1	g_2	g_3	g_4
v	2	0.74	2.00	1
w	3	0.60	3.00	1
x	2	0.45	2.00	0
y	3	0.56	4.00	1
z	1	0.80	4.00	0

Preference Weights (p_i)	5	3	3	4
$\hat{c}_1 = 0.80$				
$\hat{c}_2 = 0.60$				
$\hat{d} = 0.10$				

ELECTRE II Step 1.1

In executing the calculations as they apply to the numerical example, Table 2 represents the interim results as indicated in the upper left corner of each table with the last containing the pairwise concordance indices for this numerical example. Note that the sum of preference weights is 15, which is used in calculating these pairwise indices.

Table 2 - Numerical Example 1: Pairwise Concordance Indices

$\sum_{j: g_j(a) \geq g_j(b)} p_j$	Candidate (b)				
Candidate (a)	v	w	x	y	z
v	-	7	15	7	9
w	12	-	15	12	9
x	8	0	-	0	9
y	12	12	15	-	12
z	6	6	10	6	-

$\sum_{j: g_j(a) > g_j(b)} p_j$	Candidate (b)				
Candidate (a)	v	w	x	y	z
v	-	3	7	3	9
w	8	-	15	3	9
x	0	0	-	0	5
y	8	3	15	-	9
z	6	6	6	3	-

$c(a, b)$	Candidate (b)				
Candidate (a)	v	w	x	y	z
v	-	0.47	1.00	0.47	0.60
w	0.80	-	1.00	0.80	0.60
x	0.53	0.00	-	0.00	0.60
y	0.80	0.80	1.00	-	0.80
z	0.40	0.40	0.67	0.40	-

ELECTRE II Step 1.2

The corresponding pairwise discordance indices calculated with a value of $\delta = 2$, are contained in Table 3.

Table 3 - Numerical Example 1: Pairwise Discordance Indices

$d(a, b)$	Candidate (b)				
Candidate (a)	v	w	x	y	z
v	-	0.5	0.00	1.00	1.00
w	0.07	-	0.00	0.50	0.50
x	0.50	0.50	-	1.00	1.00
y	0.09	0.02	0.00	-	0.12
z	0.50	1.00	0.50	1.00	-

ELECTRE II Step 2.1

The first series of decision rules are applied to the values in Tables 2 and 3 to develop the relation $Q(a,b)$.

$$\begin{aligned}
 Q(a,b) &= \{(v,x),(v,z),(w,v),(w,x),(w,y),(w,z),(x,z),(y,v),(y,w), && \text{applying rule 3.a} \\
 &\quad (y,x),(y,z),(z,x)\} \\
 &= \{(v,x),(v,z),(w,v),(w,x),(w,z),(y,v),(y,x),(y,z),(z,x)\} && \text{applying rule 3.b} \\
 &= \{(v,x),(w,v),(w,x),(y,v),(y,x)\} && \text{applying rule 3.c}
 \end{aligned}$$

It should be noted that the comparisons $(v,z),(w,z),(y,z),(z,x)$ are of interest as they are refused results based on rule 3.c. The strong outranking relation can be similarly determined, but it should be noted that removing the strong outranking relation from further consideration yields an empty set of weak outranking relations.

$$\begin{aligned}
 S(a,b) &= \{(v,x),(w,v),(w,x),(y,v),(y,x)\} \\
 W(a,b) &= \{ \}
 \end{aligned}$$

It will be helpful in the following analysis step, to view these relations from a directed graph perspective as illustrated in Figure 2. For each ordered pair in the relation, a directed edge between implicated candidates is annotated with the strength of the pairwise concordance index.

Figure 2 - Directed Graph Representation of Numerical Example

ELECTRE II Step 2.2

The directed graph representations of the two rankings applied to the numerical example are illustrated in Figure 3. Each level of the graph is annotated with the respective out-degree or in-degree. Based on these two rankings the results can be ordered, forming the output.

Figure 3 - Directed Graph of Ranking of Numerical Example Candidates

ELECTRE II Output

To draw the numerical example to a conclusion Table 4 presents the final ordering of candidates grouping together those candidates for which there is no distinguishable preference. Hence, it cannot be concluded that candidate *w* is superior to candidate *y*. As well candidates that are incomparable to all others are isolated. Candidate *z* is the only completely incomparable candidate with the recommended action entailing relaxation of the concordance thresholds or accepting this incomparability and assessing candidate *z* along with candidate *w* and *y* more closely.

Table 4 - Numerical Example 1: Ordered Results

Final Order	Completely Incomparable
w	z
y	
v	
x	

2.4 Dealing with Many Criteria

During subsequent discussion of pattern forces and decision criteria, the pattern decision problem can be described by a diverse set of criteria. Depending on the description of criteria related to patterns, the number has the potential to be quite extensive. This number will likely detract from the feasibility of developing a pattern decision aid if the decision maker must assign a preference to each of the criteria. This difficulty has been examined and it has been previously postulated elsewhere that an increasing number of decision criteria impacts negatively on the reliability of an associated decision result [39]. Accompanying this assertion are options to remedy the challenge. The option applied in this thesis entails establishing a *hierarchy of criteria*, where many criteria can be categorized or generalized. The decision maker then assigns weights at the highest level in the hierarchy. This alternative was selected over conducting a multivariate or interdependence analysis between criteria to reduce the numbers. Such analysis among object-oriented design metrics is a field of study possessing its own merits.

The remainder of this section outlines an extension to conducting ELECTRE II by applying a hierarchy of criteria. It should be noted that the proposed pattern decision analysis adapts the extension offered in [39]. It is tailored to deal with the specific pattern decision problem, and address perceived shortcomings in the presented extension. It should be further noted that the original extension [38] attempted to simplify ELECTRE II. In essence, this simplification entails removing the application of the ELECTRE II

decision rules and does not make use of the concordance and discordance thresholds. Pattern decision analysis re-examines these specific issues. For lack of a specific title dealing with a hierarchy of criteria will be entitled here: ELECTRE II Extension.

2.4.1 ELECTRE II Extension Input

The input for this method is the same as the original ELECTRE II with the exceptions:

1. The Criteria. The decision criteria are organized into categories, hence forming a hierarchy of criteria.
2. The Decision Maker's Preference. The decision maker will assign preference to the highest level of criteria categories in the related hierarchy.

Similar to the original ELECTRE II explanation, a numerical example adapted to a hierarchy of criteria will follow the description.

2.4.2 ELECTRE II Extension Method

The method that extends ELECTRE II in order to deal with a hierarchy of criteria [38] diverges somewhat from the original method. Presented here is a version more consistent with the details of the original ELECTRE II, with the exception of the handling of discordance. The method will be described in stages consistent with the original ELECTRE II explanation including:

1. Comparing the candidates; and
2. Analyzing the comparison results.

It should be noted that these two stages are repeated, beginning with the lowest level of the criteria hierarchy and moving up until results have been generated by the comparison

of candidates against the super-criteria. It is at this level that the decision maker's preferences will be assigned.

Stage 1: Comparing the Candidates.

This stage is conducted in three steps:

1. Normalizing the criteria
2. Calculating a candidate concordance index; and
3. Calculating a candidate discordance index.

Step 1.1: Normalizing the criteria

Criteria values are normalized to avoid calculations with criteria representing different quantitative units or qualitative indices. An assumption regarding normalization made in the original extension [38] and applied to this thesis, is that a particular criteria value can be divided by a norm aligned to a desired criteria value. In this regard the normalized values are oriented towards the desirable criteria value [39]. In particular:

Equation 5 - Normalizing Benefit Criteria [34]

$$h_{ji} = 1 - \frac{g_j^{\max} - g_{ji}}{g_j^{\max} - g_j^{\min}}$$

1. Benefit Criteria. The desired value is the maximum value. Given the value, g_{ji} for criteria j , candidate i , the associated benefit normalization, h_{ji} , bounded by $0 \leq h_{ji} \leq 1$, is included in Equation 5; and
2. Cost Criteria. The desired value is the minimum value. The calculation for normalizing cost criteria, h_{ji} , bounded by $0 \leq h_{ji} \leq 1$, is calculated according to Equation 6.

Equation 6 - Normalizing Cost Criteria [34]

$$h_{ji} = 1 - \frac{g_{ji} - g_j^{\min}}{g_j^{\max} - g_j^{\min}}$$

Step 1.2: Calculating a candidate concordance index

Rather than calculating pairwise concordance indices based on all criteria assigned individual weights, multiple pairwise concordance indices are generated corresponding to each super-criterion. For the purposes of this explanation, these pairwise concordance indices are calculated in a similar fashion to the original ELECTRE II method. However, the preference weights will be assigned the same value. The weights intended to distinguish the decision maker's preference will only implicate the calculating of the final pairwise concordance indices.

Based on the pairwise concordance indices for each super-criteria j , a candidate concordance index, c_{ji} , is calculated for each candidate i in the set of candidates C . In essence, this index, establishes the dominance, based on pairwise concordance indices, of candidate i versus all other candidates, where:

Equation 7 - Candidate Concordance Index

$$c_{ji} = \sum_{\substack{i, b \in C \\ b \neq i}} c(i, b) - \sum_{\substack{i, b \in C \\ i \neq b}} c(b, i)$$

As noted previously, the generation of normalized criteria, pairwise concordance indices, and a candidate concordance index will be repeated until a single candidate concordance index has been calculated.

Step 1.3 Calculating a candidate discordance index.

This step entails calculating pairwise discordance indices in a manner that differs from the original ELECTRE II method. For the purposes of this explanation the amended pairwise discordance index, $d^*(a,b)$, for candidate a compared to b , is calculated according to Equation 8.

Equation 8 - Amended Pairwise Discordance Index

$$d^*(a,b) = \sum_{j \in D_{ab}} \frac{p_j(h_{jb} - h_{ja})}{n_{ab}}$$

$$\text{where : } D_{ab} = \{j \mid h_{ja} < h_{jb}\}$$

$$n_{ab} = |D_{ab}|$$

Similar to the treatment within the previous step a candidate discordance index, d_{ji} , for each candidate i in the set of candidates C is calculated based on the pairwise discordance indices for each super-criteria j as outlined in Equation 9.

Equation 9 - Candidate Discordance Index

$$d_{ji} = \sum_{\substack{i,b \in C \\ b \neq i}}^C d^*(i,b) - \sum_{\substack{i,b \in C \\ i \neq b}}^C d^*(b,i)$$

Stage 2: Analyzing the Comparison Results.

This stage is comprised of one step that is repeated as each level in the hierarchy of criteria is considered. This stage marks the completion of the method with the calculation of a single ranking. The step is entitled:

1. Rank candidates based on candidate concordance and discordance indices.

Step 2.1: Rank Candidates Based on Candidate Concordance and Discordance Indices.

The optimal candidate, on average, will have the highest candidate concordance index and the lowest candidate discordance index [38]. To this end candidates are ranked from

highest candidate concordance index to lowest, with ties broken by the candidate discordance index. At the interim level in the comparison the rank numbers generated act as a cost criteria for that level.

2.4.3 ELECTRE II Extension Output

A repeatable behaviour exists whereby stage 1 can be repeated followed by stage 2 until a single ranking has been generated.

Using this approach has the effect of reducing the information required from the decision maker in order to arrive at a decision. It further establishes that the original ELECTRE II method can be extended or manipulated [39]. However, the attractive feature of the original approach was the manner in which it allowed incomparability to exist between two candidates. The extension, as presented here, removes this sensitivity with a view to simplify the original method. Collaterally the extension insists on an ordering of candidates.

The features of this extension that will be leveraged into the proposed pattern decision analysis include:

1. Decision Maker assigns preference weights to highest level among criteria;
2. Normalizing criteria values (Equation 5, Equation 6); and
3. Computing a candidate concordance index.

However, pattern decision analysis will reintroduce the input of concordance and discordance thresholds and reapply the ELECTRE II decision rules (Equation 3, Equation 4). Chapter 6 deals specifically with the resulting approach to pattern decision analysis.

2.4.4 Example 2

The example presented here to demonstrate the ELECTRE II extension uses the same input values as in Example 1, but the results will be markedly different. This difference is based on the manner in which criteria have been categorized and the preferences assigned. Hence, the results of both examples should not be compared.

ELECTRE II Extension Input

Table 5 presents the input values to be used in this numerical example. Note, that beside the criteria label is an indication of whether it is a Benefit criteria (+) or Cost criterion (-). As presented at [38] the requirement for specifying the concordance and discordance thresholds was removed. This will be revisited in the proposed pattern decision analysis.

Table 5 - Numerical Example 2: Candidate Profiles

Candidate	Criteria (g_i)			
	g_A		g_B	
	$g_{1i}(+)$	$g_{2i}(-)$	$g_{3i}(+)$	$g_{4i}(+)$
v	2	0.74	2.00	1
w	3	0.60	3.00	1
x	2	0.45	2.00	0
y	3	0.56	4.00	1
z	1	0.80	4.00	0

Preference Weights (p_i)	5	3
------------------------------	---	---

ELECTRE II Extension Step 1.1

The input values are appropriately normalized with results listed in Table 6.

Table 6 - Numerical Example 2: Normalized Candidate Profiles

Candidate	Criteria (g_i)			
	g_A		g_B	
	$h_{1i}(+)$	$h_{2i}(-)$	$h_{3i}(+)$	$h_{4i}(+)$
v	0.50	0.29	0.00	1.00
w	1.00	0.57	0.50	1.00
x	0.50	1.00	0.00	0.00
y	1.00	0.69	1.00	1.00
z	0.00	0.00	1.00	0.00

ELECTRE II Extension Step 1.2

The step begins by calculating pairwise concordance indices for criteria g_A , $c_A(a,b)$ and g_B , $c_B(a,b)$. For the purposes of this numerical example, the preference weights for the criteria at the lowest level, p_1 , p_2 , p_3 , p_4 , will each be assigned a preference weight of 1. With this in mind the following are the pairwise concordance indices corresponding to each of the super-criteria.

Table 7 - Numerical Example 2: Interim Pairwise Concordance Indices

$c_A(a,b)$	Candidate (b)					$c_B(a,b)$	Candidate (b)				
Candidate (a)	v	w	x	y	z	Candidate (a)	v	w	x	y	z
v	-	0.00	0.50	0.00	1.00	v	-	0.50	1.00	0.50	0.50
w	1.00	-	0.50	0.50	1.00	w	1.00	-	1.00	0.50	0.50
x	1.00	0.50	-	0.50	1.00	x	0.00	0.00	-	0.00	0.00
y	1.00	1.00	0.50	-	1.00	y	1.00	1.00	1.00	-	1.00
z	0.00	0.00	0.00	0.00	-	z	0.50	0.50	1.00	0.50	-

Table 8 contains the corresponding candidate concordance indices for the super-criteria. The final candidate concordance index will be presented in the example's second stage results.

Table 8 - Numerical Example 2: Interim Candidate Concordance Indices

Candidate (i)	c_{Ai}	c_{Bi}
v	-1.50	-1.00
w	1.50	1.00
x	1.50	-4.00
y	2.50	2.50
z	-4.00	0.50

ELECTRE II Extension Step 1.3

Table 9 contains the calculated pairwise discordance indices calculated within each of the super-criteria A and B given preference weights are all 1. In turn Table 10 contains the corresponding candidate discordance indices for the super-criteria.

Table 9 - Numerical Example 2: Interim Pairwise Discordance Indices

$d^*_A(a,b)$	Candidate (b)					$d^*_B(a,b)$	Candidate (b)				
Candidate (a)	v	w	x	y	z	Candidate (a)	v	w	x	y	z
v	-	0.50	0.71	0.50	0.00	v	-	0.50	0.00	1.00	1.00
w	0.00	-	0.43	0.12	0.00	w	0.00	-	0.00	0.50	0.50
x	0.00	0.50	-	0.50	0.00	x	1.00	0.50	-	1.00	1.00
y	0.00	0.00	0.31	-	0.00	y	0.00	0.00	0.00	-	0.00
z	0.50	0.00	0.00	0.00	-	z	1.00	1.00	1.00	1.00	-

Table 10 - Numerical Example 2: Interim Candidate Discordance Indices

Candidate (i)	d_{Ai}	d_{Bi}
v	1.21	-0.50
w	0.00	-1.00
x	-0.45	-4.00
y	-0.36	2.50
z	-0.50	0.50

ELECTRE II Extension Step 2.1

The second numerical example is drawn to its conclusion, first by considering the ranks of the candidates, and with the generation of the final candidate indices and rankings. Table 11 contains the relative rankings among candidates based on the interim candidate concordance index and the corresponding discordance index.

Table 11 - Numerical Example 2: Interim Rankings

Candidate (i)	Rank _A	Rank _B
v	4	3
w	3	4
x	2	5
y	1	1
z	5	2

Based on these rankings a single candidate concordance and discordance index is calculated in accordance with the ELECTRE II Extension steps 1.1, and 1.2 respectively. With the preferred rank being 1, the ranks in Table 11 are normalized as a cost criterion followed by calculating pairwise concordance and discordance indices. Subsequently Table 12 and Table 13 are generated, with each table containing the candidate concordance and discordance indices respectively.

Table 12 - Numerical Example 2: Final Candidate Concordance Index

Candidate (i)	c_i
v	-1.25
w	-0.75
x	-0.25
y	4.00
z	-1.75

Table 13 - Numerical Example 2: Final Candidate Discordance Index

Candidate (i)	c_i
v	3.62
w	1.37
x	-0.88
y	-10.00
z	5.88

ELECTRE II Extension Output

Considering a descending ordering of the final candidate concordance index and an ascending ordering of the final candidate discordance index generates the final results. The resulting ordering of candidates that forms the output of this numerical example is presented in Table 14.

Table 14 - Numerical Example 2: Ordered Results

Candidate
y
x
w
v
z

The recommended action, based on this output, entails the decision maker exclusively considering candidate y before considering any other candidate.

With the completion of the second numerical example sufficient information has been presented, for this thesis, concerning decision analysis. Based on this background, the object-oriented software pattern must be examined for specific properties applicable to the respective decision problem. The remainder of the background material will explore the pattern, its properties, and how these properties are applicable to decision analysis.

CHAPTER 3 PATTERNS

3.1 Defining Patterns

Pattern decision analysis tailors the ELECTRE II method and extension to suit specific pattern properties. To situate the explanation of the proposed approach, the pattern decision problem is considered. Accompanying this discussion, pattern terminology applicable to this thesis is introduced.

In essence a pattern, regardless of setting, is a mechanism for expressing knowledge or experience regarding the resolution of a problem to an interested audience. Conceptually, originating from architecture in the late 1970's, patterns have been adapted by object-oriented practitioners as a means of conveying solutions to problems experienced in implemented object-oriented programs. The continued popularity, since the mid-nineties, highlights the inherent effectiveness of patterns to present reusable structures and behaviour regardless of context [45]. To this point the emphasis among pattern advocates has been on writing patterns so that a reader can understand the presented solution and subsequently implement a consistent interpretation in the context of the reader's own design.

Although the pattern user is the focus, a fundamental challenge is not explicitly considered in typically documented patterns. In essence, a pattern user must first decide on an appropriate solution before they must understand the details of that solution. If the number of possible solutions to a particular problem were limited, the user could be expected to review the candidate patterns, judge the trade-offs and make a selection in a timely fashion. Unfortunately the number of patterns emerging and in particular the number of different solutions to similar problems challenges the simplicity of such a decision. Principally there exist three types of pattern related decision problems:

1. Variations. There are certain patterns that resolve a fundamental problem, but are often limited in utility. A common practice among pattern writers is to devise patterns such that new patterns extend or enhance an original pattern with a view to increasing the utility. Collectively the associated family of patterns forms a *pattern language*. These enhanced patterns form *variations* to the original pattern. An example is the State pattern [26] with originally six variations introduced to deal with basic capabilities [22] and further enhanced with nine variations to specifically deal with finite state machine issues [58].
2. Alternatives. Given that one pattern writer's experience and context may differ to that of another's, alternatives to a particular pattern may emerge. This is certainly evidenced by the Visitor pattern [26] where two sets of alternative solutions were proposed by different authors [36], [41]. Likewise it is possible that the original problem is solved differently, as in the case of the Model-View-Controller and Presentation-Abstraction-Controller [13].
3. Mixed. Supporting information accompanying patterns typically alludes to, or comments on related patterns. In some cases this discussion highlights other patterns, similar in structure but different in context. It is possible that the pattern user is posed with a pattern decision problem that mixes related patterns, variations or alternatives. The Proxy pattern [26] is an example of this issue where it refers to the Decorator pattern [13] and in turn the Null Object [57] refers to the Proxy. Another, more concrete mixed decision problem is in pattern languages where the language does not stem from a single pattern.

Basic Relationship Patterns [40] exemplifies this particular problem, where the language originates from three unrelated patterns.

The patterns used in tests of the submitted pattern decision analysis are representative of these types of pattern decision problem. These patterns are described in the specified structure in Appendices 2, 3, and 4 respectively. The base patterns that provide the testing candidates include:

1. The State Pattern. This pattern handles an object that behaves differently according to the state that it is in [22]. The associated comparison is representative of the pattern decision problem involving pattern variations;
2. The Visitor Pattern. This pattern considers the manner in which different operations are applied to the elements of a collection of objects [26]. This pattern and related patterns implicated in the comparison represent the pattern alternatives decision problem; and
3. The Proxy Pattern. This pattern is used as the focal point of a mixed decision problem. This pattern deals with a client object's interaction with a server object where direct access is not suitable [13].

To further situate the pattern decision problem, focus is drawn to the internal structure of patterns and those elements that are applicable to comparing patterns.

3.2 Pattern Structure

There exist many different perspectives concerning the structure of a pattern emerging from Alexander [1],[2] with various object-oriented specializations promoted by the Gang of Four (GOF) [26], Pattern – Oriented Software Architecture (POSA) [13], and

Meszaros with Doble who expressed a pattern for writing patterns [37]. At its core, a pattern consists of a statement of a problem with an explanation of the solution in a context. Accompanying this foundation are elements intended to provide sufficient supporting information that a reader will understand and implement the pattern consistent with the writer's original intent. It is here that different pattern structures diverge. Although those necessary pattern elements that contribute to a comparison are identified, it is not intended to suggest a particular format. It is expected that the required elements can be explicitly included in the pattern document or alternatively left to shadow the documented pattern as supporting information. In general, the explanation of patterns contained within this thesis will be organized according to the following elements of interest, prefixed by a pattern title and including a list of authors:

1. Problem. The statement of a specific design problem. The problem statement is typically placed within a context where it originally occurred and may further list constraints on the application of the proposed solution [26]. Sections typically related to the statement of the problem are entitled: *intent*, *context*;
2. Forces in Problem. The various forces at play in the problem. In pattern structures, the statement of forces in the problem is either implicitly implied within the context of the pattern, or explicitly announced in their own section.
3. Solution. An explanation of the solution in terms of the participating classes, *participants*, and a description of the relationships or *collaborations* with other classes [26]. For the purposes of pattern comparison particular emphasis will be placed on the structure and behaviour of the presented solution; and

4. Forces in Solution. Often entitled *rationale* [37] or *force resolution* [51], this discussion entails explaining the manner in which the solution resolves the forces at play. The treatment of forces in the solution tends to be through a description of the benefits and liabilities associated with applying the particular pattern. Although, these consequences will undoubtedly shape the final choice of a pattern, the associated discussion is considered supporting information. It is accounting the presence of forces in the solution that plays a role in pattern comparison.

Of these main elements the articulation of forces in the solution will prove pivotal to making design decisions regarding pattern selection. Recall, that by applying decision analysis the intent is to organize the candidate patterns into categories of most applicable to least, given the pattern user's preferences. The results will present an ordered reading list upon which the pattern user should base their final decision on the trade-offs among the associated descriptions of the pattern consequences. What will be required to support this pattern decision analysis is an indication of the various forces present in the solution. The accounting of forces is dealt with specifically in Chapters 4 and 5. Subsequent discussion in this chapter concentrates on the relationship between forces and the other elements within a pattern structure.

3.3 Forces

"A man is alive when he is wholehearted, true to himself, true to his own inner forces, and able to act freely according to the nature of the situations he is in.

...To be alive, in this sense, is not a matter of suppressing some forces or tendencies, at the expense of others; it is a state of being in which all forces which arise in a man can find expression; he lives in balance among the forces which arise in him..." C. Alexander[1]

The above quotation introduces the notion that there exist forces that are universal and exist to varying degrees in each entity as originally envisaged by C. Alexander. It is by focusing on these forces in the pattern that comparison of two patterns is facilitated. This section introduces forces, their role among pattern elements, and their potential application in forming a pattern decision. Based on this discussion Chapter 4 is dedicated to casting forces as decision criteria.

There exists a consensus that *forces* are considerations to be taken into account when choosing a solution to a problem [37], [53]. By refining this definition, forces can be leveraged as criterion into decision analysis methods. Hence, the following is the refined definition of a force to be used throughout this thesis:

- Force. A force is a point of view upon which the suitability of a solution to a problem can be weighed. The force has the potential to be used as *criterion* in a decision-making problem when it is used to represent the decision maker's preferences in a given context.[54]

Figure 4 - A Force Centric View of Relationships Between Pattern Elements

Departing from a traditional view of relationships among pattern elements [37], Figure 4 reinforces the defined role that the force plays in the pattern decision problem. Based on this force centric view, a candidate pattern is the pairing of the problem with a solution supported by information related to the specification and resolution of forces. Comparing how each solution resolves forces is instrumental in forming a decision concerning candidate patterns. Although, the pattern writer operated in a context that prejudiced the originally documented pattern, a pattern user's preference for forces is influenced by the context that they themselves are operating within.

With the force definition already refined, the remaining elements of the pattern structure are reconsidered in the following definitions:

1. Context. The context outlines the problem being solved further restricting basic forces directly at play. However, constraining a particular force may in turn raise the profile of another force not directly involved.
2. Problem. It is considered that a problem has innate forces at play upon which the compliance of a candidate solution must be specified. Specifying compliance acts to filter out non-solutions and further serves to formally define the problem.
3. Solution. For a potential solution to be considered a candidate it must meet a specification or *constraint* [42] [21] of forces relevant to the problem. A solution is left to resolve those forces introduced and specified by the problem and further constrained by the context.

By considering the pattern in this regard, an abridged structure for pattern descriptions will be used throughout this thesis. The descriptions provided in Appendices 2, 3, and 4

only provide a minimum amount of information to grasp the basic premise of the pattern. An emphasis will be placed on describing those elements that will eventually impact the comparison of associated variations and alternatives in the testing of pattern decision analysis. The specific structure used includes sections for: Pattern title; authors; problem; forces in problem; solution with structure and behaviour; and finally forces in solution.

Although, the context section may facilitate the reader's understanding of those circumstances facing the pattern writer in deriving their pattern, it is not directly implicated in the pattern decision problem. Consequently this section is omitted in the abridged structure. The original context is represented in the decision problem by a description of the forces in the problem. Recall that in the pattern decision problem it is the pattern user's context that is of interest, manifested by the pattern user's preferences for specific forces as illustrated in Figure 4.

CHAPTER 4 PATTERN FORCES AS DECISION CRITERIA

“One of the principal objectives of software engineering is to improve the **quality** of software products. But quality, like beauty, is very much in the eyes of the beholder.” N.E.Fenton [24]

Alluded to previously, the aim of decision analysis is to heighten the decision maker’s “total satisfaction” with the results of the decision. This total satisfaction is influenced by a number of the decision maker’s values. By examining these values, a value tree [28] can be developed from which the hierarchy of criteria is formed. For the purposes of pattern decisions, it is presumed, based on the above quotation, that software quality dominates the decision maker’s values. This chapter discusses, more closely, forces in patterns and introduces a software quality oriented value tree applicable to this thesis, otherwise entitled *force hierarchy*. This force hierarchy is consistent with the hierarchy of criteria discussed in Chapter 2 during the description of the extension to ELECTRE II.

As will be outlined in descriptions of the pattern decision aid, there exists a hierarchy of forces that will be applied by the proposed decision aid. The higher-level forces tend to be abstract and not directly measurable. Pattern user preferences will be applied to this level. The lower level forces will influence these general forces.

4.1 Influences Among Forces

The concept of an influence is important in determining whether a particular criterion is a benefit or cost. The causal loop modeling associated with the operations research field of system dynamics, defines a concept of influence, in that an entity, described by a control variable, can be influenced as the variable in another entity moves up or down. The impact of this influence is either positive or negative corresponding to the direction of

change. A positive impact occurs when changes in two entities are in the same direction and negative if the changes are in the opposite direction.

Figure 5 - Influences Among Forces

Figure 5 illustrates the influence as an association between forces. As way of example, as the fault proneness increases in a pattern then the reliability decreases. To continue, if the reliability decreases then so does the total satisfaction. These influences can be compressed by taking the product of the impacts along the path from fault proneness to total satisfaction. The conclusion drawn is that a measurable value associated with the fault proneness represents a cost criterion. To summarize the two rules, if the compressed impact of a leaf on the root of the value tree is:

1. positive (+), then the measured value is a benefit criterion; and
2. negative (-), then the measured value is a cost criterion.

In pattern decision analysis, another impact is introduced to handle special circumstances, to be introduced in the next section. This special impact is defined as:

3. neutral (). The lower level force has no perceived impact on the higher-level force.

Special treatment of associated forces must be specified.

By relating forces in this manner, a hierarchy of forces can be derived which represents the pattern user's values as they pertain to a pattern. Before articulating such a force hierarchy, a distinction will be made between the types of forces at play in a design pattern.

4.2 Tangible and Intangible Forces

By surveying various patterns, the discussion of forces is either implicitly inferred or explicitly announced. This is often dictated by the format used by the pattern writer. Through most patterns a balance of forces can be identified. While reviewing the discussion of forces in patterns two trends can be identified. The first trend, among most documented pattern solutions, is a tendency to describe the enabling of a capability that possesses a desired force that does not lend itself readily to measurement. Another trend is that those forces that can be measured are not mentioned. With this in mind the force hierarchy, developed for the purposes of the proposed pattern decision analysis, distinguishes between different classes of forces, principally tangible and intangible.

4.2.1 Tangible Forces

The term *tangible force* is introduced figuratively to indicate a force that can be measured. In the physical world if something is tangible it can be touched. Consequently if it can be touched it typically can be measured. In the scenario that a decision is being made concerning a pattern, tangible forces will influence the decision results by their impact on the general forces that have been assigned preferences. Chapter 5 deals specifically with how these forces can be accounted for in assessing a software pattern.

4.2.2 Intangible

The *intangible force* is introduced to represent forces that cannot be measured or, currently there is not a sufficient metric available to indicate its presence. This should not

preclude, as metrics continuously evolve and emerge supported by empirical evidence that an intangible force can transform into a tangible force.

The manner in which intangibles are treated is different from a tangible. Whereas a tangible force can influence a preferred force, the intangible has a neutral impact. If an intangible is supported by a pattern, it does not increase the utility of the pattern. In essence it is present. In the decision procedure the intangible can only be accommodated if the decision maker specifies whether or not such an intangible is required. In turn the pattern acts as a statement of whether or not the intangible force is supported. Pattern decision analysis can then match those intangibles requested to those that a particular pattern complies with. Further discussion concerning the treatment of intangible forces is left to the description of the approach to pattern decision analysis.

With the relationships between forces and a distinction among different types of force established, articulating a force hierarchy relevant to the pattern decision problem is feasible.

4.3 A Hierarchy of Forces Dominated By Software Quality

In examining the traditional approaches to making design decisions in software engineering, product quality is typically evaluated by applying metrics. Although the manifestation of traditional structures for software measurement take a different form, the intent remains to make pattern related decisions based on quality. This intent is confirmed by initiatives in Requirements Engineering to model the linkage between non-functional requirements and pattern elements, namely forces [30]. Traditional software engineering structures and views on quality have been reviewed to draw a representative value tree to be applied to pattern decision analysis.

Throughout the section, hierarchies and related illustrations, pertaining to specific quality factors, represent the material proposed by others. It is considered that the domain of describing and substantiating software quality, along with non-functional requirements, remains an open problem. Although it is expected that presenting a force hierarchy will continue to evolve over time, the treatment of this force hierarchy is the keystone of pattern decision analysis. With this in mind the force hierarchy submitted is representative of one applicable to pattern decision analysis.

The intent of this section is not to present a complete framework for software measurement, as both the target audience and the point in time at which a pattern related decision is made are very specific. Rather, the focus is placed on the identification of metric relationships to product quality. Two typical structures for software measurement will be discussed:

1. Goal/Question/Metric introduced by Basili and Rombach[5] outlined that measurement must answer specific questions relevant to a particular goal. The definition of goals and questions are left very open to allow the tailoring of this approach to the interests of various communities of interests. For example the goals of corporate management may differ from the client, project manager, or design team. Hence, measurement models can be developed to answer the variety of interests by reapplying metrics. Recall that the community of interest applicable to this thesis is isolated to the pattern user expected to be a member of a design team, so the definition of a set of goals/questions will be specific; and
2. Factor/Criteria/Metric introduced by Fenton [24] as part of his measurement framework, related external attributes reflecting user interests concerning the software

product, *quality factors*, to the internal attributes, *criteria*, of the software product. The measurable properties of the software product are likewise traced to these criteria.

An observable trend in non-functional requirements initiatives, related to object-oriented software patterns [30], is to model the chain of non-functional requirements and their contributions to particular goals. This trend is consistent with the former approach to software measurement and tends to be open to the specialization of a particular software project's statement of requirements. The approach proposed for making associated design decisions entails a pattern user initiating a search of design goals with patterns addressing these goals retrieved for possible selection. Based on the manner in which the selected pattern is incorporated, a resulting context leads to subsequent searches for other sets of applicable patterns [30].

Based on the perceived complexity formed in trying to support the navigation of a graph or network of design goals, pattern decision analysis adapts the structured view accompanying the latter perspective on software measurement. In essence, a synthesis of commonly reoccurring quality factors has been applied. However, rather than distinguishing between factors and criteria, the use of forces generalizes these two concepts. That is the force hierarchy developed for the proposed pattern decision analysis relates more readily with tracing metrics to internal design attributes to the desired product characteristics.

Software quality models exist, ranging from the extensive, whereby attempts have been made to present a comprehensive view of software quality,[24] to others that are more concise. In the concise model, a core set of definitions has been identified to which other

sub-factors can be related [52]. Within this core set, some are clearly relevant to a design decision regarding a pattern, but others are dependent on the context of a complete system design. The latter include such issues as usability, functionality, and portability, if they have been defined as follows:

1. Usability. “Usability of a software product is the extent to which the product is convenient and practical to use.”[24]
2. Functionality. Associated definitions are related to the system in scope. That is whether the system’s functional requirements specified have been met. With this in mind, functionality is defined as “... the set of functions derived from the same source; this source may be the set of functions requested, the set of functions delivered, or some other source.”[56]
3. Portability. “The effort required to transfer the program from one hardware and/or software system environment to another.”[43]

Hence, at the exclusion of these three product evaluation issues, those issues that will form the highest level in the force hierarchy include:

1. Performance. “Performance is that attribute of a computer system that characterizes the timeliness of the service delivered by the system.”[3] Where timeliness is typically the response time or throughput associated to a user’s expected workload.[49]
2. Maintainability. The ease by which software can be maintained or changed.
3. Reliability. “The extent to which a program can be expected to perform its intended function with required precision.”[43]

4. Utility. Utility is interpreted as the benefit that a designer receives from a pattern, rather than the functionality a client user requests of a software product [56]. It is introduced to accommodate the intangible forces that will be discussed in this thesis, and serves to isolate these intangibles from the remainder of the tangible forces.

UML: Object Force Hierarchy

Figure 6 - Highest Level of Force Hierarchy

With this in mind the force hierarchy applied to this thesis appears in Figure 6. Subsequent levels within the force hierarchy will not be completely specified. As will be explained in appropriate detail later, an incremental approach has been used in defining the hierarchy. Although, the following sections highlight considerations within the highest level of forces, not all of these issues will be applied in the force hierarchy. In associated diagrams, these issues are depicted in grey. The concept of the force hierarchy has been left open so that it may be defined in more detail in work subsequent to this thesis.

4.3.1 Performance

In examining performance, two types of software resources are typically associated. The manner in which software uses time and space to handle assigned workloads is examined. Within the design, indications of performance can be sought by examining the algorithmic time and space complexity [56], or an analysis of the response time and system throughput for particular workloads [49]. Not only is there typically an internal struggle to balance within the force, but also most other forces typically impact negatively on performance. For example, it is expected that by striving to make more reusable software, it will be at the cost of performance [52]. The force hierarchy is not concerned with peer level influences, but performance is a suitably important issue that it must be accommodated in any decision model. The challenge with the pattern, in order to promote context independence and enhance its potential for reuse, is it will not contain functional details that would be implementation specific. Consequently measuring performance of a pattern poses a challenge.

At this juncture it is expected that standard performance measurement is not widely available for the use in a decision procedure regarding patterns. By investigating some patterns that discuss the negative impact the solution may have on run-time efficiency, the fact that there is increased message indirection [13],[22] is often cited as the cause of this liability. In order to represent performance in the force hierarchy, response time will be considered. Likewise increased message indirection will be used to indicate increased response time. This is applied as an interim remedy until such time as standardized pattern performance analysis makes available more suitable and representative indicators. Figure 7 provides a summary view of this discussion.

UML:Object Performance Force Hierarchy

Figure 7 - Performance

4.3.2 Maintainability

Maintainability is rooted in the amount of effort, in terms of time, that an individual must spend to either fix or change a program. This effort is influenced by the amount of time that an individual spends understanding the program with a view to locating the code that must be fixed or changed. The program or its original manifestation, the design, contributes to this effort by its internal complexity, which has otherwise been termed *psychological complexity* [56]. The whole issue of how to effectively indicate or measure maintainability continues to evolve. However, a classic conflict in software exists in the struggle to make software code more understandable tempered with enhancing the performance of the software. Such conflict or tension remains of interest in a pattern decision problem. In order to represent maintainability within pattern decision analysis, some forces that are typically attributed to maintainability are considered:

1. Size. Size has many connotations [56] with population as the interpretation applied. *Population* is considered a count of design elements, in the case of object-orientation the number of classes and methods. In a program, population is

often attributed to lines of code, but given the stated motivation concerning making decisions during design rather than implementation, this measure is considered outside of the scope of this thesis. The manner in which a pattern design can be measured for population is considered in the following chapter.

2. Reusability. In some communities, reusability is considered an important quality factor in its own right [24], [43], [52]. Patterns inherently promote reusability, but internal to the proposed solution exist elements of reuse. It is these elements and their impact on maintainability that will be of interest in pattern decision analysis. A typical mechanism of object-oriented designs supporting reuse is the application of *inheritance* structures. That is classes lower in the inheritance hierarchy can reuse methods already specified higher in the hierarchy.
3. Cohesion. A cohesive class is one where the properties compliment each other, working together to complete a unit of work. It is suggested that the more a class promotes *cohesion* the easier it is to develop, maintain and reuse that class [11]. The challenge with examining cohesion is that it requires the knowledge of the functional responsibility of a class. A pattern, in promoting reusability, provides a solution with a generic structure where this functional detail is not expected. For this reason cohesion has not been applied to the proposed pattern decision analysis.

UML:Object Maintainability Force Hierarchy

Figure 8 - Maintainability

4.3.3 Reliability

Reliable software performs assigned workloads within specified constraints, be it performance, or precision. Once again, it is difficult to examine precision, or correctness of a design pattern's solution, because of the lack of sufficient context detail. Faults in software are another contributing factor to reliability. Studies have examined the internal design attributes that contribute to fault proneness [12]. The following forces have been associated with design reliability and are summarized in Figure 9:

1. Cohesion. It is perceived that a class that is not cohesive is more likely to promote flaws in the design. A less cohesive class has also been attributed to an increased likelihood that errors will occur during development [16]. However, current cohesion measures do not seem to appropriately account for fault proneness [12].
2. Inheritance. Deep inheritance hierarchies tend to be prone to more faults, due to the potential for deeper classes to not consistently specialize ancestor classes [12]. As

well more extensive inheritance hierarchies make it more difficult to predict the behaviour of classes [16]. With this in mind, it is considered that examining inheritance can serve as an indicator for fault proneness [12].

Figure 9 - Reliability

3. Coupling. *Coupling* is the dependence that one class has on another. Different forms of coupling have been distinguished including: inheritance; aggregation; or navigable associations [33], [56]. With inheritance already accounted for separately, it is the last form of coupling that is of particular concern where a client class relies on a server class to perform a service by passing messages or method invocation. It is considered that increased coupling between classes implies a more complicated debugging effort on the resulting program [12], [17].

4.3.4 Utility

Given that there exist properties in a design that are useful to the designer, but otherwise intangible, accommodating these special considerations in the proposed pattern decision

analysis, poses a challenge. It is considered that the intangibles sought by designers can be classified according to two main dimensions: structure and behaviour. Structure relates to properties concerning the composition of classes within the design. Structural properties can in turn be considered according to data attributes, and functional operations. The behaviour dimension corresponds to the manner in which classes interact in response to an expected impetus. Describing intangibles according to these two dimensions is consistent with a number of object-oriented views [16], [26], [56].

The description of intangibles tends to be an open problem in that, depending on the semantics that a designer or a pattern writer uses, different words may be used to describe the same desired intangible. In order to initiate a vocabulary regarding the description of intangibles, a number of documented patterns were considered. Particular emphasis has been placed on those patterns implicated in testing pattern decision analysis. Intangibles such as elegance, sophistication, or beauty were avoided, stressing instead those that could be directly related to the utility of a pattern. Summarized in Figure 10, the following is a description of some intangible forces listed alphabetically.

Structural Intangibles

1. Data Sharing. In order for a number of classes to complete a particular workload, they may require access to information not possessed locally. Data Sharing implies that accessing necessary information is facilitated by the pattern solution. This intangible is present in the State Member and Exposed State patterns [22];
2. Distribution. The classes implicated in the pattern solution are sufficiently independent with an appropriate communicating behaviour to be distributed among

different sites. This intangible is present in patterns including Broker [13], and Distributed Proxy [47];

3. Extensibility. This intangible implies that the structure is capable of being expanded. It should be noted that all patterns encountered in this thesis possess this intangible to a certain degree. Simple examples are evidenced in the structure diagrams of the State, Visitor, and Proxy patterns depicted in Appendices 2,3 and 4 respectively, where inheritance hierarchies have been left open. Other example patterns that promote this force include Composite [26], Master-Slave [13], and Role Object [6]; and

4. Structural Flexibility. This intangible force relates to the ability to change composition of the software structure at run-time. It may also be entitled changeability. This is exhibited by the Whole-Part [13], Role Object [6] patterns.

UML:Object Utility Force Hierarchy

Figure 10 - Utility

Behavioural Intangibles

1. Behavioural Flexibility. Given different structural compositions, the pattern solution promotes classes behaving differently according to different circumstances. This is promoted by the State Pattern variations[22] and Type Object pattern.
2. Dynamic Transitions. This intangible promotes the ability to transition between different behaviours at run time within the pattern solution. This intangible is present in patterns including State Driven and Owner Driven Transitions[22] and Finite State Machine patterns[58].
3. Transparency. A client class requests a service of a server class. In reality there may exist intermediate classes that will handle the request. Transparency implies, that despite such intermediaries, the client does not have to change its behaviour in order to request the service. Transparency is exhibited in the Broker [13], Composite [26], Decorator [26], Distributed Proxy [47], Null Object [57], Proxy [13] [26], and Two-Way Adapters [26] patterns.

4.4 Summary of Pattern Decision Analysis Force Hierarchy

This chapter has introduced a hierarchy of criteria in the form of a force hierarchy that can be applied to pattern decision analysis. At this juncture, the composition of the force hierarchy is limited to those forces that can be reasonably accounted for within the solution structure of a design pattern. This includes tangible forces, which can be measured, and intangible forces, which cannot be directly measured, but remain pertinent to any approach to making pattern decisions. Figure 11 provides a summary of the forces that will be considered further at this juncture. In order to treat tangible forces, an important ingredient is the manner in which the presence of such forces is measured.

Consequently, the following chapter discusses those metrics that will be applied to indicate tangible forces in the pattern decision analysis approach.

UML:Object Pattern Decision Analysis Force Hierarchy

Figure 11 - Pattern Decision Analysis Force Hierarchy

CHAPTER 5 METRICS

5.1 The Current State of Object-Oriented Design Metrics

The accounting for tangible forces in patterns hinges on applying object-oriented design metrics. The calculation of these metrics should originate from the pattern writer, as measurement assists explaining the rationale of their proposed solution. This may in time be delegated to software. It is envisaged that the software, to calculate specified metrics, would read solution structures. Regardless, the measured results form static values upon which comparison can be run multiple times within a variety of pattern decision problems.

The identification of object-oriented design metrics has paralleled the evolution of object-oriented analysis and design during the 1990's. The intent in most suites of metrics is to measure the software design in order to enhance software quality before programming resources have been allocated. This is accompanied with the perception that Lines of Code (LOC), a typical development metric, does not suitably account for software complexity [35] while the case for measuring the design has been established [3].

In 1994, Chidamber and Kemerer re-introduced a suite of object-oriented design metrics (CK metrics)[16] that is one of the more frequently discussed suites of metrics. Since this second work, efforts have entailed validating [3],[7], advocating [8],[15],[27],[46] or presenting counter positions to their work [33],[35]. Although, the work stemmed from the absence of published empirical metrics data concerning commercial object-oriented applications, to date there continues to exist a widespread shortfall in empirical data supporting related metrics [10]. The lack of solid empirical evidence has not stemmed the introduction of new metrics. In certain studies the potential overlap of what different object-oriented metrics actually measure has been observed [11]. The conclusion derived

here, from these recent studies, is that the field of object-oriented metrics continues to evolve and there does not exist a definitive set of empirically sound metrics that may be applied in pattern decision analysis.

For the purposes of pattern decision analysis, a representative set of metrics, is based on the CK metrics. The CK metrics are applied as the suite covers most tangibles in the force hierarchy with a small set of metrics. Other metrics have been introduced here where CK metric coverage is not sufficient. In some cases selecting these additional metrics was based on recommendations [56], [35], and in other cases based on a critique of the CK metrics [33]. Individual or complete suites of metrics have been incrementally introduced as deficiencies were encountered. An attempt has been made in this thesis to select suites with a limited number of metrics rather than introducing large sets of metrics that may overlap that which is already measured by the CK metrics. Overall Pattern decision analysis is open to accommodate the evolving state of object-oriented design metrics, but the remainder of this chapter will focus on those metrics applied. The six metrics that comprise the CK metrics [16], that will be subsequently outlined using examples, include:

1. Weighted Methods Per Class (WMC). This metric is focused on measuring method complexity;
2. Depth of Inheritance Tree (DIT) which is an inheritance metric focused on the depth of inheritance;
3. Number Of Children (NOC) is a second inheritance metric focused on the breadth of inheritance;

4. Coupling between Object Classes (CBO) a coupling metric;
5. Response For a Class (RFC) is a second metric oriented towards simple coupling;
and
6. Lack of Cohesion in Methods (LCOM) is a cohesion metric, or lack there of.

The other metrics applied include:

1. Number of Concrete Classes. This metric is introduced herein to suitably account for software size, where it has been previously accounted for by number of key classes and number of supporting classes [35];
2. Number of Abstract Classes. Accounts for abstract classes, which is associated with promoting reuse of methods and state data among sub-classes [35];
3. Message Passing Coupling. This metric was proposed as part of a complete definition of coupling metrics [33];
4. Data Abstraction Coupling. Similarly proposed as part of a definition of coupling [33].

Beginning with a discussion concerning the manner in which metrics are implicated in pattern decision analysis, the remainder of this chapter is organized according to the tangible forces that are measured. That is, measuring size, inheritance, coupling, and cohesion. An updated Pattern Decision Analysis Force Hierarchy will summarize the chapter.

5.2 Metrics and their Influence on Forces

At this point the core of pattern decision analysis is comprised of a force hierarchy. However, to be meaningful, there must be an indication to the extent that a particular

force is present in the pattern solution. The metric fulfills this role. However, the metric in turn may have different impacts on the force than it actually indicates. An example is the impact that Lack of Cohesion in Methods (LCOM) has on the tangible force Cohesion. In essence, as an LCOM value increases, the level of cohesion decreases. Thus, there is an influence association between metrics and forces, just as lower forces influence higher-level forces as depicted in Figure 12.

Figure 12 - Metric Influence on Forces

While it has been explained that intangible forces will have a neutral impact on utility, metrics will not possess such neutrality. That is a metric inherently must either have a positive or negative impact on the tangible force it measures.

5.3 Measuring Size

There exist a number of different perspectives on the manner in which an object-oriented design's size can be accounted [56]. In the proposed pattern decision analysis model, one such perspective, population, is considered to represent size. Two common ways in which population can be measured is by applying the Weighted Method Per Class (WMC) [56] and the Number of Concrete Classes which are both benefit criteria based on their influence on population as depicted in Figure 13.

Figure 13 - Population Metrics

5.3.1 Weighted Methods Per Class (WMC)

Equation 10 - Weighted Methods Per Class

$$WMC \text{ for a class} = \sum cx_i, \text{ where } cx_i \text{ is the complexity of the method } i$$

As conveyed in Equation 10, the WMC value is the sum of the perceived complexity of each method in a class. The complexity value is left as an implementation decision. However, as inferred to in their response to initial reviews of the suite of proposed metrics, [17] Chidamber and Kemerer outlined that the value may be 1 for a method that requires additional design effort and 0 for methods that do not. This, in effect, accounts for those methods that require additional design effort. The manner in which complexity will be rated herein is that the WMC is 0 for abstract methods, and for all other methods the value will be 1. There are obvious limitations with this convention as there is no sensitivity to distinguish simple operations, such as accessor methods, from operations responsible for algorithmic units of work. Figure 14 illustrates the resulting WMC assessment of the State Pattern.

Although, there are different perspectives concerning measuring method complexity, the perspective pertinent to its application in conjunction with population is as a predictor of the required time and effort to develop and maintain the class [16].

Figure 14 - WMC for State Pattern Classes

5.3.2 Number of Concrete Classes (NCC)

The number of concrete classes is simply a total count of concrete classes in the associated pattern structure. For example in Figure 14 the total number of concrete classes is 3.

5.3.3 Number of Abstract Classes (NAC)

Whereas the number of concrete classes can be directly attributable to the compiled or run time population of an application, the number of abstract classes is not. Rather, it is suggested that the use of abstract classes does encourage the reuse of methods, while not directly contributing to software size [35]. In Figure 14 the total number of abstract classes is 1.

5.4 Measuring Inheritance

Inheritance can be accounted for in different manners. In general, the metrics described here respectively consider the depth and breadth of the inheritance hierarchy. Other metrics further account for the relative position of a class in the inheritance hierarchy by counting the number of ancestor classes [32]. However, the metrics described here will be sufficient, for the purposes of this thesis, to indicate reusability [16],[20] and fault

proneness [12]. The influence each of these metrics has on inheritance is represented in Figure 15.

Figure 15 - Inheritance Metrics

5.4.1 Depth of Inheritance Tree (DIT)

Equation 11- Depth in Inheritance Tree

class DIT = depth of class in inheritance tree given root is depth 0. [15][16]

For multiple inheritance taking maximum length from class to root further specifies the DIT definition presented in Equation 11. Figure 16 provides an illustrative depiction of applying DIT to the state pattern. The deeper a class in the inheritance hierarchy the more methods it is likely to inherit, and consequently the greater potential for reuse [16]. However, a side effect of deep inheritance is an increase in the unpredictability of the class's behaviour[16].

5.4.2 Number Of Children (NOC)

Equation 12 - Number of Children

Class NOC = Number of immediate subclasses

As defined in Equation 12 and further highlighted in Figure 16 the NOC values give insight into the immediate breadth of inheritance. An increased NOC value implying

more reusability while large NOC values highlighting a requirement for more testing, are perspectives concerning the role of this metric [16].

Figure 16 - Illustration of Applying DIT and NOC

5.5 Measuring Coupling

Given that coupling represents the dependence that one class has on another class, there are three views concerning coupling [33]:

1. Message Passing. Represented by those navigable associations between two classes, one class relies on another to perform an operation or service initiated by message passing.
2. Inheritance. Here the reliance is on an ancestor class for particular services. This is accounted for by applying the already defined DIT and NOC.
3. Data abstraction. One class depends on another class for a particular service, data or operation. Whereas message passing coupling accounts for navigable associations, other associations between classes, for example aggregation and simple associations, can be accounted for within the UML structure diagram.

UML:Object Coupling Metric Influences

Figure 17 - Coupling Metrics

For the purposes of this thesis the inheritance form of coupling will be considered separately from other forms of coupling. The following sections identify metrics that have been applied to measure coupling, as represented in Figure 17.

5.5.1 Coupling between Object Classes (CBO)

Equation 13 - Coupling between Object Classes

$$class\ CBO = count\ of\ classes\ it\ is\ coupled\ to\ [16]$$

Two classes are coupled, as interpreted in the context of Equation 13, if one class uses methods or instance variables of another class [16]. The scope of this metric extends to accounting for inheritance where a class relies on the method or instance variable of an ancestor class in the inheritance hierarchy[15]. Figure 18 provides an illustrative example of applying the CBO metric where the only class to have any values for this and related metrics is Context as can be accounted by the navigable aggregation relationship to a state.

5.5.2 Message Passing Coupling (MPC)

The limitation to CBO is there is no visibility into the specific form of coupling under scrutiny. By distinguishing the various forms of coupling [33],[56], associated metrics [33] have been applied. Applying message passing coupling (MPC), alternatively entitled Coupling Through Message Passing (CTM), partially resolves this deficiency by accounting for message passing. MPC “...is the number of messages sent out from a class” [33]. Figure 18 provides a sample of MPC values applied to the State pattern.

5.5.3 Data Abstraction Coupling (DAC)

Data Abstraction Coupling (DAC), similar to MPC, refines the measurement of coupling by accounting for data abstraction. DAC, alternatively defined as Coupling Through Abstract Data Type (CTA) [32], is limited in scope to classes that have a variable of another abstract data type. Referring to Figure 18 DAC values are “...the number of instances of other classes declared within a class.” [33]

Figure 18 - Illustration of Applying CBO, MPC and DAC

5.5.4 Response for a Class (RFC)

Equation 14 - Response for a Class

$$\begin{aligned} \text{class RFC} &= |RS| \\ \text{where } RS &= \bigcup_{m \in M} \{m, r_m\} \\ \text{given } M &\text{ is the set of methods for a class;} \\ m &\text{ is a particular method; and} \\ r_m &\text{ are those methods called by method } m. \end{aligned}$$

RFC for a class is the size or cardinality of the set of methods that can potentially execute in response to the arrival of a message [16]. In the example, depicted by Figure 19, the RFC for the context class of the state Pattern shows that given a single method entitled `request()` the expected behaviour is that a `handle()` will be invoked in the appropriate state.

UML:Sequence.State ExpectedBehaviour

Figure 19 - Illustration of Applying RFC

Considering the previously mentioned challenge of accounting for performance in pattern structure and behaviour diagrams, it is perceived that RFC can be used to indicate message indirection, based on the expected behaviour of the pattern. Hence, RFC will be

used in two capacities, first as a coupling metric, but as well as a representative indicator for message indirection.

5.6 Measuring Cohesion

Cohesion is related to a class's functional strength, which is the manner in which internal operations work together to achieve a common responsibility [24]. As outlined earlier, as patterns tend not to specify functional details measuring cohesion is currently not appropriate for pattern decision analysis. However, with a cohesion metric available in the CK metric suite, and for the sake of completeness, a description of it is presented and test results in Appendix 2, 3 and 4 include associated values. It should be noted that of all metrics in the CK suite, this metric has come under the most criticism and variation [11],[12],[14],[32].

5.6.1 Lack of Cohesion in Methods (LCOM)

Equation 15 - Lack of Cohesion in Methods[16]

$$class\ LCOM = | M_r(m_i, m_j) | - | M_s(m_i, m_j) |, \text{ if } | M_r(m_i, m_j) | > | M_s(m_i, m_j) |$$

Where: m_i, m_j are a class's methods given $i \neq j$;

$(m_i, m_j) \in M_r(m_i, m_j)$ if m_i, m_j do not use the same instance variables; and

$(m_i, m_j) \in M_s(m_i, m_j)$ if m_i, m_j use the same instance variables.

Whereas most of the CK metrics are straightforward to calculate, LCOM is not as intuitive. As outlined in Equation 15 the LCOM value for a class is the count of method pairs that do not use the same instance variables minus the count of method pairs that do share the same instance variables [15],[16]. Figure 20 outlines two examples of calculating the LCOM value. A symptom of the problem in applying this metric is that a number of pattern classes only define one method per class. Thus, the LCOM will be 0 in these cases. It is considered presumptuous to assign the Context class in Figure 20 an

LCOM value of 0 in the base State pattern, where in the applied State version, presumably cast into a pattern user's context, LCOM equals 1 indicating that it is less cohesive than when in pattern form. In the applied state class diagram of Figure 20, Context has three methods, request1 and request2 both using the reference to the state object. The request3 might not use any instance variable or it may use another variable, but it does not access the state object. Hence $M_d(m_i, m_j) = \{(request1, request3), (request2, request3)\}$ and $M_c(m_i, m_j) = \{(request1, request2)\}$. The Context class LCOM value is equal to $2 - 1 = 1$.

UML:Class.State / State Object

UML:Class.Applied State

Figure 20 - Illustration of Applying LCOM

As represented in Figure 12, it should be noted that LCOM is a cost metric rather than a typical benefit metric. That is, a low LCOM value is considered representative of good cohesion, whereas a high value is considered poor.

5.7 Updated Pattern Decision Analysis Force Hierarchy

Covered within this chapter is a set of metrics that will be applied to indicate how much a particular tangible force is present in a pattern. Figure 21 presents an updated view of the Pattern Force Hierarchy with metrics appropriately represented. With this updated hierarchy, the necessary background material has been presented. Although, not comprehensive, the hierarchy presented is sufficiently representative for applying decision analysis to the pattern decision problem.

Figure 21 - Updated Pattern Force Hierarchy

PART 3 APPROACH

CHAPTER 6 APPROACH TO PATTERN DECISION ANALYSIS

6.1 General Approach

Although, the goal of this thesis is to present an approach to compare candidate patterns, any presented approach must be confirmed within representative pattern decision problems. Furthermore, proposing an approach to pattern decision analysis becomes of value if it can be automated, reducing the amount of effort that a pattern user must expend while trying to identify a suitable pattern. During the conduct of work related to this thesis, three major units of work were conducted. Applying the background material presented, this part describes the manner in which each unit of work was undertaken. The chapters corresponding to these units include:

1. Chapter 6 - Approach to Pattern Decision Analysis. In principle, this chapter outlines the manner in which existing outranking methods have been tailored to present an approach to pattern decision analysis. The associated procedure explained, forms the expected result of this thesis. Although a result, it is presented to set the stage for the remainder of the chapters;
2. Chapter 7 - Approach to Testing Pattern Decision Analysis. Described is the manner in which tests were set up and conducted to confirm the suitability of the proposed approach to pattern decision analysis; and
3. Chapter 8 - Approach to Pattern Decision Aid Tool. This chapter briefly discusses the approach to designing a distributed pattern decision aid, and the design artifacts that were generated as part of the thesis.

The remainder of this chapter deals with pattern decision analysis. It should be observed that while the results of subsequent chapters will be treated separately, the approach to pattern decision analysis represents a key result of this thesis of its own merit.

An incremental and iterative approach was adopted throughout the conduct of work associated with this thesis. Incremental in that the proposed pattern decision analysis was formed as research progressed into different ways to make a decision along with identifying the different pattern decision problems. In turn, as an understanding of pattern decision analysis was forming, limited tests of the approach were set up and conducted. Concurrently, a general concept of the distributed pattern decision aid was established with analysis and limited design initiated to uncover the challenges requiring resolution. To comprehend the target implementation environment, limited development effort was undertaken.

The general approach was iterative in that as problems were experienced during the conduct of one activity, the necessary action was undertaken to remedy the problem, in the next iteration of thesis work. It is certain that at different times in the conduct of the thesis different activities dominated, for instance initial activities were dominated by more research work than tool design, but overall the spirit of this incremental and iterative approach was consistently applied.

6.2 Approach to Pattern Decision Analysis

Before proposing pattern decision analysis, the manner in which this result was formed is provided. Initially, as ELECTRE II emerged as the selected decision analysis method, the original pattern decision problem was considered using the original method. However, two things became apparent, the complexity of conflicting decision objectives that are

inherent to the pattern decision problem, and the number of criteria that would need pattern user input concerning preference weights. With the pattern decision aid in mind, a literature research was initiated to investigate possible methods that resolve these challenges. While reviewing different explanations of ELECTRE II one explanation proposed an extension that dealt with both of these issues [38]. The first challenge could be resolved by organizing the conflicting objectives into a hierarchy. In turn the second issue can be addressed by assigning preferences to the highest level in the hierarchy, thus reducing the requirement for a decision maker's input. Albeit this extension was intended for a hierarchy of two levels, it was expected that it could be tailored to a hierarchy of multiple levels. While tests were conducted of this ELECTRE II extension, in the context of the pattern decision problem, it was observed that associated candidate concordance indices appeared to imply that a candidate would receive credit for assertions that it was preferred to another candidate when the concordance threshold would otherwise preclude such an assertion. Triggered by this observation, a more ELECTRE II oriented approach was formed, reintroducing the application of concordance and discordance thresholds. However, the benefits of the ELECTRE II extension, such as the use of the hierarchy of criteria along with assigning weights at the highest level of the hierarchy, have been applied to the proposed pattern decision analysis. The following is a step-by-step description of that result.

6.3 The Pattern Decision Analysis Approach

The perspectives of the decision maker in question influence the approach to decision making. In the pattern world there are typically two potential actors implicated in patterns, the pattern writer and the pattern user. Although, distinguishing these roles does not preclude one person acting in both, each role does possess a different perspective and

set of values. In this thesis the pattern user is the role examined, whereby a search has been initiated for a candidate pattern to be applied in the design. A raw keyword search may be used to yield and further restrict the candidate patterns under consideration. With the pattern user in mind it is presumed that their total satisfaction is dominated by software quality factors. Hence, the balances sought among forces are those established in Chapter 4 to be accounted by the metrics described in Chapter 5. With this in mind Figure 21 represents the pattern user's hierarchy of values rooted on total satisfaction. This hierarchy will be leveraged into pattern decision analysis.

The description of the approach is consistent with the description of ELECTRE II and the extension. That is, the input, method, and output will be explained, followed by an illustrative example. For the purposes of the example, a sample of the Visitor test is taken due to the relative size of the decision problem compared to the other tests.

6.3.1 Pattern Decision Analysis Input

The specific input for this method includes:

1. The Candidates. The candidates are those patterns to be evaluated during the decision analysis method.
2. The Criteria. In the context of the pattern decision problem the criteria are analogous to the forces that have been organized into a hierarchy rooted on total satisfaction.
3. The Decision Maker's Preference. The pattern user will assign preference to the highest level of forces in the related hierarchy. For the purposes of this thesis this level is represented by Performance, Maintainability, Reliability, and Utility.

4. Concordance Thresholds. Defined in terms consistent with the original ELECTRE II method, two concordance thresholds $\hat{c}_1$ and $\hat{c}_2$ will be required.
5. Discordance Threshold. Likewise, a discordance threshold $\hat{d}$ will be required.

6.3.2 Pattern Decision Analysis Method

Figure 22 - Bottom Up Comparison

The representative force hierarchy in Figure 22, equivalent to that presented in the summary of Chapter 5, will be applied to the pattern decision analysis method. As stressed, this force hierarchy is scalable to reflect different or refined views of software quality and/or refined metrics that measure tangible forces. An issue to keep in mind, which will be discussed further in conjunction with the pattern decision aid, is that as the hierarchy expands, the responsiveness of the associated tool is negatively impacted.

The method will be described in stages consistent with the ELECTRE II method including:

1. Comparing the candidates; and
2. Analyzing the comparison results.

It should be noted that these two stages are repeated, beginning with the deepest forces of the force hierarchy and moving up until the results have been generated from comparison against the highest level of forces. Figure 22 depicts the general process applied to force

hierarchy. The black filled circles indicate that before finishing the comparison at a particular force all results from the lower level must be generated. Introducing this view of the procedure is important to the manner in which the distributed pattern decision aid will calculate the final comparison results.

Stage 1: Comparing the Candidates.

This stage is conducted in three steps:

1. Normalizing the criteria;
2. Calculating pairwise concordance and discordance indices; and
3. Calculating candidate concordance indices.

Figure 23 - Determining Cost or Benefit Criteria

Step 1.1: Normalizing the criteria

Beginning with the metrics in the updated force hierarchy, metric values will be normalized as explained in the extended ELECTRE II method covered in section 2.4. As the comparison proceeds up the force hierarchy, the number of candidates a particular

candidate outranks will be normalized. This latter calculation will be dealt with in Step 2.2. In the case of the number of outranks, it will always be treated as a benefit criteria.

A particular challenge will be detecting whether a particular criteria is a benefit or cost criteria. It is here that the concept of an influence and associated impact is of value. For example, the chain of influences created from CBO to Total Satisfaction, as illustrated in Figure 23, implies that cast in this particular context, the CBO is a cost criteria. That is, the decision maker is seeking a lower CBO value.

Step 1.2: Calculating pairwise concordance and discordance indices

It is at this step that pattern decision analysis diverts from the extension to ELECTRE II and introduces a different extension to ELECTRE II to accommodate a hierarchy of criteria. The pairwise concordance and discordance indices are calculated consistent with the original ELECTRE II. That is Equation 1, and Equation 2 are applied to the normalized values to get the respective pairwise index. Specific points relating to the calculation of each index include:

1. Pairwise Concordance Index. Given that preference weights at lower levels in the force hierarchy are equal, each weight can be assigned a value of 1. The treatment of preference weights in Equation 1 becomes a count of the number of instances candidate a is preferred to candidate b for the sub-criteria divided by the count of metrics or forces implicated in the aggregation of that particular force level. Equation 16 is the amended Equation 1 where k is the force level number in the force hierarchy, given that Total Satisfaction is found at level 0.

Equation 16 - Pattern Pairwise Concordance Index

$$c_k(a, b) = \left\{ \begin{array}{ll} \frac{m_{k+1}}{n_{k+1}} & \begin{array}{l} \text{given:} \\ a \text{ force in hierarchy level } k \geq 1; \\ G_{k+1}, \text{ set of sub-criteria;} \\ j : g_j \in G_{k+1} \\ m_{k+1} = |g_j(a) \geq g_j(b)| \\ n_{k+1} = |G_{k+1}| \end{array} \\ \frac{\sum_{\substack{j: g_j \in G_1 \\ g_j(a) \geq g_j(b)}} p_j}{\sum_{j: g_j \in G_1} p_j} & \begin{array}{l} \text{given:} \\ k = 0; \\ \text{preference weight, } p_j, \text{ for criteria } j \end{array} \end{array} \right.$$

2. Pairwise Discordance Index. An interesting side effect of using normalized criteria values entails the maximum separation, δ , which will always be 1. Thus, the calculating of a pairwise discordance index, transforms into a search for the maximum disagreement that candidate a outranks candidate b .

Equation 17 - Pattern Pairwise Discordance Index

$$d_k(a, b) = \left\{ \begin{array}{ll} 0 : \forall j (g_j(a) \geq g_j(b)) & \begin{array}{l} \text{given:} \\ a \text{ force in hierarchy level } k \geq 1; \\ G_{k+1}, \text{ set of sub-criteria;} \\ j : g_j \in G_{k+1} \end{array} \\ \max_j [g_j(b) - g_j(a)] & \end{array} \right.$$

Step 1.3 Calculating candidate concordance indices.

The original ELECTRE II was interested in conditions where the strong and weak outranking of candidate a over candidate b could be established. Figure 24 depicts the role that Equation 3 and Equation 4 play in filtering assertions of preference. In turn, if Equation 7 is considered figuratively within the ELECTRE II extension of Figure 24, it can be viewed as establishing the net flow of preference between two candidates regardless of the strength of that flow. With a view to accommodate incomparable candidates, the calculation of candidate concordance indices is amended to reintroduce

the decision rules of the original ELECTRE II method. In so doing, the net flow of preference between two candidates can be qualified by the strength of that preference. For example in the pattern decision analysis illustration in Figure 24 candidate *a* has a *strong preference out-flow* of 0.3 to candidate *b* whereas candidate *b* has a *strong preference in-flow* of 0.3 from candidate *a*.

Figure 24 - Updating Candidate Concordance Index

Using the notion of a preference out-flow and in-flow has been used in the exploitation of other outranking methods [54]. Consequently a precedent is not being established herein. For the purposes of pattern decision analysis calculating four candidate concordance indices will be of interest, distinguishing between strong and weak flows and whether it is an in-flow or out-flow.

Equation 18 - Pattern Decision Analysis Candidate Concordance Indices

strong out-flow	$c_{ki}^{so} = \sum_{\substack{i, b \in C \\ b \neq i \\ c_k(i, b) \geq \hat{c}_1 \\ c_k(i, b) - c_k(b, i) \geq 0 \\ d_k(i, b) < d}}^C c_k(i, b) - c_k(b, i)$	Equation 18 - a
weak out-flow	$c_{ki}^{wo} = \sum_{\substack{i, b \in C \\ b \neq i \\ \hat{c}_2 \leq c_k(i, b) < \hat{c}_1 \\ c_k(i, b) - c_k(b, i) \geq 0 \\ d_k(i, b) < d}}^C c_k(i, b) - c_k(b, i)$	Equation 18 - b
strong in-flow	$c_{ki}^{si} = \sum_{\substack{i, b \in C \\ b \neq i \\ c_k(b, i) \geq \hat{c}_1 \\ c_k(b, i) - c_k(i, b) \geq 0 \\ d_k(b, i) < d}}^C c_k(b, i) - c_k(i, b)$	Equation 18 - c
weak in-flow	$c_{ki}^{wi} = \sum_{\substack{i, b \in C \\ b \neq i \\ \hat{c}_2 \leq c_k(b, i) < \hat{c}_1 \\ c_k(b, i) - c_k(i, b) \geq 0 \\ d_k(b, i) < d}}^C c_k(b, i) - c_k(i, b)$	Equation 18 - d

With the calculation of these candidate concordance indices, the comparison stage for a particular force is completed. Although psychologically complex, the intent is not for a person to have to carry out the calculations. Rather the distributed decision aid will handle all related calculations while deriving the final ordering of pattern candidates. With the candidate concordance indices calculated, the following analysis stage will entail considering the associated preference flows similar to the analysis accompanying the original ELECTRE II explanation.

Stage 2: Analyzing the Comparison Results.

This stage is comprised of one step that is repeated as each level in the force hierarchy. This stage marks the completion of the method with the calculation of the number of outranks. The step is entitled:

1. Determine the number of other candidates a particular candidate outranks.

Step 2.1: Determine the number of other candidates a particular candidate outranks.

As explained previously, the original ELECTRE II derived its final ranking by treating each candidate as a vertex in a directed graph and considered the strong out-degree and in-degree of each candidate vertex. The respective weak degrees were considered to break ties. In the pattern decision analysis a particular visit to a force node in the force hierarchy is completed by generating the *number of outranks* a candidate possesses which indicates that the candidate in question has a more dominant strong out-flow and lower strong in-flow than the other candidates. Similarly weak out-flow and in-flow is used to break ties.

This number of outranks forms the relative position of preference for the force in question and in turn forms the basis for normalization in the subsequent visit to a higher-level force in the force hierarchy.

6.3.3 Pattern Decision Analysis Output

When the number of outranks is generated at the Total Satisfaction level in hierarchy the number of outranks for each candidate is normalized one last time. This final normalized value represents the relative order of candidates from highest to lowest, with preferred candidates having a normalized value of 1.00. It must be stressed that this final value is not indicative of the amount that one candidate is better than another, but is useful to generate a final ordering of candidates. That is, if candidate *a* has a final value of 1.00 and candidate *b* has a value of 0.75. This does not imply that candidate *a* is 25% better than candidate *b*. The only inference suggested is that based on the input preference weights it is concluded that candidate *a* will be preferred to candidate *b* by the decision maker.

The explanation of pattern decision analysis is subsequently illustrated by using an isolated portion of the test of pattern decision analysis in the context of the Visitor pattern and alternatives.

6.3.4 Example 3

In order to illustrate the approach to pattern decision analysis a portion of the Visitor pattern and alternatives test is presented. For an abridged explanation of each candidate pattern, refer to Appendix 3. For the purposes of a manageable example the criteria considered are limited to those rooted on performance. This example was selected based on the intuitive tracing of calculations through each step. Although ranking the original input candidate profiles is equivalent to the output, keep in mind that a more complex criteria hierarchy is expected in the pattern decision problem. The remainder of the explanation is divided into parts consistent with the explanation of pattern decision analysis.

Pattern Decision Analysis Input

For the purposes of this example Performance will be assigned a preference weight of 5 by a notional pattern user, with the lower level forces set to a default weight of 1 by the analyst. Table 15 presents the candidate profiles populated with the mean RFC scores accompanied by two concordances and one discordance threshold. It should be noted that beside the criteria label is an indication of the influence the criterion has on the next higher level.

Table 15 - Example 3: Candidate Profiles

		Total Satisfaction
		Performance (+)
		Response Time (-)
		Message Indirection (+)
		RFC(+)
Candidate		
1	Visitor	2.00
2	Acyclic Visitor	1.50
3	Default Visitor	1.63
4	Extrinsic Visitor	2.00

$$\hat{c}_1 = 0.7$$

$$\hat{c}_2 = 0.5$$

$$\hat{d} = 0.1$$

Pattern Decision Analysis Step 1.1

By reading the impacts annotating each force the cost/benefit orientation of the RFC value can be determined. Assigning a value of positive or negative 1 to the respective impact and calculating the product yields the orientation with a negative result implying a cost whereas a positive result implies a benefit. Alternatively, the number of negative impacts can be counted with an odd number of negative impacts implying a cost criterion. In the case of this example, with one negative impact in the chain of influences, RFC in this case is normalized as a cost criterion with normalized values included in Table 16.

Table 16 - Example 3: Normalized values

		Total Satisfaction
		Performance (+)
		Response Time (-)
		Message Indirection (+)
		RFC (+)
Candidate		
1	Visitor	0.00
2	Acyclic Visitor	1.00
3	Default Visitor	0.74
4	Extrinsic Visitor	0.00

Pattern Decision Analysis Step 1.2

Given the index of each candidate in Table 15, the pairwise concordance and discordance values are calculated according to Equation 16 and Equation 17 respectively. The corresponding results are included in Table 17. Note by reading row 2 of the pairwise concordance index table $c(a,b)$, as one might expect from the input, there is an agreement that Acyclic Visitor is preferred to all other candidates. Similarly by reading row 2 in the pairwise discordance index table $d(a,b)$ there is no disagreement that candidate 2 is preferred to all other candidates.

Table 17 - Example 3: Pairwise Concordance and Discordance Values

$c(a,b)$	1	2	3	4
1	-	0.00	0.00	1.00
2	1.00	-	1.00	1.00
3	1.00	0.00	-	1.00
4	1.00	0.00	0.00	-

$d(a,b)$	1	2	3	4
1	-	1.00	0.74	0.00
2	0.00	-	0.00	0.00
3	0.00	0.26	-	0.00
4	0.00	1.00	0.74	-

Pattern Decision Analysis Step 1.3

With step 1.2 completed, the decision rules of Equation 3, Equation 4, as applied in Equation 18, are used to determine the four candidate concordance indices. These candidate concordance indices, denoted by the strength and direction of the net flow, are provided in Table 18. Figure 25 presents an illustrated view of the individual flows, which represent an interim view prior to summation of Equation 18. It can be observed, based on the assertions made in the previous step, that Acyclic Visitor possesses a strong out-flow concordance index of 3.00 whereas it does not have any in-flow, which is consistent with the assertions of step 1.2.

Table 18 - Example 3: Pattern Candidate Concordance Indices

Candidate	Strong Out	Weak Out	Strong In	Weak In
1	0.00	0.00	2.00	0.00
2	3.00	0.00	0.00	0.00
3	2.00	0.00	1.00	0.00
4	0.00	0.00	2.00	0.00

Figure 25 - Example 3: Concordance Out-flows and In-flows

Pattern Decision Analysis Step 2.1

Based on the calculated concordance indices of step 1.3, the results can be assessed and the number of times a particular candidate outranks other candidates calculated. Recall awarding a particular outrank is based on a higher out-flow and lower in-flow using the strong flows as a primary discriminator while the weak flows are used in the case of ties.

Based on the calculated results of Table 19, work pertaining to the message indirection force is complete. These results are then passed up the hierarchy for further consideration, in particular if the next higher level is aggregating conflicting objectives. In the case of this example, because response time only has one child force, the results continue to be carried up the hierarchy. In fact, for the purposes of this example, the

results are carried up to the root of the hierarchy. Hence, the results of Table 19 are used to form the output of the pattern decision analysis.

Table 19 - Example 3: Number of Outranks

	Candidate	# outranks
1	Visitor	1
2	Acyclic Visitor	3
3	Default Visitor	2
4	Extrinsic Visitor	1

Pattern Decision Analysis Output

As mentioned in the explanation of the pattern decision analysis, the results included in Table 19 are normalized one last time and subsequently sorted from highest to lowest value. The results of this action are represented by the sorted list of candidates in Table 20. For the purposes of highlighting different regions in the results, a solid line separates each grouping of candidates.

Table 20 - Example 3: Pattern Decision Analysis Output

	Candidate	
2	Acyclic Visitor	1.00
3	Default Visitor	0.50
1	Visitor	0.00
4	Extrinsic Visitor	0.00

The recommendation drawn from this example is that Acyclic Visitor, when a decision is isolated to considering performance, is the preferred candidate pattern when Visitor, Default and Extrinsic Visitor are alternatives.

This concludes the explanation of the manner in which pattern decision analysis has been approached in conjunction with this thesis. The confirmation of the suitability of pattern decision analysis is outlined in the following chapter.

CHAPTER 7 APPROACH TO TESTING PATTERN DECISION ANALYSIS

7.1 General Approach to Testing

In order to confirm the suitability of the proposed pattern decision analysis, explained previously in Chapter 6, three main tests are performed. The patterns implicated represent each of the three types of pattern decision problems identified in Chapter 3: variations; alternatives; and mixed. This section specifies the test aim and describes the tests performed.

7.1.1 Test Aim

The aim of a test in this thesis is to observe reasonable results generated by pattern decision analysis. A reasonable result is an ordering of candidate patterns with relative positions justified either intuitively, or substantiated.

7.1.2 Description of Tests

The main tests conducted are based on the State, Visitor, and Proxy patterns. The specific tests include:

1. State Pattern and Variations. The pattern language is rooted on the State pattern providing a basic test of pattern decision analysis. The original test, restricted to the State pattern language, serves as an iterative testing environment. That is, as pattern decision analysis evolved the State patterns served as the test where changes in the approach were observed and further challenges identified. The associated patterns include: State; State Member; Pure State; Exposed State; State Driven Transitions; Owner Driven Transitions; and Default State. Within the original State Member pattern three variations are identified. These variations have been split out and are subsequently entitled according to the scope of data

sharing: Isolated; Among Select States; and Common. These variations are also considered in the original test. With the risk that the original test may be biased by the development of pattern decision analysis another pattern is consequently introduced to confirm the results. For this purpose the Strategy pattern is introduced into the original decision problem. The Strategy pattern is different in context, but has the same structure as the State pattern. For an abridged description of each of these patterns refer to Appendix 2.

2. Visitor Pattern and Alternatives. Rather than three of the associated patterns being variations to the Visitor pattern, they each offer alternatives to the basic problem originally resolved. Hence, this test considers the second pattern decision problem identified in Chapter 3. The implicated patterns include: Visitor; Acyclic Visitor; Default Visitor; and Extrinsic Visitor. Accompanying the documented explanations of two of the patterns, Default and Extrinsic Visitor, is a discussion on selecting between these alternatives and the original Visitor pattern. In presenting this discussion a table presents a subjective assessment of each pattern relating to coupling, run-time efficiency and maintainability [41]. This assessment is helpful, setting a benchmark by which results of the pattern decision analysis can be compared. A description of the associated four patterns is contained in Appendix 3.
3. Proxy Pattern and Alternatives. Whereas the other tests implicated patterns all written by a limited group of authors, this test represents a more ad hoc selection. This test has been set up to represent the mixed pattern decision problem. The patterns selected as candidates for this test include: Proxy; Decorator; Null

Object; Distributed Proxy; and Broker. Two of the alternatives are different in context, but similar in structure and another two are variations on the Proxy pattern. Refer to Appendix 4 for a description of the patterns related to this test.

With the tests and associated candidate patterns identified, this chapter introduces the manner in which the candidates are measured, as well as how the criteria, decision maker's preferences, and required thresholds are modeled for all tests. The chapter concludes with a description of the test serials conducted.

7.2 Approach to Measuring Candidates

Each candidate pattern is represented in pattern decision analysis by their measured values. Thus measurement is an important ingredient in testing the decision analysis approach. In principle the structure diagram of each pattern solution was measured. As the majority of metrics applied are class metrics a value representing the solution was required. To this end, although a number of statistics were generated, the value representing the indicator of central tendency, the *mean*, was used where the metric value of each class was divided by the number of classes. The number of classes included both abstract and concrete. The other values calculated include: minimum, median, maximum and standard deviation.

Overall each pattern was measured according to the definition of the metric presented in Chapter 5. In the case of Weighted Method Complexity a simple weighing scheme was applied. Any concrete method received a value of 1 and an abstract method a value of 0. With unity values, WMC becomes a count of methods in a class. This definition is sufficient for the purposes of indicating population in the force hierarchy. A more refined definition of method complexity is left as an implementation decision external to this

thesis. An example of associated tailoring may include distinguishing between accessor methods versus methods that perform more complex operations.

In the case of intangible forces a simple indicator was used. In essence, if the pattern solution supported a particular intangible it received a value of 1, and 0 if it did not. The documented patterns were instrumental to determine if a pattern supported an intangible. In many cases the pattern's discussion of consequences clearly identified the support of an intangible. In other cases, as different semantics were being used, the pattern description was interpreted to determine support for an intangible. In the latter case, an attempt was made to interpret as objectively as possible.

7.3 Approach to Criteria

Given the manner in which the force hierarchy has been stressed throughout the background and approach to pattern decision analysis, it follows that the force hierarchy represented in Figure 21 is applicable to all pattern decision analysis tests.

In essence all intangible forces were requested, hence receiving a preference value of 1. In reality, it is expected that any user will value any different combination of utilities. It is for this reason that assigning weights to intangibles is treated external to the assigning of weights to general forces in the pattern decision aid. However, for the purposes of this thesis, it is sufficient that all utilities are requested.

7.4 Approach to Decision Maker Preferences

It is expected that a user of an automated pattern decision aid, which manifests pattern decision analysis, may possess any one of many different preference profiles. That is one pattern user may value performance more than maintenance, or vice versa. As well a

pattern user may prefer one of these forces to a different degree than another user. The decision maker's preferences were approached from two directions:

1. Simple confirmation. In order to conduct a basic confirmation of pattern decision analysis, four serials were conducted where a preference was set to 1 while all others were set to 0. By setting a preference to 0 it has the effect of switching off that force from further consideration. This isolates the conduct of pattern decision analysis to the only force under consideration. The ordered results can subsequently be intuitively confirmed from the original input measured values for the particular force. An example of this simple confirmation is serial 1 entitled Performance, where the performance preference has been set to a value of 1 and all other high-level force preferences have been set to 0. Similarly serials were run for maintainability, reliability and utility; and
2. Pattern User Profiles. In order to present a reasonable representation of the pattern user in testing pattern decision analysis, a set of three preference profiles were used to represent users with different perspectives. The following section covers the rationale to constructing the associated preference profiles.

7.4.1 The Pattern User's Preference Profile

Over time, the software engineering community has presented various models for describing software quality. These models tend to encompass issues that may be observed during the complete duration of a project. While patterns pose micro-architectures that can be incorporated into a larger design, at this juncture they do not document a complete product. With this in mind it cannot be expected that the range of qualities that a

complete product is expected to possess are applicable to a design pattern under consideration during design.

Figure 26 - Highlighting Points of View

Based on the definition of a force, the various perspectives at play in a possible decision are considered. Figure 26 introduces a conceptual view of the life of a piece of software for the purposes of discussion. In brief, the formulation of a concept through analysis and design will lead to the construction of the software, which upon satisfactory integration, will enter into operation. Once in operation the software will either transition through maintenance to repair deficiencies, or it will transition back through construction for improvements. In time the software will be taken out of service, hence terminating the life of the software.

If the design community as a whole measures the individual success of software, and in turn the design, by how well the software performs during these various states, then it is suggested that a design decision will take into consideration points of view dominated by a decision maker's priorities related to these states. Thus, in order to represent the pattern

user's preferences in testing, three profiles were developed. For each pattern decision analysis test, three decision maker context profiles were considered:

1. Balanced Preferences. This profile holistically considers the software and consequently presumes that all general forces are of equal value to the decision maker.
2. Preference for Operations. A higher weighting is assigned to those general forces that would cause the operation of the end software design to dominate over the maintenance. The associated forces include performance and reliability and an applicable software domain may include real time object-oriented software.
3. Preference for Maintenance. Given that construction is often considered a component of maintenance, a higher weighting was assigned to those general forces that would cause the maintenance of the end software design to dominate over the operation. The associated forces include maintainability, and utility, where reusability has been attributed to maintainability. This profile may be representative of a business application where the software may undergo continuous improvements over its life.

7.5 Approach to Required Thresholds

Recall that three threshold values are required, including two concordance thresholds, $\hat{c}_1$, $\hat{c}_2$, corresponding to strong and weak concordance respectively, and one discordance threshold, $\hat{d}$. Initially, the state pattern test was conducted with $\hat{c}_1$, $\hat{c}_2$ values of 0.80 and 0.60 respectively and $\hat{d}$ was set to 0.1. However, the results of the associated test included a significant number of incomparable candidates. As suggested in the original ELECTRE II explanation of chapter 2, the threshold value can be relaxed and

subsequently $\hat{c}_1$, $\hat{c}_2$ have been calibrated to 0.70 and 0.50 respectively with $\hat{d}$ unchanged. Given these calibrations the results proved more meaningful in the original state test and subsequent tests.

7.6 Approach to Test Serials

As alluded to in the introduction to this chapter, three main tests are executed using the proposed pattern decision analysis approach. Within each of these tests a further seven serials are performed, consistent with the decision maker profiles discussed in Section 7.4. For each serial the same threshold, and candidate measurement values are applied. The only change is the preference profile. Each serial is titled with a descriptive name, representative of the particular decision maker preference profile under consideration. These serials include: performance, maintainability, reliability, utility, balanced, operations, and maintenance. Table 21 provides a summary of the preference weights assigned to the high level forces according to the seven test serials performed.

Table 21 - Test Serial Preference Profiles

Force	Test Serial Preference Weights						
	1	2	3	4	5	6	7
Performance	1	0	0	0	5	7	3
Maintenance Effort	0	1	0	0	5	3	7
Reliability	0	0	1	0	5	7	3
Utility	0	0	0	1	5	3	7

1 – Performance

2 – Maintainability

3 – Reliability

4 – Utility

5 – Balance

6 – Operations

7 – Maintenance

With the necessary input specified, the pattern decision analysis method is applied in accordance with the explanation in Chapter 6. Based on the output of each serial, the implicated patterns are ordered from most appropriate to least appropriate. These results will be discussed in terms of the expected orderings and the actual results in Chapter 9.

CHAPTER 8 APPROACH TO PATTERN DECISION AID DESIGN

8.1 General Approach to Analysis and Design

Consistent with the general approach to this thesis, an incremental and iterative theme was applied to object-oriented analysis and design of the proposed pattern decision aid. Rather than providing definitions of what is implied by analysis or design, the set of activities undertaken during each stage will be outlined. If the construction of the pattern decision aid is viewed from a UML statechart perspective, a traditional incremental-iterative development process [19] has been adapted to explicitly denote the handling of feedback as represented in Figure 27. In principle, a single iteration accounts for a fixed set of increments, where an increment represents a discrete functional unit of development. The iterations continue as long as there remain increments to be incorporated into the software. As main actions are completed in each state, denoted by the outer sub-states, the construction effort transitions to the next state. As challenges are encountered they form feedback to previous and subsequent states in the iteration, thus handling this feedback is considered separately from the main actions.

UML Statechart: Construction of Pattern Decision Aid

Figure 27 - General Approach to Analysis and Design

In total three iterations were initiated as part of the construction of the pattern decision aid in conjunction with this thesis. Development was limited during the third iteration due to the number of challenges encountered and restricted resolution of issues in a timely manner. Further development of the pattern decision aid is consequently left as future work. The iterations can be generalized as follows:

1. Iteration 1 – Identify Domain Classes. Associated work included identifying core functional classes and specifying the basic behaviour and assigning responsibility to each class. The formation of these classes and basic behaviour are subsequently entered into the development environment. Testing of this iteration was limited as the value of associated workloads was not significant.
2. Iteration 2 – Treatment of Data Structures. During the initial iteration the importance of handling pattern related force information was highlighted. In order to incorporate the persistence of associated data a persistence strategy was developed. As this persistence strategy implied using the eXtensible Markup Language (XML) convention, a document type definition (dtd) and sample xml files were developed. These products were tested with a utility, which converts xml files into a view of the corresponding Document Object Model (DOM). With an understanding of the persistence strategy formed, related implications were leveraged into the design of the domain classes.
3. Iteration 3 – Treatment of Domain Behaviour. Whereas there are a number of scenarios that deal with setting up of the necessary pre-conditions to running the actual decision analysis approach, it was decided to focus on the domain behaviour. This decision is justified because focusing on the decision approach behaviour is

more consistent with the thesis goal, versus focusing what are otherwise supporting scenarios. This does not preclude treatment of these scenarios, but this treatment is relatively restricted compared to the core decision analysis scenarios. Development and testing of this iteration have been left as future work.

Further description of the general approach to analysis and design is described in respect to the action states of Figure 27. A description of the approach to specific construction activities, accompanied by an outline of those construction artifacts of principle interest will form the remaining structure of this chapter.

Table 22 - Guide to Assumption / Requirement Catalogue

ID	Title	Description	Source Reference
A – Assumption. F – Functional Requirement. NF - Non Functional Requirement			
id number as specified below	Concise, Descriptive Title	A statement of the description as stated by the source and agreed to by the client. (professor)	Reference of where and when the requirement originated.
001-A	Assumptions	The 000 – 099 series represent the assumptions made during the course of this project.	June 7, 2000
100-F[NF]	System	The 100 – 199 series represent high level system requirements of either the functional [F] or non functional variety [NF]	
200-F[NF]	Decision Analysis	The 200-299 series represent requirements that are related to the specification of the decision analysis approach.	
300-F[NF]	Forces	The 300-399 series represent requirements that are related to the specification of the treatment of pattern forces.	
400-F[NF]	Candidates	The 400-499 series represent requirements that are related to the specification of the candidates.	
500-F[NF]	Context	The 500-599 series represent requirements that are related to the specification of the context of the decision maker's preferences.	
900-F[NF]	Miscellaneous	The 900-999 series represent requirements that are considered miscellaneous and do not fit into the other classifications.	

8.2 Approach to Capturing Requirements.

Associated activities entailed examining the pattern decision analysis approach articulated in Chapter 6. From this examination, functional requirements and assumptions

were identified. Basic information was recorded concerning each of these items. Appendix 6 contains the latest version of this information, organized in tabular form, and organized according to the scheme provided in Table 22. The identification numbers can be used for the purposes of traceability throughout the construction of the pattern decision aid. Accompanying the specification of functional requirements, non-functional requirements were articulated, dominated by the expected implementation environment consistent with the existing pattern repository.

8.3 Approach to Analysis

Based on the specified requirements and assumptions a Use Case diagram is formed to represent the perceived interactions between those entities or *actors* external to the pattern decision aid and the specific use cases internal to the software. It should be noted that these use cases correspond to high-level *scenarios* that form the skeleton of the expected behaviour or workloads handled by the pattern decision aid. In essence the flesh is added to this skeleton through the analysis and design activities.

Along with the proposed Use Case Diagram for the pattern decision aid, the main domain oriented classes are identified. The description of the proposed pattern decision analysis approach is assessed further. Subsequently associations between classes are identified, from which a UML structure diagram can be formed. Accompanying the diagram is a set of Class, Responsibility, Collaboration (CRC) tables [19] that further describe or define the domain classes in terms of responsibilities and their collaborations with other classes. Table 23 presents the structure of associated tables.

Table 23 - Outline of Class Responsibility Collaboration Tables

Class Name: Identifying Name	
Description: A basic description of the class.	Collaborations: A list of other classes that the subject class interacts with either by message passing or invoking operations.
Responsibilities: A listing of responsibilities assigned to the class representative of the class's behaviour.	

Based on the description of classes and the Use Case diagram, Scenario Textual Descriptions (STD) [9] are introduced to begin refinement of the workloads that the pattern decision aid is expected to perform. The domain structure model, CRC and STD tables are those artifacts maintained throughout the construction of the pattern decision aid.

Table 24 - Sample Scenario Textual Description [9]

STD Identifier: USE CASE NAME HERE	Requirement Reference
Description: Enter a brief textual description of the overall objective of the scenario.	
External Actors: Enter the set of external actors that participate in the scenario.	
Precondition: Enter the precondition that must be satisfied in order to enable the execution of the scenario.	
Triggering Event: Enter the triggering event description here.	
Sequence of Responsibilities: R1. Enter responsibility one here. R2. Enter responsibility two here R3. And so on	
Postcondition: Enter the postcondition that must evaluate to true after the execution of the scenario.	
Resulting Event: Enter a set of possible resulting events here.	
Alternatives: (optional) enter a set of alternative scenarios.	
Nonfunctional Requirements: Enter a set of nonfunctional requirements that apply to the scenario (optional)	
Comments: Enter a comment section that may be used by designers as a free format text window to specify different issues related to the scenario.	

8.4 Approach to Design

Based on the analyzed descriptions of the software structure and expected behaviour, the implementation environment is first considered to outline potential implementation challenges that must be resolved. This forms feedback that must be accommodated within the artifacts that have been introduced to this point. Based on the handled feedback, UML

sequence diagrams are formed representing detailed behaviour expected while the software performs an expected workload.

8.5 Approach to Development

Based on the number of challenges encountered in analysis and design related activities, development effort was limited. For the most part the implementation environment was considered to identify challenges that would need further examination. Particular development work was applied to setting up the basic classes in the development environment, and developing with a sample implementation, the persistence strategy to using XML. Recall that one of the thesis objectives is to propose a design. Hence, supporting the proposal of a sound design was stressed rather than focusing on developing a complete product.

8.6 Approach to Testing

Along with limited development effort, testing was similarly restricted. During the second iteration, the proposed XML dtd and sample xml files were tested using a utility (http://java.sun.com/xml/docs/tutorial/dom/4_tree.html) that permits the conversion of an XML file, parsed into a valid Document, the structure of which can be subsequently viewed.

8.7 Approach to Handling Feedback

Throughout each iteration, a number of challenges may be encountered during a particular activity. The challenge will form feedback that has not been otherwise considered within the previous activities or it may form pending work for another iteration. Within the construction of the proposed pattern decision aid, a number of challenges were encountered, some of which required resolution external to the construction effort. This has been alluded to in the general approach that introduced Part

3 of the thesis. It is for this reason that *handling feedback* has been separated from the other states within Figure 27. For instance during the third iteration in the design the manner in which benefit and cost criteria could be identified posed a challenge. In the handling of this feedback, further design and development effort was pre-empted until the concept to deal with the challenge could be formed.

There is no artifact directly related to this activity. Rather the existing artifacts are amended to account for the challenge and its resolution. In cases where the challenge would need to be resolved in a subsequent iteration, it was recorded as a Pending Requirement within the requirements catalogue.

Chapter 10 provides a high level view of the resulting design, in particular focusing on the persistence strategy and the limitations of the proposed design. Refer to Appendix 6 for a more detailed description of the proposed design.

PART 4 RESULTS

CHAPTER 9 TEST RESULTS

This chapter begins with a discussion of observations made while preparing each of the tests. This discussion is relevant to specifying the requisite pattern elements that pattern writers should consider to facilitate pattern comparison. Findings of each test are subsequently organized by a statement of the expected results, a description of observed results and concluded by general comments pertaining to pattern decision analysis. It is recommended that Appendices 2, 3, and 4 be referenced while reviewing the test results discussed in this chapter.

9.1 Test Preparation

It was observed while preparing candidate pattern structures for each test, that certain issues must be resolved before measuring the candidate patterns. These issues pertain to those pattern elements necessary for comparison including structural requirements and modeling the expected behaviour of a particular pattern. The remainder of this section treats each of these issues with discussions concerning structural elements further focused on four specific points.

Generic Structures

While setting up the comparison of State Pattern Variations, included in Appendix 2, the structures within the source reference [22] are illustrative in nature. The value of representing the solution in this manner is acknowledged, as it readily assists expressing and understanding the intent of the solution. However, for the purposes of comparison, it is not suitable. Rather, the solution structure should be as generic as possible, or in other words *context independent*. Within the documented alternatives to Visitor patterns [41],

both illustrative and generic structures are presented, which more readily enables comparison.

Complete Structures

Again, while preparing the comparison of State Pattern Variations, certain documented pattern solutions are related to a specific extension to the original pattern. In order to compare patterns, the solution structure must provide a complete view of the relevant patterns in the pattern language. That is, the structure of a variation must depict the original pattern extended to the specific specialization proposed. However, while presenting complete structures, those classes that are not directly implicated in the pattern should be highlighted to identify that they are not specifically within the solution's scope. A common example is the use of a Client class. In certain cases, such as the Visitor patterns, the presence of a client is useful to tie together the various parts of the pattern. However, as the Client does not directly participate in the performing of the expected behaviour of the pattern, it is excluded from measurement. In the structure diagrams of Appendix 2, 3 and 4, those classes that do not participate directly are presented in grey.

Structures Represent Minimum Solution

A challenge with comparing patterns originates from proposed solution structures presenting a skeleton with functional behaviour added only after selection. In order to promote as unbiased a comparison as possible, it was decided that the minimum structure must be specified for the purposes of comparison. For example, in the State pattern, there must exist two concrete State classes in order to warrant the use of any of the State variations. In the case of the documented Visitor patterns, it was observed that in some alternatives more than one concrete Visitor class was introduced to demonstrate the

differences needed to support different algorithms. However, the Visitor pattern can be relevant with only one concrete Visitor with the understanding that the implemented structure will likely be extended in the future. Thus, the structure used for comparison was reduced in scope to account for the minimum solution.

Structures must be consistent within a Pattern Language

In preparing the State and Proxy tests it was observed that there may exist slight discrepancies between an original pattern that is the root of the language and the remaining patterns in the pattern language. For example, consider an original pattern that defines an interface, which implies no instance variables are declared and no concrete methods are specified. By restricting this definition early in the pattern language it precludes redefining the same class elsewhere in the pattern language as an abstract class declaring some abstract variables, coupled with concrete methods. In order to avoid introducing inappropriate patterns into the pattern decision problem it is assumed that solution structures within a pattern language are consistent. That is, in the case of the example, it is permitted to declare an abstract class in the original pattern, which may be later refined by specifying an interface, but not the opposite. This issue may be revisited as part of future work, and if it is found that introducing inconsistency does not prejudice decision analysis results, this condition may be relaxed.

Expected Behaviour

With the metrics applied to measure the patterns it was determined that sufficient information is available in the UML structure diagrams. However, in measuring coupling and message passing a more clear measurement was facilitated by considering UML Sequence Diagrams depicting the expected behaviour of the pattern's solution. Expected

behaviour infers that in resolving the problem the pattern classes will interact in a particular sequence, forming a typical workload. Using sequence diagrams in this manner proved particularly useful in measuring RFC and MPC values.

9.2 Test 1: Comparison of State Pattern Variations

Recall that this comparison was composed of two sub-tests, hereafter entitled Test 1.a to denote the original State Pattern and Variations test, and Test 1.b representing the State Pattern, Variations and Strategy Test. To present the results of each test the result section will be organized with a discussion of the expected results, observed results, and issues deemed worthy of comment.

9.2.1 Test 1.a Expected Results

The improvements Pure State proposes over State cannot be measured in the solution structure. The benefit that distinguishes Pure State from State Object is that the number of State objects taking memory space during run-time is limited. Visibility of this benefit is not feasible by measuring the design. Based on the metrics applied here, all metric values are the same so it is expected the two patterns will be preferred the same amount. Hence, indifference is expected between State and Pure State.

Two extremes can be identified in the measured profiles of candidates. On one extreme the State pattern, limited in behaviour and structure, versus the other extreme the State and Owner Driven Transition patterns, which are rich in behaviour and imply a more extensive structure in the design. It is consequently expected that the State pattern will be preferred in contexts where performance dominates over other concerns, whereas it is expected that the other extreme will be preferred where utility dominates the decision context.

Another pattern for which preference will likely not be expressed is Exposed State. It is anticipated that the increased coupling and more message indirection required in a client's direct access to a State object will be to the detriment of Exposed State's rating in most decision problems involving the state pattern language.

Table 25 - Test Serial Performance: Input versus Output

	Performance(+)		Results	
	Message Indirection (-)		Number of Outranks	Rank
	RFC (+)	Rank		
Candidate	Normalized		Normalized	
State Object	1.00	1	1.00	1
State Member.isolated	0.88	3	0.86	3
State Member.amongSelectStates	0.82	4	0.71	4
State Member.common	0.49	7	0.29	7
Pure State	1.00	1	1.00	1
Exposed State	0.52	6	0.43	6
State Driven Transitions	0.00	9	0.00	9
Owner Driven Transition	0.03	8	0.14	8
Default State	0.76	5	0.57	5

9.2.2 Test 1.a Observed Results

The intent of test serials 1 - 4 is to simply confirm that the decision analysis approach performs appropriately. By isolating each main criteria the proposed pattern decision analysis can be confirmed more readily than in the pattern user profile serials. In order to illustrate this confirmation consider the first Test serial where Performance is the only criteria upon which a decision is to be based. Table 25 provides an extraction of the normalized measured values input into this first test serial with an associated ranking. This ranking can be compared to the results generated by applying pattern decision

analysis with its corresponding ranking. It is observed that pattern decision analysis in this example treats the given decision problem input appropriately in the results.

In examining test serial 2, Maintainability, the role of inheritance can be observed to temper the results. It is based on Default State's dominance while State and Pure State cannot establish a clear position in regard to reusability that attribute to the test results of Appendix 2. Again in test serial 3, Reliability, inheritance plays a role to heighten State Member when data sharing is limited to an isolated state. Furthermore, State Member (isolated) rates well in regard to coupling despite State and Pure State's overall dominance in this regard. The comparison results of test serial 4, Utility, are intuitive based on the original input values. Whereas, most other patterns support an intangible force, State and Pure State do not support an intangible that would distinguish them from the others.

With these confirmation test serials conducted, the results of the artificial representation of the decision maker's preference profile can be more readily confirmed. With equal preference assigned to forces in the test serial 5, Balanced, the State Object, Isolated State Member Isolated and Pure State are rated equally. It should be noted that although State Object and Pure State will outrank the same set of candidates because their metric values are the same, State Member Isolated does not outrank the same set. Thus the result implies there is not sufficient evidence to distinguish between the State / Pure grouping and the isolated State Member. In particular, when the interim results generated by pattern decision analysis are placed under closer scrutiny it is observed that while the State / Pure grouping dominate in terms of performance and are rated fairly high with regards to maintainability, State Member dominates reliability and is part of the candidate

coalition that are preferred with respect to utility. On the other side of the spectrum, those candidates that rate as less appropriate include, as expected, State Driven Transitions, Owner Driven Transitions, and Exposed State.

While considering the results of test serial 6, Operations, the equality of State, isolated State Member and Pure State can again be attributed to their respective ratings in test serials 1 and 3, given that performance and reliability are assigned more preference in the decision maker's profile, than maintainability and utility. Once again, as expected, Exposed State, State Driven Transitions and Owner Driven Transitions rate as less appropriate for inclusion in the target design.

Table 26 - Strong and Weak Flows for Test 1.a

Candidate	Strong Out	Weak Out	Strong In	Weak In
State Object	0.00	1.50	0.40	0.00
State Member.isolated	1.25	0.00	0.15	0.00
State Member.amongSelectStates	0.50	0.00	0.60	0.60
State Member.common	0.35	0.00	0.35	0.60
Pure State	0.00	1.50	0.40	0.00
Exposed State	0.00	0.00	2.10	0.60
State Driven Transitions	0.20	0.00	0.65	0.60
Owner Driven Transition	0.20	0.00	0.65	0.60
Default State	3.20	0.00	0.00	0.00

The final test serial 7 - Maintenance, provides interesting results recalling that preference is allocated more to maintainability and utility. The latter force promotes the State Member variations that are rated below the leader, Default State.

Another result in test serial 7 entails the more complex patterns State and Owner Driven Transitions grouped with State and Pure State. Whereas examining the decision results from test serial 1 – 4 help to confirm the results of test serial 5 and 6, this is not the case in this last serial. By scanning through the resulting flows in Table 26 one might question how the basic patterns can fall into the same grouping as the more complex. However,

through the analysis stage, as the number of outranks is the discriminating factor, it can be observed that each of these four patterns only outrank one other, Exposed State. Note that as the State Object flows are compared to the State Driven Transition, it cannot be established which of the two is preferred based on strong flows. Further analysis of the candidate results is pre-empted and neither can be credited for outranking the other.

9.2.3 Test 1.a Comments

In all cases State Object and Pure State were rated together. This acts as a basic confirmation in all serials that pattern decision analysis performs appropriately. Another trend in all serials is the relative closeness of State and Owner Driven Transitions. The two either rate the same, or one is rated higher than another. This trend is confirmed in the original description of the State patterns, where it is noted that the two patterns are in tension [22]. Although, the originally documented tension is driven by the assignment of responsibility, in this case the tension is force driven. For example, when considering Performance, Owner Driven Transitions rates higher than State Driven Transitions, but with respect to Reliability the opposite can be observed. This tension has an interesting side effect. In the pattern user profile serials, the two patterns rate the same. Consequently it would be left to the decision maker to make the final distinction concerning this tension by reading each pattern in more detail.

In serials 1 – 6 the results are relatively consistent with the expected results. The general trend that appeared is that those patterns that are closer to the original pattern, in this case the State Object and Pure State patterns as well as the isolated State Member, will naturally tend to rate higher than “richer” patterns which offer more utility and more complex behaviour. As a consequence the resulting list can be viewed as a reading list of

patterns. That is, the recommended action to a pattern user, subsequent to the receipt of comparison results, should be to start their reading from the top of the list to the bottom. This will expose the pattern user to the core features of the pattern language, which will be of further use when trying to understand the more rich variations towards the end of the list.

In the case of serial 7, the expected grouping of results did not match those actually generated. Regardless, the results are encouraging. Although specific and select intangibles were not required, the heightened emphasis on the Utility of the pattern has the effect of raising the profile of the “richer” patterns. Furthermore, by analyzing flows, the possibility that two candidates are incomparable has been accommodated. Overall this test is a success in that the results are reasonable and can be substantiated.

9.2.4 Test 1.b Expected Results

By extending the patterns under consideration in Test 1.a to include the Strategy Pattern, two results are expected. First, the Strategy pattern is expected to rate the same as State Object and Pure State, based on their similar structures despite essentially different contexts. Otherwise, it is expected that the results will be very similar to Test 1.a.

9.2.5 Test 1.b Observed Results

By considering the test results in Appendix 2, it can be observed that in all test serials Strategy rates the same as the State Object and Pure State patterns. For the most part the results remain unchanged from Test 1.a. There do exist different orderings, but this can be explained by the effect of introducing another candidate. Essentially, the introduction of the Strategy pattern either reinforces the preference for a candidate or it reinforces the dissenting position. Given this effect, the change is subtle in all cases where there is a

difference in ordering. For example in the case of serial 7, rather than State Member 2 rating in a middle grouping it is demoted into the second last grouping.

9.2.6 Test 1.b Comments

The comments concerning Test 1.a remain applicable. However, an additional point concerns differences in pattern decision problems. Although, it has been stressed that the results of a particular decision problem depend on the pattern user's context driven preferences, the results depend as well on the particular candidate patterns under consideration.

9.3 Test 2: Comparison of Visitor Pattern Alternatives

9.3.1 Test 2 Expected Results

During tests pertaining to variations within a pattern language, it is reasonable to expect that the root pattern in the language will be rated higher, as this pattern provides the minimum solution to the problem within the context of the pattern language. However, in the case of alternatives, competing views are offered to the original solution. Hence, it is expected that in this test the original Visitor pattern will not necessarily rate the highest according to the different perspectives of each test serial.

Table 27 - Visitor Pattern Tradeoffs [41]

Pattern	Coupling Between Visitors and Elements	Runtime Efficiency	Ease of Maintenance
Standard Visitor	Very High	High	Low
Default Visitor	High	High	Medium
Extrinsic Visitor	Low	Low	High

In the case of the Visitor, Default Visitor and Extrinsic Visitor, the documented patterns[41] include a description of tradeoffs useful for choosing the use of a particular pattern. Table 27 is an extraction of the related tradeoffs identified. However, this analysis is not substantiated in the accompanying discussion. This discussion is pertinent

to this test as it serves as a benchmark upon which the pattern decision analysis results can be compared.

9.3.2 Test 2 Observed Results

The associated results discussed in this section are included at the end of Appendix 3. In the confirmation serials 1 and 4, the results are self-explanatory while serials 2 and 3 pose issues related to the respective placement of Default Visitor and Extrinsic Visitor.

Table 28 - Results of Serial 2

Serial 2 – Maintainability	
Visitor	1.00
Acyclic Visitor	0.67
Default Visitor	0.33
Extrinsic Visitor	0.00

In test serial 2, Default Visitor is the dominant candidate from the perspective of reusability. However, it does not fare well based on its size. It is the imbalance between these forces for which Default Visitor is placed lower than the Visitor and Acyclic Visitor patterns in the results of Table 28. In both of the higher rated patterns, they are more balanced in regard to both forces. For example the Acyclic Visitor pattern outranks two candidates in terms of each of the two forces, whereas the Default Visitor doesn't outrank any candidates in terms of size, but does outrank all other candidates in terms of reusability.

Continuing with serial 2, recall it was anticipated, from Table 27, that Extrinsic Visitor should rate highest in regards to maintainability. However, as claims in Table 27 are not substantiated, it is not possible to establish a similar result in this test. It is perceived, because Table 27 highlights coupling, and given some communities of interest suggest that as coupling increases maintainability decreases, this may have influenced the given

rating for Extrinsic Visitor's ease of maintenance Table 27. Note, in the force hierarchy used during these tests, the contribution that coupling makes to maintainability has not been mapped. Even within coupling, related measures applied to this thesis create discrepancies. For example, compared against three coupling metrics, Extrinsic Visitor has the lowest values of the candidates and would likely be preferred. However, the Extrinsic Visitor value for RFC contradicts the other assertions that it has low coupling. Hence, even though pattern decision analysis does order the results according to the input values, there exists a requirement to further calibrate the force hierarchy in terms of force relationships and metrics indicating tangible forces.

In the case of Serial 3, where coupling has been applied, it can be observed that based on Extrinsic Visitor's relatively low coupling it dominates the comparison when isolated to reliability. The finding at this juncture is that the test results are consistent with the trade-off values of Table 27 for coupling and runtime efficiency. However, such consistency does not pertain to ease of maintenance. Calibrating the force hierarchy to account for the relationship between coupling and maintainability is left as future work to this thesis.

The issue of Extrinsic Visitor's ease of maintenance further impacts the test serials related to pattern user profiles. In particular in serial 6, Extrinsic Visitor is rated second based on its dominance in regard to reliability and in serial 7 it falls into the last grouping because of its low rating in regard to maintainability. Beyond the issues posed by the Extrinsic Visitor, the overall trend is that Acyclic Visitor and Visitor tend to consistently fall near the top of the ordered results. Rather than asserting that these patterns are the recommended solutions, it is suggested that the results form a reading list whereby the pattern user still must decide on the most suitable pattern for their particular context.

9.3.3 Test 2 Comments

The issue concerning the Extrinsic Visitor may challenge the credibility of pattern decision analysis. However, the issue only highlights a weakness in the force hierarchy, and not in the overall approach. In fact pattern decision analysis provides reasonable results despite this issue. It was expected that because alternatives to Visitor were under consideration that Visitor would not always be preferred to the other candidates. Indeed, this is the case in serials 1, 3, 5, and 6 where the Visitor Pattern is not the top rated candidate. Additionally, given the assessed tradeoffs in Table 27, the associated benchmarks set in terms of coupling and runtime efficiency are consistent with the measured values in test 2. With the results substantiated, test 2 of pattern decision analysis, applied to the pattern alternatives decision problem, is considered a conditional success.

9.4 Test 3: Comparison of Proxy Pattern Alternatives

9.4.1 Test 3 Expected Results

With this test, the mixed pattern decision problem is examined. It is expected that a clear delineation between the alternative patterns, Proxy, Null Object and Decorator will be observed compared to the variations on the Proxy pattern: Broker and Distributed Proxy. Just as in the alternatives of test 2, it is expected that among Proxy, Null Object and Decorator, different patterns will be rated highest depending on the preference profile. That is, it is expected that Proxy will typically outrank Broker and Distributed Proxy because they are variations, but Proxy will not always outrank its competitors, Null Object and Decorator.

9.4.2 Test 3 Observed Results

Similar to the other test results, find at the end of Appendix 4 the results for test 3. In the simple confirmation serials 1, 2, and 4, the results are consistent with the normalized input values. In serial 3, Reliability, the Broker pattern is positioned higher than expected. The results of serial 3 are discriminated by Null Object's clear dominance in terms of coupling followed by Broker, which makes limited use of inheritance resulting in its dominance over other candidates from this point of view.

Considering pattern user profile test serials, Decorator receives a higher rating in the balanced profile because of its superiority in utility and maintainability. However, in serial 6, the Operations scenario, Null Object is either of equal or superior rating to Proxy and Decorator in performance and reliability. In these same two serials it is observed that the "richer" patterns fall to the bottom of the list as expected. Furthermore, in serial 6, the Broker's higher rating in terms of reliability can be traced to its better rating than Distributed Proxy. What is less apparent is the high rating of Distributed Proxy in Serial 7, Maintenance, not only against Broker but also to the other alternatives. Once again the role of utility comes into play where Broker and Distributed Proxy are dominant in this regard. In turn Distributed Proxy is preferred over Broker in terms of maintainability, which affects the final placement.

9.4.3 Test 3 Comments

Whereas certain trends could be observed in the previous two tests, the mixed problem cannot be similarly generalized. In this pattern decision problem the results are truly dependent on the candidates and the preferences that have been assigned to the forces. Although in some cases there is a separation between the grouping of alternatives and variations, this does not always apply. In particular, in serials 3 and 7 it was observed that

each of the variations managed to rate higher than some of the alternatives. What can be confirmed is that the test provided reasonable results thus concluding that the pattern decision analysis will provide ordered results based on the input values.

9.5 Summary of Findings

Throughout all three tests of the proposed pattern decision analysis, the approach provided reasonable results that were either intuitive from the input values or subsequently substantiated. For this reason, the testing aim, to observe reasonable results generated when applying the pattern decision analysis approach, has been met. However, observations further raise some issues that require investigation. In particular the composition of the force hierarchy requires further calibration. This issue has been acknowledged from the outset and hence further evolution of the force hierarchy is left as future work. With this issue raised, the application of pattern decision analysis in its current form is cautioned. Certainly in providing the pattern user a suggested ordered reading list, pattern decision analysis is appropriate. However, pattern decision analysis, as a mechanism to preclude further application of a particular pattern is not suitable: That is removing a pattern from the pattern repository based on a pattern decision analysis result is not recommended.

CHAPTER 10 PATTERN DECISION AID DESIGN

The design artifacts outlined in Chapter 8 are presented in Appendix 6. This chapter focuses on specific elements of the design. In particular a high level view regarding the pattern decision aid, the software domain structure, an outline of the persistence strategy suited to the proposed design and the expected behaviour of the software. Additionally, an assessment of the expected behaviour is expressed in terms of time complexity.

Figure 28 - Use Case for Pattern Decision Aid

In order to temper the remainder of the discussion herein, Figure 28 has been extracted from Appendix 6. In general, it is anticipated that the decision aid will be used subsequent to a keyword search for potential patterns, with the user selecting patterns from the search results. These selected patterns become candidate patterns. Once the decision aid has been initiated, there are some scenarios that must execute to set up the software before the actual decision analysis is performed. Although these scenarios have been treated in Appendix 6, they have not been of prime focus during the current design effort. Given the thesis goal, the focus of design effort has been on the scenarios: Pattern Decision Analysis; Compare Candidate Pattern Profiles; and Analyze Comparison Results. For a more detailed description of each of these Use Cases refer to Appendix 6.

UML Model: Pattern Decision Aid

Figure 29 - Model View of Pattern Decision Aid

10.1 Pattern Decision Aid Structure

In order to place the pattern decision aid into architectural context, the envisaged tool is composed of three main packages as illustrated in Figure 29: presentation; domain; and persistence.

Presentation. The responsibilities of associated classes pertain to the presentation of information to the user. The main class of concern is the DecisionAidViewer but it is also expected that utilities for converting data to a HTML format will be required. Currently this architectural package of classes is outside the focus of work.

Domain. Related classes act to carry out the logic that drives the particular application. In this case the logic pertains to executing the proposed pattern decision analysis. Understandably specifying the package classes has formed the main emphasis of the design effort.

Persistence. The need to store data concerning patterns has been identified very early in the design. Thus the persistence of candidate pattern information is crucial to the pattern decision aid. The requirement arises from the nature of metric values indicating the forces at play within individual patterns. Given that a pattern can be implicated in many different decision problems while underlying metric values remain static, persistence of that information must be handled external to the pattern decision aid. The existing pattern repository makes use of the eXtensible Markup Language (XML) to structure pattern documents. Given the static nature of the information implicated in the pattern decision problem, XML was selected based on its suitability and to remain consistent with existing pattern repository capabilities.

Although, the composite classes of the Domain were considered initially and a high level description will be presented next, the need to examine the implications of using XML on the domain behaviour formed the second iteration of work. It is for this reason that the persistence strategy is presented before the description of domain behaviour in this chapter.

10.2 Pattern Decision Aid Domain Structure

Although Appendix 6 includes a UML Class diagram describing the main domain classes, key responsibilities are highlighted here. With the domain behaviour described separately, attention is drawn to two responsibility assignments:

1. Assigning Comparisons. Pattern decision analysis is formed around the comparison of pairs of candidates. While the manner in which this is managed is treated separately, a candidate will be assigned another candidate to compare it to. The candidate assigned the comparison will, without bias, also consider the point of view of the other candidate. That is, considering Table 29, a comparison with candidate 2 (cdt2), can be assigned to candidate 1 (cdt1). With this assignment cdt1 must calculate x , the concordance that cdt1 is preferred to cdt2 as well as y , the concordance that cdt2 is preferred to cdt1. The results of these comparisons, triggered by cdt1, will be updated in each of the two candidates.

Table 29 - Pairwise Concordance for any two Candidates

$c(a,b)$	1	2
1	-	x
2	y	-

2. Handling Comparison. While coordinating comparison is assigned to the candidate, the actual handling of the comparison is dependent on the type of force being compared. With this in mind the actual responsibility to handle a comparison is assigned to the ForceType class. Figure 30 provides a view of the Domain structure restricted to the relationships between Force and ForceType classes. Note that arguments for each method call have not been presented. A Tangible will need knowledge concerning the metrics that contribute to it, whereas an Intangible only requires knowledge concerning whether or not the intangible is supported. In

handling the comparison, the Tangible must consider all of the metrics before providing a result. Whereas the Intangible refers to the indicator that it is or is not supported. Although the General force is typically a parent to Tangible and Intangible forces, its behaviour extends the behaviour of the Tangible. Considering the force hierarchy applied to this thesis, Reusability is a general force with a metric and a general force contributing to it. In terms of behaviour and this example, the force must consider the metrics and then the child forces. It is for this reason that a General force inherits its behaviour from the Tangible force.

Figure 30 – Force Type

For a description of other classes in the Domain model, Class Responsibility Collaboration (CRC) tables have been provided in Appendix 6.

10.3 Persistence Strategy For The Pattern Decision Aid

A number of issues were raised early while considering the automation of pattern decision analysis. As illustrated in Figure 31 there exist complex inter-relationships between: the force hierarchy; the candidate patterns under consideration; and the decision

maker's context. The force hierarchy forms the general map that is followed while conducting pattern decision analysis. Recall the approach works from the bottom of the hierarchy upwards and thus the force hierarchy will be central to the execution of the pattern decision aid. The candidate patterns under consideration possess static metric values, which contribute to particular forces. Finally a decision maker's context is manifested by a profile of preferences for the forces in the hierarchy. Note in the latter case the context will change from one decision problem to another.

Figure 31 - Inter-relationship between Pattern Decision Analysis Objects

In order to resolve the relationship complexity, a persistence strategy was applied whereby knowledge concerning the structure of the force hierarchy is separated from the knowledge concerning specific candidate and context associations regarding measured values and preferences respectively. This separation is further warranted by the force hierarchy's specification of a common semantics regarding the force information

applicable to the pattern repository, leaving individual patterns to record pertinent metric values.

Figure 32 - Untangling the Association Web

Figure 32 illustrates the resulting separation. Using a unique identifier concerning the force as common semantics, the force hierarchy structure will be stored separately from the context and candidates. The context will be constructed at run-time reflecting a mapping of decision maker preferences to the relevant forces. Each candidate will

possess a mapping of metric values to the forces that the metric indicates. These profiles will be stored external to the pattern decision aid.

10.3.1 Implementing the Persistence Strategy using XML

As alluded to previously, there exists a XML specification for the storage of the pattern document structure [25]. Currently force information is treated therein as a description of a particular force at play in the pattern. It is not proposed that use of this textual description be dropped from the current structure. Rather the discussion will assist the pattern user to further qualify their decision. Hence, it is expected that force related information, for the purposes of decision analysis, shadows the existing pattern documents. Albeit the objects illustrated in Figure 33 are .xml files, each individual pattern will have a force shadow capturing static metric data, while a common force shadow captures the force hierarchy structure. The common force shadow is useful to ensure that the individual force shadow files are synchronized to reflect updates in the hierarchy or metrics. Handling of this synchronization is an issue left as future work to this thesis. For the purposes of this thesis it has been assumed that the candidate pattern force shadows are consistent with the common force shadow.

UML Object: PatternDecisionAid.Persistence

Figure 33 - XML Force Shadows Applied to Pattern Repository

Appendix 5 contains a guide to XML and a definition of the related tags used to define the structure internal to these force shadows. Accompanying this definition is a sample of the force hierarchy applicable to this thesis, cast as a common force shadow. As well a sample of a pattern's force shadow is also presented.

10.4 Pattern Decision Aid Expected Behaviour

The implementation of pattern decision analysis is not intended for human application. The number of computations required to move up the hierarchy in generating a final result is tedious. However, the computations required in pattern decision analysis are repeatable and the behaviour, based on force type can be reused from one decision problem to another as long as the force hierarchy structure and measured patterns are stored external to the pattern decision aid. These properties lend well to automating pattern decision analysis as a distributed decision aid. Consistent with the description of the approach in Chapter 6, the expected behaviour of the pattern decision aid will entail:

1. Input. As part of setting up the decision problem, which entails constructing the force hierarchy and getting candidate pattern profiles, the user will be prompted for the preferences and the intangibles required. Recall from Figure 28 that this has not formed the focus of the design effort. For an initial description of related scenarios refer to Appendix 6.
2. Method. The pattern decision analysis method of Chapter 6 forms the main focus of the design effort. The expected behaviour entails moving up the force hierarchy with candidate pairs compared and results analyzed for each force in the hierarchy. This also forms the main domain behaviour that will be described in the following section.

3. Output. Once the final comparison has yielded results for the complete force hierarchy, the candidates can be ranked and the ordered list of candidates displayed to the user. Here again this has not formed the focus of the design effort and will be dealt with only in Appendix 6.

10.4.1 Pattern Decision Aid Domain Expected Behaviour

Given that the pattern decision aid has been set up with candidate patterns instantiated and a context reflecting user input preferences, the focus in this discussion is on the execution of the pattern decision analysis method. In order to affect the bottom up processing of the force hierarchy, a post order traversal is conducted, as illustrated in Figure 34.

Figure 34 - General Process for Pattern Decision Analysis

At each visit during the traversal, denoted by the black filled circles in Figure 34, pairwise comparisons will be initiated among all candidates. As outlined in the description of the pattern decision aid domain structure (Section 10.2), the specific manner in which the comparison is handled is dependent on the force type. At the end of the pairwise comparisons each candidate's force instance, of the node visited, will have

been updated with values for each of the strong and weak flows. Upon completion of the pairwise comparison, the forceNode then initiates a pairwise analysis in which the strength of the flows will be compared between candidates to determine if a candidate outranks another candidate. At the end of the analysis each candidate will have a force object updated with the number of outranks. The calculation of this value signals the end of the Analysis stage, and in turn a particular iteration of the pattern decision analysis approach. For a description of the detailed behaviour for the Comparing Candidates and Analysis of Comparison Result scenarios, refer to the associated sections of Appendix 6.

The traversal continues until the root node has been processed. There is a slight difference in behaviour at the root node, entitled “total satisfaction” in this thesis. The difference is based on the behaviour of a general force type, but the specific preference weights that are included in the context have to be considered when calculating the pairwise concordance values. This specialization is rather minimal and has not been explicitly modeled in Appendix 6. To conclude this chapter, an assessment of the value and limitations of the proposed design is provided.

10.5 Assessment of the Proposed Design

The main benefit promoted by the proposed pattern decision aid is its open design. That is, with a hierarchy of decision criteria that can be redefined or evolved depending on a different pattern practitioner’s view on software qualities, there does not exist a definitive force hierarchy around which the pattern decision aid can be constructed. By specifying the hierarchy structure external to the software, at run time the pattern decision aid is configured to reflect the up-to-date force hierarchy.

Given that the pattern decision aid is to be implemented in a distributed environment, the amount of time required to handle a submitted comparison is a concern. Given that a force hierarchy can be scaled to evolve as software quality and non-functional requirement models are specified, there exists a risk that, by accommodating every discrete force, the response time of a pattern decision aid will be significantly affected.

10.5.1 The Complexity of the Proposed Design

This section provides an assessment of the implications of the proposed design with regards to time complexity. The asymptotic analysis accompanying this assessment is focused on the core behaviour of the pattern decision aid. This core behaviour is the comparison and analysis workload associated to the post order traversal of the force hierarchy, as this has been the focus of the design effort. Although, the design may not consider every nuance of the end software, it is this core workload that is expected to dominate the asymptotic time complexity of the complete design. The assessment begins by considering the expected behaviour entailed by traversing the force hierarchy, and then specifically the workload expected at a visit to each type of force in the hierarchy. With these assessments an overall expected number of operations for pattern decision analysis is presented accompanied by appropriate comments.

Cost to Traverse the Force Hierarchy

Overall there will be n candidates in the decision problem and f force nodes in the force hierarchy. The high level view of the pattern decision aid algorithmically entails a post order traversal of the force hierarchy. Equation 19 contains the associated asymptotic assessment of traversing such a hierarchy, which, as stressed previously, will be used to represent the complexity of the pattern decision aid *pda*. In essence it is the summation of costs to visit, v_k , of a force node k in the hierarchy.

Equation 19 - Cost to Traverse the Force Hierarchy

$$pda = \sum_{k=1}^f v_k \quad \text{see post order traversal [29]}$$

In general, each visit, v_k entails pairwise comparison of all candidates taking $(n^2 - n)$ such comparisons sequentially followed by pairwise analysis of all candidates taking the same number of assessments. Thus a visit to a force node will take $v_k = (n^2 - n)c_k + (n^2 - n)a_k$, regardless of force type, where c_k represents the time complexity of a pair comparison, and a_k represents the time of a pair analysis. Whereas, c_k , will differ dependent on the force type, a_k will apply to each force type. Thus, while the following sections consider the tangible, general, and intangible, time complexity of a pair comparison respectively, the analysis operation assessment is introduced first.

Within the proposed design the java 2 class TreeMap is used to store force related information. Accordingly get() and put() operations will take $\log(z)$ operations, with z being the number of elements in a particular treeMap as assessed at:

(<http://java.sun.com/products/jdk/1.2/docs/api/java/util/TreeMap.html>).

Given that there will be f such elements, it follows that accessing or updating the force information will cost $\log(f)$ operations. With each pair of analysis taking a constant number of associated operations the associated asymptotic assessment of $a_k = O(\log(f))$. To conclude this section and lead into the following assessment, the assessment of v_k is updated in Equation 20.

Equation 20 - Operations to Visit a Force Node

$$\begin{aligned} v_k &= (n^2 - n)c_k + (n^2 - n)O(\log(f)) \\ &= (n^2 - n)c_k + O(n^2 \log(f)) \end{aligned}$$

Cost to Compare a Candidate Pair within a Tangible Force Node

Currently the manner in which the pattern decision aid has been designed, the tangible force node can only be an external node in the force hierarchy. That is, the associated node will not have any other forces contributing to it. Considering the set of tangible forces in the force hierarchy, T , the tangible force node $t \in T$ is visited to compare candidate pairs against associated metrics. The number of operations included in a pair comparison, c_t , corresponds to the number of metrics m_t . Where the metrics of interest are those that are used to indicate the presence of a tangible force. Similar to the analysis operation, a_k , each comparison of metric values will entail a sequence of accesses and updates to a TreeMap. Thus a pair comparison during a visit to a tangible force node, c_t , will take $m_t O(\log(m_t))$ or $O(m_t \log(m_t))$ operations.

Cost to Compare a Candidate Pair within a General Force Node

While a pair comparison within a visit to a tangible force node entails comparing metric values, the corresponding comparison, c_g , during a general force visit entails comparing metric values as well as contributing force *number of outranks* values. Given, f_c , is the number of child forces that contribute to a general force, the corresponding assessment is $c_g = m_g O(\log(m_g)) + f_c O(\log(f))$. To further refine this assessment, conditions under which the most number of contributing forces will occur are examined. In any force hierarchy there will always be at least one general force, the root, and typically it is entitled total satisfaction. If the force hierarchy were flat this would imply that the force hierarchy would consist of one node entailing c_g operations per comparison of a pair of candidates and $f-1$ contributing forces so that $f_c = f-1$. With this in mind Equation 21 summarizes the cost to compare a pair of candidates at a general force node.

Equation 21 - Cost to Compare a Pair at a General Force Node

$$O(c_g) = O(m_g \log(m_g)) + O(f \log(f))$$

Cost to Compare a Candidate Pair within a Intangible Force Node

A comparison within an intangible force node, c_i , entails comparing whether the intangible is requested and whether it is supported. In the proposed design, an intangible is an external node in the force hierarchy. That is there will not be any contributing forces to an intangible force. The associated pair comparison again only entails basic accessing and updating operations of a TreeMap. However, in this case, whereas the other forces must refer to subordinate objects, an intangible will require a constant number of TreeMap operations. The resulting comparison at an intangible force node $c_i = O(\log(f))$ operations.

Overall Expected Number Of Operations For Pattern Decision Aid

With the possible mixture of force types within a force hierarchy, Equation 19 is revisited to reflect the diversity of visit complexities encountered during a traversal of the force hierarchy. Listing 1 represents the subsequent analysis, with pda number of operations required to complete the pattern decision aid. To account for the number of different forces types encountered within the force hierarchy, f_t , f_g , and f_i correspond respectively to the number of tangible, general, and intangible forces. At the end of Listing 1 there remain some dimensions that can further impact the assessed pda complexity: the number of candidates, the number of metrics used to measure tangible forces, and the number of general forces.

$$\begin{aligned}
pda &\leq \sum_{k=1}^f v_k \\
&= \sum_{k=1}^f ((n^2 - n)c_k + O(n^2 \log(f))) \\
&= (n^2 - n) \sum_{k=1}^f c_k + \sum_{k=1}^f O(n^2 \log(f)) \\
&= (n^2 - n) \left(\sum_{i=1}^T c_i + \sum_{g=1}^{G_i} c_g + \sum_{i=1}^{I_i} c_i \right) + O\left(\sum_{k=1}^f n^2 \log(f) \right) \\
&= (n^2 - n) \left(\sum_{i=1}^T (O(m_i \log(m_i))) + \sum_{g=1}^{G_i} c_g + \sum_{i=1}^{I_i} (O(\log(f))) \right) + O(f \log(f) n^2) \\
&\leq (n^2 - n) \left(\sum_{i=1}^T (O(m_i \log(m_i))) + \sum_{g=1}^{G_i} (O(m_g \log(m_g))) + O(f \log(f)) + \sum_{i=1}^{I_i} (O(\log(f))) \right) + O(f \log(f) n^2) \\
&= (n^2 - n) (O(f_i m_i \log(m_i)) + O(f_g m_g \log(m_g)) + O(f_g f \log(f)) + O(f_i \log(f))) + O(f \log(f) n^2) \\
&= O(n^2 (\max(f_i m_i \log(m_i), f_g m_g \log(m_g), f_g f \log(f), f_i \log(f)) + O(f \log(f) n^2) \\
&\quad \text{Given that there is always one general force in a force hierarchy} \\
&\quad f > f_i \\
pda &= O(n^2 (\max(f_i m_i \log(m_i), f_g m_g \log(m_g), f_g f \log(f), f_i \log(f)))) \\
&= O(n^2 (\max(f_i m_i \log(m_i), f_g m_g \log(m_g), f_g f \log(f))) \\
&\quad \text{Given metrics indicating general forces are rare, as evidenced in the representative force hierarchy,} \\
&\quad \text{it is expected that } m_g \text{ will be significantly smaller than the number of forces, } f. \\
pda &\leq O(n^2 (\max(f_i m_i \log(m_i), f_g f \log(f)))
\end{aligned}$$

Listing 1 - Time Complexity of Pattern Decision Aid

The user controls the number of candidate patterns selected for the comparison. It is under their control because any set of candidates provided to the software must be compared pairwise. This requirement originates from ELECTRE II accommodating potential incomparability between candidates. Understandably in order to control this dimension the smaller the size of candidates under comparison, the more quickly the pattern decision aid will perform. One remedy is running a keyword search to reduce the pool of candidates vying for the pattern user's attention. Subsequently an iterative decision process can be initiated by running the pattern decision aid on a subset of candidates, with the pattern user reducing the pool of candidates before introducing a new candidate for a subsequent iteration. Executing this iterative process cannot be delegated

to the pattern decision aid. To do so is a violation of the spirit of decision analysis, whereby the user owns the responsibility to make a final decision.

Essentially there remain two dimensions that are within the control of the software: the number of tangible force metrics applied and the number of general forces. Figure 35 illustrates samples of different compositions of the force hierarchy. Sample A represents a shallow force hierarchy with many metrics measuring tangible forces. In this case the number of tangible metrics dominate the assessed complexity of the pattern decision analysis. In the second sample force hierarchy, the hierarchy is more descriptive in that the chain of forces and their influence on each other have been explicitly specified.

Figure 35 - Illustration of pda Complexity

Based on the depth of the hierarchy, the general forces and in turn the overall number of forces, dominate the assessment of the pattern decision aid complexity.

Both of these dimensions can be controlled external to the pattern decision aid, during the construction of the force hierarchy. By conducting interdependence analysis, a statistical correlation technique, a subset of indicators can be identified [39]. Hence, the number of metrics used to indicate the presence of a tangible force can be reduced. Further investigation of this reduction technique is left as future work and is considered outside the scope of this thesis.

Compressing the chain of influences can control the depth of the force hierarchy. That is, by examining the impacts from a tangible force to its parent general force, taking the product of impacts, can be used to reduce the number of general forces. This is particularly relevant when a general force only has one tangible force contributing to it. In this case, the general force does not add value to the force hierarchy within the context of the pattern decision aid. It should be noted that the force hierarchy associated with the proposed pattern decision aid, has been left uncompressed in order to expose a consistent representative view of the hierarchy. Again further treatment of this issue is left as future work during the implementation of the pattern decision aid.

Overall the time complexity of the pattern decision aid's core behaviour is reasonable. With a view to further improve the proposed design's performance, in terms of response time, suggestions have been made. Although it is evident there remains work to be completed in order to implement a distributed pattern decision aid, at this juncture the proposed design is sufficient to meet the objectives of this thesis.

PART 5 THESIS CONCLUSION

CHAPTER 11 CONCLUSION

11.1 Review of Objectives and Contributions

To draw the thesis to a close, the material contained herein is summarized followed by a detailed breakdown concerning how the thesis objectives and contributions have been realized. The thesis concludes with recommendations concerning the future directions of work subsequent to this thesis.

The articulation of pattern decision analysis, proposed in Chapter 6, facilitates the comparison of candidate object-oriented software patterns with the intent to establish those patterns more suitable for use in a design. By applying this approach in three independent tests, reasonable results confirm the suitability of the proposed approach to the pattern decision problem. Acknowledging that issues exist, pertaining to the composition of a force hierarchy around which a pattern decision must be formed, this thesis presents a legitimate starting point upon which further examination of comparing object-oriented software patterns can be based. With the suitability of the approach confirmed and by accommodating the evolution of the force hierarchy and object-oriented metrics, a design for a pattern decision aid has been formed.

11.1.1 Confirming Thesis Objectives

In order to confirm the objectives, the manner in which they were originally presented in the introduction is used to structure related comments.

1. Decision Analysis. Objectives relating to comparing patterns have been achieved as follows:

- a. Understanding Comparison Techniques. Chapter 2 is introduced by a review of two competing approaches that have the potential to be applied to the pattern decision problem. This review serves to demonstrate an understanding of the techniques and concludes by identifying the more suitable decision analysis approach for the pattern decision problem. Based on this selection, Chapter 6 proposes an adapted method to deal with pattern specific criteria and metrics.
 - b. Develop a Pattern Decision Analysis Approach. With pattern decision analysis articulated in Chapter 6 and confirmed by the discussion of test results in Chapter 9, a suitable approach has been proposed for comparing candidate object-oriented software patterns.
2. Patterns. Associated objectives, specifically related to patterns, that have been met by this thesis, include:
- a. Understanding Object-Oriented Software Patterns. Chapter 3 introduces the pattern from the perspective of the pattern decision problem followed by Chapters 4 and 5, which deal exclusively with distinguishable features that make patterns comparable.
 - b. Identify Pattern Samples. Given the pattern decision problem can take three different forms, three sets of patterns that display the characteristics of each were identified in Chapter 2. Namely: for pattern variations the State pattern and variations is suitable; for pattern alternatives four Visitor pattern alternatives were identified; and for the mixed pattern decision problem an arbitrary problem was constructed based on the Proxy and related patterns.

3. Testing. Chapter 7 outlines the manner in which the tests of the proposed decision analysis were prepared with the results subsequently presented in Chapter 9. Relevant descriptions of the pattern candidates implicated in each test accompany this thesis in Appendices 2, 3 and 4.
4. Decision Aid Tool. Chapter 8 outlines the approach to designing a distributed tool that supports the confirmed pattern decision analysis. Subsequently Chapter 10 presents a high level view of the design accompanied by an assessment of the proposed design. Details concerning the associated persistence strategy and detailed pattern decision aid design are presented in Appendix 5 and 6.

11.1.2 Confirming Thesis Contribution

In the introduction to this thesis a number of contributions were claimed. This confirmation is presented similar to the manner in which contributions were originally stated.

1. Pattern Decision Analysis. Contributions associated to patterns, in particular in the context of making a selection decision include:
 - a. Distinguishing Types of Forces. The discussion concerning tangible and intangible forces in Chapter 4 presents a distinction between forces while sections in Chapter 4 and 5 are dedicated to respectively highlighting the influences that another force or metric may have on a particular force.
 - b. Requirements for Pattern Writers. The Chapter 9 discussion, related to preparing the tests, highlights a number of requirements that pattern writers should consider while preparing their patterns in order to facilitate the selection of their pattern. These requirements focus on the structure models included to illustrate the pattern.

In particular these structures should be generic, complete in terms of the participants, represent the minimum solution, and be consistent when associated to a pattern language. Furthermore, the importance of modeling the expected behaviour of the pattern is also noted.

- c. Tailoring Existing Decision Analysis Approach. The proposed pattern decision analysis articulated in Chapter 6 adapts an existing decision analysis method, originating from operations research, to the pattern decision problem.

2. Pattern Decision Aid. Contributions that transition the proposed pattern decision analysis approach to a distributed pattern decision aid include:

- a. Persistence Strategy. A persistence strategy has been expressed in Chapter 10 with a XML document type definition accompanied by samples of the intended .xml files presented in Appendix 5.
- b. Pattern Decision Aid. A design for a pattern decision aid is introduced in Chapter 10, based on the confirmed suitability of pattern decision analysis. This design is suitable for a distributed environment, and sufficiently open to accommodate different perspectives concerning the composition of a pattern decision force hierarchy.
- c. Assessing Time Complexity. Chapter 10 finishes with a time complexity analysis of the proposed pattern decision aid design. Based on this assessment suggestions have been presented to resolve issues that may impact the response time of the software.

At this juncture the thesis goal, objectives and contributions have been successfully accounted. Although the criteria of success have been met, a number of issues, challenges and potential enhancements have been identified that should be treated in work subsequent to this thesis. This document finishes with a discussion of the future directions this work may take.

11.2 Future Directions

During the activities undertaken in conjunction with this thesis, a number of issues were identified posing challenges that could not be dealt with reasonably within the scope of the associated work. It is considered that the resolution of these issues may prove interesting for further investigation. The future directions are presented consistent with the major objectives of the thesis:

1. Patterns and Decision Making. Further issues that may be investigated include:
 - a. Intangible Forces. The intangible forces collected within this thesis largely originate from the Gang Of Four patterns, and those patterns implicated in the three tests. This does not preclude the existence of other intangible forces that contribute to a pattern's utility. It is cautioned that with every pattern writer, the semantics used to describe a particular intangible force may be different than another, and yet, in essence they are referring to the same force.
 - b. Tangible Forces. It has been highlighted in the test results that the composition of the force hierarchy requires further examination. A critical review of internal design attributes and their contribution to external product qualities or non-functional requirements will form the impetus to

evolve the force hierarchy. Furthermore, work related to evolving pattern decision analysis should be shared with continued initiatives to link non-functional requirements and object-oriented patterns[30].

- c. Pattern Metrics. Based on the evolution of object-oriented metrics towards the accounting of precise design properties, their application to patterns becomes unreasonable. The usefulness of patterns is heightened when they are context independent. Hence a number of the values required for a particular design metric will not be available within the pattern. It is recommended that investigating pattern-oriented metrics that appropriately contribute to pattern decision analysis is worthwhile. It is envisaged that pattern-oriented metrics are not new metrics but rather account for a subset of the properties collected by existing object-oriented design metrics. While empirical evidence confirming object-oriented metrics remains a challenge, with the number and diversity of patterns there is ample opportunity to pursue the empirical confirmation of established metrics in the context of pattern measurement.

2. Testing. Although, this thesis confirms pattern decision analysis with three tests, the approach will be further strengthened with the results of further testing. The range of patterns available is extensive, with an associated range of possible pattern decision problems. As further testing effort is applied to pattern decision analysis, trends should be concurrently monitored. Trends may exist across different pattern decision problems. As well trends should be examined within each pattern decision problem as different pattern user preference profiles are

considered. An approach has been applied to the ELECTRE II method whereby conflicting decision groups' preferences can be examined and conclusions subsequently drawn from the decision problem [38]. This approach to model different pattern user preferences can be investigated in light of the proposed pattern decision analysis.

3. Pattern Decision Aid. There remains development and testing effort required to implement the pattern decision aid. In conjunction with pursuing this effort it is recommended that the pattern force shadow be integrated with the existing pattern XML specification, as well as integrating the pattern decision aid with the existing pattern repository.
4. Enhancing Web-based Object-Oriented Software Design using Patterns. The design accompanying this thesis has focused on automating pattern decision analysis, which is driven by the measurement of pattern structures. However, the pattern decision aid is one capability under the umbrella of web-based object-oriented software design tools that can be based on object-oriented software patterns. With the emergence of the UML XML Metadata Interchange (XMI) specification [42], it will be possible for pattern structures and expected behaviour to have a related XMI representation of the UML class and sequence diagrams. With these representations available, it will be a valuable initiative to develop a calculator that measures patterns with results populating the pattern force shadow. By delegating the responsibility of pattern measurement to software, the pattern writer, who would be otherwise responsible, will be allowed to concentrate on preparing a comprehensive pattern.

CHAPTER 12 BIBLIOGRAPHY

- [1] Alexander, C., *The Timeless Way of Building*, Oxford University Press, 1979
- [2] Alexander, C., Ishikawa, S., Silverstein, M., *A Pattern Language*, Oxford University Press, NY, 1977.
- [3] Barbacci, M., Klein, M.H., Longstaff, T.A., Weinstock, C.B., *Quality Attributes*, Technical Report CMU/SEI-95-TR-021, Dec 1995 Retrieved May 29, 2000 from the World Wide Web: <http://www.sei.cmu.edu/publications/documents/95.reports/95.tr.021.html>
- [4] Basili, V.R., Briand, L.C., Melo, W.L., *A Validation of Object-Oriented Design Metrics as Quality Indicators*, IEEE Transactions on Software Engineering, Vol 22, No. 10, 1996, 751-761.
- [5] Basili, V.R., Rombach, H.D., 'The TAME project: Towards improvement-oriented software environments', IEEE Transactions on Software Engineering., Vol 14, No. 6, 1988, 758-773.
- [6] Bäumer, D., Riehle, D., Siberski, W., Wulf, M., *Role Object, Pattern Languages of Program Design 4*, Addison Wesley Longman, Inc., Reading, MA., 2000. 15-31.
- [7] Binkley, A.B., Schach, S.R., *A comparison of sixteen quality metrics for object-oriented design*, Information Processing Letters, Vol 58, 1996, 271-275.
- [8] Booch, G., *Object-Oriented Analysis and Design with Applications*, The Benjamin/Cummings Publishing Company, Inc, 1994.
- [9] F. Bordeleau. *A Systematic and Traceable Progression from Scenario Models to Communicating Hierarchical State Machines*. Ph.D. Thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1999. (available at <http://www.scs.carleton.ca/~francis>).
- [10] Briand, L., Arisholm, E., Counsell, S., Houdek, F., Thévenod-Fosse, P., *Empirical Studies of Object-Oriented Artifacts, Methods, and Proceses: State of the Art and Future Directions*, ISERN Technical Report 99-12. Retrieved May 15, 2000 from the World Wide Web: http://www.iese.fhg.de/network/ISERN/pub/isern_biblio_tech.html
- [11] Briand, L.C., Daly, J.W., Wüst, J., *A Unified Framework for Cohesion Measurement in Object-Oriented Systems*, ISERN Technical Report, 97-05. Retrieved May 15, 2000 from the World Wide Web: http://www.iese.fhg.de/network/ISERN/pub/isern_biblio_tech.html
- [12] Briand, L.C., Wurst, J., Daly, J.W., Porter, D.V., *Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems*. ISERN Technical Report 98 - 07. Retrieved May 15, 2000 from the World Wide Web: http://www.iese.fhg.de/network/ISERN/pub/isern_biblio_tech.html
- [13] Buschmann, F., Meunier, R., Ronhert, H., Sommerlad, P., Stal, M., *Pattern – Oriented Software Architecture, A System of Patterns*, John Wiley & Sons Ltd., 1996.
- [14] Cartwright, M., *An Empirical View of Inheritance*, Information and Software Technology, Vol 40, No 14, 1998, 795-799.
- [15] Chidamber, Darcy, Kemerer, *Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis*. IEEE Transactions on Software Engineering, Vol. 24, No 8, 1998 Aug, pp 629 - 639.
- [16] Chidamber, Kemerer, *A Metrics Suite For Object-oriented Design*, IEEE Transactions on Software Engineering, Vol 20 No 6, 1994, Jun, pp. 476 - 493.
- [17] Chidamber, Kemerer, *Author's Reply to Comments on 'A Metrics Suite for Object-oriented Design'*, IEEE Transactions on Software Engineering, Vol. 21, No. 3, Mar 1995, p. 265.
- [18] Cormen, T.H., Leirerson., C.E., Rivest, R.L., *Introduction to Algorithms*, MIT Press, 1990
- [19] Corriveau, J.P., Nel, L.D., Radonjic, V.D., *Introduction to Object-Oriented Software Engineering*, vers 3.0, 1999.
- [20] Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M., *The Effect of Inheritance on the Maintainability of Object-Oriented Software: An Empirical Study.*, Proceedings IEEE International Conference on Software Maintenance, 1995, pp. 20 – 29.
- [21] Dromey, R.G., *Cornering the Chimera*, IEEE Software, Jan 1996, pp 33 – 43.
- [22] Dyson, P., Anderson, B., *State Patterns*, Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998, pp. 125-145.
- [23] Etzkorn, L., Bansiya, J., Davis, C., *Design and Code Complexity Metrics for OO Classes*, Journal of Object-Oriented Programming, March/April, 1999, pp. 35 – 40.
- [24] Fenton, N.E., *Software Metrics a Rigorous Approach*, Chapman and Hall, 1991.

- [25] Ferguson, D, XML Declaration for Patterns: An XML Specification for Documentation of Patterns, Carleton University, Retrieved 8 Jan., 2000 from the World Wide Web: <http://muffin.nexus.carleton.ca/~darrell/papers/PatternReport.pdf>
- [26] Gamma E., Helm R., Johnson, R., and Vlissides, J., Design Patterns CD Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Inc., Reading, MA, 1998.
- [27] Gillibrand, D., Kecheng, L., Quality Metrics for Object-Oriented Design, Journal of Object-Oriented Programming, Jan 1998, 56 – 59.
- [28] Golub, D.L., Decision Analysis An Integrated Approach, John Wiley & Sons, Inc, 1997.
- [29] Goodrich, M.T., Tamassia, R., Data Structures and Algorithms in Java, John Wiley & Sons, Inc., 1998.
- [30] Gross, D., Yu, E., From Non-Functional Requirements to Design through Patterns, Proceedings of the 6th International Workshop on Requirements Engineering: Foundations for Software Quality (June 5-6, 2000), Stockholm, Sweden. Submitted version., Retrieved Oct 12, 2000 from the World Wide Web: <http://www.cs.toronto.edu/~eric/#Publications>.
- [31] Johnson, R., Woolf, B., Type Object, Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998, 47-65.
- [32] Li, W., Another metric suite for object-oriented programming., The Journal of Systems and Software, Vol. 44, 1998, 155-162.
- [33] Li, W., Henry, S., Kafura, D., Schulman, R., Measuring Object-Oriented Design, Journal of Object-Oriented Programming, July-Aug., 1995, 48 – 55.
- [34] Lin, C-Y., Teng, J-Y., Multicriteria Decision Making for Selecting Freeway Interchange Locations in Taiwan, Multiple Criteria Decision Making, Proceedings of the Ninth International Conference: Theory and Applications in Business, Industry, and Government., Springer-Verlag New York, Inc., 1992, 227-238.
- [35] Lorenz, M., Kidd, J., Object-Oriented Software Metrics, Prentice-Hall, Inc, 1994.
- [36] Martin., R.C., Acyclic Visitor, Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998. 93 – 103.
- [37] Meszaros, G., Doble, J., A Pattern Language for Pattern Writing, , Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998. pp. 529 – 574.
- [38] Nijkamp, P., Stochastic Quantitative and Qualitative Multicriteria Analysis for Environmental Design. Papers of the Regional Science Association, Regional Science Association, Vol. 38, 1977, 175-199.
- [39] Nijkamp, P., van Delft, A., Multi-Criteria Analysis and Regional Decision Making., H.E. Stenfert Kroese B.V., Leiden, 1977.
- [40] Noble, J., Basic Relationship Patterns, Pattern Languages of Program Design 4, Addison Wesley Longman, Inc., Reading, MA., 2000, 74-89.
- [41] Nordberg III, M.E., Default and Extrinsic Visitor, Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998, 105 – 123.
- [42] OMG Unified Modeling Language Specification, Version 1.3, June 1999.
- [43] Pressman, R.S., Software Engineering A Practitioner's Approach, McGraw-Hill, Inc., 1992.
- [44] Roy, B., Partial Preference Analysis and Decision-Aid: The Fuzzy Outranking Relation Concept, Conflicting Objectives in Decisions, Wiley, 1977.
- [45] Schmidt, D.C., Using Design Patterns to Develop Reusable Object-Oriented Communication Software, Communications of the ACM, October 1995, Vol 38, No 19, 65-74.
- [46] Sherif, J.F., Metrics for Object-Oriented Software Projects, The Journal of Systems and Software, Elsevier, Vol 44, 1998, 147-154.
- [47] Silva, A.R., Rosa, F.A, Gonçalves, T., Antunes, M., Distributed Proxy: A Design Pattern for Distributed Object Communication, Conference of Pattern languages of Programs, Allerton Park, Il, 1997.
- [48] Simpson, L., Do Decision Makers Know What They Prefer?: MAVT and ELECTRE II, Journal of the Operational Research Society, Vol. 47, No. 7, 1996, 919-929.
- [49] Smith, C.U., Performance Engineering of Software Systems, Addison-Wesley Publishing Company, Inc., 1990.
- [50] Srinivasan, S., Design Patterns in Object-Oriented Frameworks, Computer, Vol. 32, No. 2, Feb. 1999, 24-32.
- [51] Subramanian, S., Tsai, W-T., Backup Pattern: Designing Redundancy in Object-Oriented Software, Pattern Languages of Program Design 2, Addison Wesley Longman, Inc., Reading MA., 1996. 207 – 225.

- [52] Tervonen, I., Support for Quality-Based Design and Inspection, IEEE Software, Jan 1996, 44 - 54.
- [53] The Patterns Handbook: Techniques, Strategies, and Applications, Cambridge University Press, 1998.
- [54] Vincke, P., Multicriteria Decision-aid, John Wiley & Sons Ltd, Chichester, UK, 1992,
- [55] Vincke, P., Robust Solutions and Methods in Decision-Aid, Journal of Multi-Criteria Decision Analysis, Vol. 8, 1999, 181-187.
- [56] Whitmire, S.A., Object-oriented Design Measurement, John Wiley & Sons, Inc, 1997.
- [57] Woolf, B., Null Object, Pattern Languages of Program Design 3, Addison Wesley Longman, Inc., Reading MA., 1998, 5 – 18.
- [58] Yacoub, S.M., Ammar, H.H., Finite State Machine Patterns, Pattern Languages of Program Design 4, Addison Wesley Longman, Inc., Reading, MA., 2000. 413-440.
- [59] Zeleny, M., Multiple Criteria Decision Making, McGraw-Hill, Inc., 1982.

APPENDIX 1 UML NOTATION

UML Use Case Diagrams

UML Static Structure Diagrams

UML Sequence Diagram

UML Statechart

Figure 36 - Overview of UML Notation

Figure 36 provides an overview of the UML 1.3 use case, static structure (class or object), sequence and state diagram notations.

APPENDIX 2 STATE PATTERNS TEST

A2.1 State Patterns Overview

With a view to expedite the presentation of information related to each of the patterns implicated in this test, the following table presents an abridged version of the pattern description presenting the pattern title, a brief statement of the problem accompanied by an outline of the resolution of the problem. For the purposes of comparison a discussion of forces at play in the pattern are important, hence subsequent columns outline the force at play in the problem and then a remark concerning the how the solution resolves the identified force. Subsequent sections in this appendix provide information related to each of the patterns, but details are limited to the authors, illustrations of the solution structure, behaviour, and measured tangible and intangible forces. The appendix concludes with the results of the two set of tests conducted in conjunction with the state pattern test. The test result starts with the values input into pattern decision analysis followed by the results of the seven test serials associated with each test. It should be noted that this outline is applicable to Appendices 3, and 4.

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
State	How to enable an object to behave according to its current internal state?	Introduce a separate object representing the state, which encapsulates the diverse state dependent behaviour.	Response Time	Not an issue that would otherwise preclude use of this pattern. If it is then using a state variable is a more suitable solution.	The state object introduces extra message indirection, which has a negative impact on response time.
State Member. Isolated State (1)	<p>"How do you decide whether a data member belongs in the owning class or in the State Object class?" [22]</p> <p>"How do you decide whether a data member belongs in the owning class or in the State Object class?" [22]</p>	<p>"If a data member is only required for a single state, then place it in the corresponding State Object class." [22]</p> <p>"If the data member is required for some, but not all, states, then it should be placed in a common superclass." [22]</p>	Extensibility	The ability to add new states with limited effort and low impact on existing states is a desired property.	Keeping state specific behaviour separated in state objects, adding new state won't impact existing ones.
			Maintainability	For the purposes of maintaining an owning class, state related behaviour must be easily located.	As state related behaviour is encapsulated within separate state objects, it can be located quickly.
			Coupling	Not an issue that would otherwise preclude use of this pattern.	"State Object is very closely coupled to its owning object." [22]
State Member. Among Select States (2)	<p>"How do you decide whether a data member belongs in the owning class or in the State Object class?" [22]</p>	<p>"If the data member is required for some, but not all, states, then it should be placed in a common superclass." [22]</p>	Data Sharing	The data is used only when the owning object is a specific state.	The data use is isolated to one State Object.
			Coupling	Not an issue that would otherwise preclude use of this pattern.	"State Object is very closely coupled to its owning object." [22]
			Data Sharing	Data must be shared between specific states.	The data can be shared between specific states.
State Member. Common (3)	<p>"How do you decide whether a data member belongs in the owning class or in the State Object class?" [22]</p>	<p>"If the data member is required for some, but not all, states, then it should be placed in a common superclass." [22]</p>	Overhead Complexity	Not an issue that would otherwise preclude use of this pattern.	Because multiple State Objects have access to the same attribute, the use of the data (State Member) must be "synchronized as transitions between states occur" [22].
			Coupling	Not an issue that would otherwise preclude use of this pattern.	"State Object is very closely coupled to its owning object." [22]
			Data Sharing	Data is state independent.	Access to the data is equally shared between the owning object and the state objects
Pure State	<p>"You have a lot of State Objects. Is it possible to cut down on the number required?" [22]</p>	<p>"When a State Object has no State Members, it can be said to represent Pure State – nothing but state-specific behaviour. A single Pure State object can be shared among any number of owning objects, drastically reducing the number of objects required." [22]</p>	Overhead Complexity	Not an issue that would otherwise preclude use of this pattern.	Because multiple State Objects have access to the same attribute, the use of the data (State Member) must be "synchronized as transitions between states occur" [22].
			Memory Usage	Memory is considered a valuable resource. Hence, controlling its usage is considered a worthwhile endeavour.	Any number of owning objects can share a single State Object reducing the number of objects required in memory at run time [22].

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Exposed State	"How do you prevent the owning class from having an excessive number of state-specific, state-dependent methods?" [22]	"Expose the State Object by defining a method in the owning class that returns a reference to it. Make state-specific enquiries directly to the State Object." [22]	Data Sharing	A large number of State Members have been used.	By accessing State Object directly, direct access to relevant data members is promoted.
			Fault Proneness	Not an issue that would otherwise preclude use of this pattern.	"The owning object must be checked to ensure it is in the required state to make state-specific inquiries.[22]
			Specialization	A problem exists when a lot of state-dependent messages delegated to the State Object are also state specific[22]. Consequently there are a large number of methods in the State Object classes[22].	The state is exposed so that it can handle the specialized requests while not adding to the owning object.
State Driven Transitions	"How do you get the State Object to change when the owning object's state changes?" [22]	"Have the State Object initiate the transition from itself (the current state) to the new State Object. This ensures transitions are atomic and removes state-dependent code." [22]	Dynamic Transitions	Seeking the ability for the software to internally transition between states.	The State object is enabled to initiate the transition between itself to the new State object.
Owner Driven Transitions	"How do you reuse State Object classes among owning classes with different state-transition profiles?" [22]	"If State Object classes are to be used by more than one owning class, and those owning classes have different finite state machines, have the owning class initiate the transition between states." [22]	Dynamic Transitions	Seeking the ability for the software to internally transition between states.	The owning class initiates the transition between states.
			Reusability	Owning classes will reuse State Object classes and finite state machines are not the same between owning class.	With the owning class responsible for initiating transitions, owning classes can still reuse the same State classes whereas using State Driven Transitions wouldn't support this.
Default State	"When creating a new owning object, how do I ensure that it has the correct initial State Object?" [22]	"Use a method, called by the #initialize method, that returns the default State Object. Redefine this method in a subclass if a different default state is required." [22]	Structural Flexibility	Owning classes have different default states and/or the default State Object may need to be changed.	Specifying default states in owning classes, with the default state returned when the initialize is invoked for an owning class. (applicable to Smalltalk not C++)

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Strategy	Many related classes differ only in their behaviour or the manner in which they are implicated in different algorithms.	"Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it." [26]	Maintainability	For the purposes of maintenance, algorithm specific behaviour must be easily located.	Keeping algorithm specific operations separate from the context and other algorithms maintenance can be focused more readily than if internal to context classes.
			Memory Usage	Not an issue that would otherwise preclude use of this pattern.	"Strategies increase the number of objects in an application."
			Response Time	Not an issue that would otherwise preclude use of this pattern.	Rather than direct access to specific algorithm behaviour, message indirection increases because of "increased communication overhead between Context and Strategy" [26] .
			Reusability	A number of algorithms are applicable to similar classes. As well certain algorithms implicate certain operations that are the same.	"Hierarchies of Strategy classes define a family of algorithms or behaviors for contexts to reuse. Inheritance can help factor out common functionality of the algorithms... Inheritance offers another way to support a variety of algorithms or behaviors" [26]

A2.2 Pattern Title: State Object

Authors

Original: E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Adapted By: P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.State / StateObject

Solution Behaviour

UML:Sequence.State ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.00	2.00	0.82	-
WMC	0.00	1.00	0.75	1.00	0.50	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.50	1.00	0.58	-
NOC	0.00	0.00	0.50	2.00	1.00	-
CBO	0.00	0.00	0.25	1.00	0.50	-
MPC	0.00	0.00	0.25	1.00	0.50	-
DAC	0.00	0.00	0.25	1.00	0.50	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	0
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.3 Pattern Title: State Member (Isolated State)

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.StateMember.isolatedState

Solution Behaviour

UML:Sequence.StateMember.Isolated ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.50	1.33	2.00	0.82	-
WMC	0.00	1.00	0.83	1.00	0.41	-
NCC	-	-	-	-	-	5.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.00	0.33	1.00	0.52	-
NOC	0.00	0.00	0.33	2.00	0.82	-
CBO	0.00	0.50	0.50	1.00	0.55	-
MPC	0.00	0.50	0.50	1.00	0.55	-
DAC	0.00	0.50	0.50	1.00	0.55	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	1
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.4 Pattern Title: State Member (Among Select States)

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.StateMember.amongSelectStates

Solution Behaviour

UML:Sequence StateMember(Among Select States)ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	2.00	1.50	2.00	0.76	-
WMC	0.00	1.00	0.88	1.00	0.35	-
NCC	-	-	-	-	-	7.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.75	2.00	0.89	-
NOC	0.00	0.00	0.50	2.00	0.93	-
CBO	0.00	1.00	0.88	2.00	0.83	-
MPC	0.00	1.00	0.63	1.00	0.52	-
DAC	0.00	0.00	0.38	1.00	0.52	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	1
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.5 Pattern Title: State Member (Common)

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.StateMember.common

Solution Behaviour

UML:Sequence.StateMember.Common ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	2.40	7.00	2.79	-
WMC	0.00	1.00	1.60	4.00	1.52	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.00	0.40	1.00	0.55	-
NOC	0.00	0.00	0.40	2.00	0.89	-
CBO	0.00	0.00	0.60	2.00	0.89	-
MPC	0.00	0.00	0.80	3.00	1.30	-
DAC	0.00	0.00	0.40	2.00	0.89	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Present
Data Sharing	1
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.6 Pattern Title: Pure State

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

Solution Behaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.00	2.00	0.82	-
WMC	0.00	1.00	0.75	1.00	0.50	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.50	1.00	0.58	-
NOC	0.00	0.00	0.50	2.00	1.00	-
CBO	0.00	0.00	0.25	1.00	0.50	-
MPC	0.00	0.00	0.25	1.00	0.50	-
DAC	0.00	0.00	0.25	1.00	0.50	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	0
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.7 Pattern Title: Exposed State

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.ExposedState

Solution Behaviour

UML:Sequence.ExposedState ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	2.00	2.33	5.00	1.97	-
WMC	0.00	1.00	1.17	2.00	0.75	-
NCC	-	-	-	-	-	5.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.00	0.33	1.00	0.52	-
NOC	0.00	0.00	0.33	2.00	0.82	-
CBO	0.00	0.50	0.83	3.00	1.17	-
MPC	0.00	0.50	0.83	3.00	1.17	-
DAC	0.00	0.50	0.50	1.00	0.55	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	0
Dynamic Transitions	0
Extensibility	1
Specialization	1
Structural Flexibility	0

A2.8 Pattern Title: State Driven Transitions

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.StateDrivenTransitions

Solution Behaviour

UML:Sequence.StateDrivenTransitions ExpectedBehaviour

Forces in Solution**Tangible**

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	5.00	3.75	5.00	2.50	-
WMC	0.00	2.00	1.75	3.00	1.26	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.50	1.00	0.58	-
NOC	0.00	0.00	0.50	2.00	1.00	-
CBO	0.00	1.00	0.75	1.00	0.50	-
MPC	0.00	2.00	1.50	2.00	1.00	-
DAC	0.00	0.00	0.25	1.00	0.50	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Present
Data Sharing	0
Dynamic Transitions	1
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.9 Pattern Title: Owner Driven Transitions

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.Owner Driven Transitions

Solution Behaviour

UML:Sequence.OwnerDrivenTransitions ExpectedBehaviour

Forces in Solution**Tangible**

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	3.00	3.67	8.00	3.61	-
WMC	0.00	2.00	1.67	3.00	1.37	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	2.00
DIT	0.00	1.00	0.67	1.00	0.52	-
NOC	0.00	0.00	0.67	2.00	1.03	-
CBO	0.00	0.00	0.33	1.00	0.52	-
MPC	0.00	0.00	1.33	4.00	2.07	-
DAC	0.00	0.00	0.33	1.00	0.52	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Data Sharing	0
Dynamic Transitions	1
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.10 Pattern Title: Default State

Authors

P. Dyson, and B. Anderson [22]

Solution Structure

UML:Class.Default State

Solution Behaviour

UML:Sequence DefaultState ExpectedBehaviour

Forces in Solution**Tangible**

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	2.00	1.67	3.00	1.37	-
WMC	0.00	1.00	1.33	3.00	1.37	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	2.00
DIT	0.00	1.00	0.67	1.00	0.52	-
NOC	0.00	0.00	0.67	2.00	1.03	-
CBO	0.00	0.00	0.33	1.00	0.52	-
MPC	0.00	0.00	0.67	2.00	1.03	-
DAC	0.00	0.00	0.33	1.00	0.52	-

Intangible

Intangible Force	Present
Data Sharing	0
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	1

A2.11 Pattern Title: Strategy

Authors

E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Solution Structure

UML:Class.Strategy

Solution Behaviour

UML:Sequence.Strategy ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.00	2.00	0.82	-
WMC	0.00	1.00	0.75	1.00	0.50	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.50	1.00	0.58	-
NOC	0.00	0.00	0.50	2.00	1.00	-
CBO	0.00	0.00	0.25	1.00	0.50	-
MPC	0.00	0.00	0.25	1.00	0.50	-
DAC	0.00	0.00	0.25	1.00	0.50	-

Intangible

Intangible Force	Supported
Data Sharing	0
Dynamic Transitions	0
Extensibility	1
Specialization	0
Structural Flexibility	0

A2.12 Testing Results: State Variations

Decision Context

Preferences		Test Serial Preference Weights						
Force		1	2	3	4	5	6	7
Performance		1	0	0	0	5	7	3
Maintenance Effort		0	1	0	0	5	3	7
Reliability		0	0	1	0	5	7	3
Utility		0	0	0	1	5	3	7

- 1 – Performance
- 2 – Maintainability
- 3 – Reliability
- 4 – Utility
- 5 – Balanced
- 6 – Operations
- 7 – Maintenance

Thresholds	
Name	Value
$\hat{c}_1$	0.70
$\hat{c}_2$	0.50
d	0.10

Candidates

Hierarchy of Forces																		
Candidate	Performance(+)			Maintainability(+)				Reliability(+)					Utility(+)					
	RFC(+)	Response Time (-)	Size(-)		Reusability(+)		Fault Proneness(-)					Data Sharing	Dynamic Transitions	Extensibility	Specialization	Structural Flexibility		
			WMC (+)	NCC (+)	NAC(+)	DIT (+)	NOC (+)	DIT (+)	NOC (+)	Coupling(+)								
										Inheritance(+)	Inheritance(+)						Coupling(+)	
																		Inheritance(+)
1.00	0.75	3.00	1.00	0.50	0.50	0.50	0.50	0.25	0.25	0.25	1.00	0.00	0.00	1.00	0.00	0.00		
State Object	1.33	0.83	5.00	1.00	0.33	0.33	0.33	0.33	0.50	0.33	0.33	0.50	0.50	0.50	1.33	1.00	0.00	0.00
State Member (isolated) [1]																		
State Member (amongSelectStates) [2]	1.50	0.88	7.00	1.00	0.75	0.50	0.75	0.50	0.88	0.63	0.38	1.50	1.00	0.00	1.00	0.00	0.00	0.00
State Member (common) [3]	2.40	1.60	4.00	1.00	0.40	0.40	0.40	0.40	0.60	0.80	0.40	2.40	1.00	0.00	1.00	0.00	0.00	0.00
Pure State	1.00	0.75	3.00	1.00	0.50	0.50	0.50	0.50	0.25	0.25	0.25	1.00	0.00	0.00	1.00	0.00	0.00	0.00
Exposed State	2.33	1.17	5.00	1.00	0.33	0.33	0.33	0.33	0.83	0.83	0.50	2.33	0.00	0.00	1.00	1.00	0.00	0.00
State Driven Transitions	3.75	1.75	3.00	1.00	0.50	0.50	0.50	0.50	0.75	1.50	0.25	3.75	0.00	1.00	1.00	0.00	0.00	0.00
Owner Driven Transition	3.67	1.67	4.00	2.00	0.67	0.67	0.67	0.67	0.33	1.33	0.33	3.67	0.00	1.00	1.00	0.00	0.00	0.00
Default State	1.67	1.33	4.00	2.00	0.67	0.67	0.67	0.67	0.33	0.67	0.33	1.67	0.00	1.00	1.00	0.00	0.00	1.00

Decision Results

Serial 1 - Performance	
State Object	1.00
Pure State	1.00
State Member 1	0.86
State Member 2	0.71
Default State	0.57
Exposed State	0.43
State Member 3	0.29
Owner Transitions	0.14
State Transitions	0.00

Serial 2 - Maintainability	
Default State	1.00
State Object	0.75
Pure State	0.75
State Transitions	0.50
Owner Transitions	0.50
State Member 3	0.38
State Member 1	0.13
State Member 2	0.13
Exposed State	0.00

Serial 3 - Reliability	
State Member 1	1.00
State Object	0.75
Pure State	0.75
State Member 3	0.63
State Member 2	0.25
Exposed State	0.25
Default State	0.25
State Transitions	0.13
Owner Transitions	0.00

Serial 4 - Utility	
State Member 1	1.00
State Member 2	1.00
State Member 3	1.00
Exposed State	1.00
State Transitions	1.00
Owner Transitions	1.00
Default State	1.00
State Object	0.00
Pure State	0.00

Serial 5 - Balanced	
State Object	1.00
State Member 1	1.00
Pure State	1.00
Default State	0.83
State Member 2	0.67
State Member 3	0.50
Exposed State	0.00
State Transitions	0.00
Owner Transitions	0.00

Serial 6 - Operations	
State Object	1.00
State Member 1	1.00
Pure State	1.00
Default State	0.83
State Member 2	0.50
State Member 3	0.50
Exposed State	0.33
State Transitions	0.00
Owner Transitions	0.00

Serial 7 - Maintenance	
Default State	1.00
State Member 1	0.88
State Member 3	0.63
State Member 2	0.38
State Object	0.13
Pure State	0.13
State Transitions	0.13
Owner Transitions	0.13
Exposed State	0.00

A2.13 Testing Results: State Variations with Strategy

Decision Context

Preferences		Test Serial Preference Weights						
Force		1	2	3	4	5	6	7
Performance		1	0	0	0	5	7	3
Maintenance Effort		0	1	0	0	5	3	7
Reliability		0	0	1	0	5	7	3
Utility		0	0	0	1	5	3	7

1 – Performance 5 – Balanced
2 – Maintainability 6 – Operations
3 – Reliability 7 – Maintenance
4 – Utility

Thresholds	
Name	Value
$\hat{c}_1$	0.70
$\hat{c}_2$	0.50
d	0.10

Candidates

Hierarchy of Forces																		
Performance(+)				Maintainability(+)				Reliability(+)				Utility(+)						
Response Time (-)				Size(-)		Reusability(+)		Fault Proneness(-)						Data Sharing	Dynamic Transitions	Extensibility	Specialization	Structural Flexibility
				Population(+)	WMC (+)	NCC (+)	NAC(+)	DIT (+)	NOC (+)	Coupling(+)								
										Inheritance(+)	Inheritance(+)	NOC (+)	MPC (+)					
RFC(+)																		
Candidate																		
State Object				1.00	0.75	3.00	1.00	0.50	0.50	0.50	0.50	0.25	0.25	0.25	1.00	0.00	1.00	0.00
State Member.isolated (1)				1.33	0.83	5.00	1.00	0.33	0.33	0.33	0.33	0.50	0.50	0.50	1.33	1.00	1.00	0.00
State Member amongSelectStates (2)				1.50	0.88	7.00	1.00	0.75	0.50	0.75	0.50	0.88	0.63	0.38	1.50	1.00	1.00	0.00
State Member.common (3)				2.40	1.60	4.00	1.00	0.40	0.40	0.40	0.40	0.60	0.80	0.40	2.40	1.00	1.00	0.00
Pure State				1.00	0.75	3.00	1.00	0.50	0.50	0.50	0.50	0.25	0.25	0.25	1.00	0.00	1.00	0.00
Exposed State				2.33	1.17	5.00	1.00	0.33	0.33	0.33	0.33	0.83	0.83	0.50	2.33	0.00	1.00	0.00
State Driven Transitions				3.75	1.75	3.00	1.00	0.50	0.50	0.50	0.50	0.75	1.50	0.25	3.75	0.00	1.00	0.00
Owner Driven Transition				3.67	1.67	4.00	2.00	0.67	0.67	0.67	0.67	0.33	1.33	0.33	3.67	0.00	1.00	0.00
Default State				1.67	1.33	4.00	2.00	0.67	0.67	0.67	0.67	0.33	0.67	0.33	1.67	0.00	1.00	1.00
Strategy				1.00	0.75	3.00	1.00	0.50	0.50	0.50	0.50	0.25	0.25	0.25	1.00	0.00	1.00	0.00

Decision Results

Serial 1 - Performance	
State Object	1.00
Pure State	1.00
Strategy	1.00
State Member 1	0.86
State Member 2	0.71
Default State	0.57
Exposed State	0.43
State Member 3	0.29
Owner Transitions	0.14
State Transitions	0.00

Serial 2 - Maintainability	
Default State	1.00
State Object	0.67
Pure State	0.67
Strategy	0.67
State Transitions	0.44
Owner Transitions	0.44
State Member 1	0.11
State Member 2	0.11
State Member 3	0.11
Exposed State	0.00

Serial 3 - Reliability	
State Member 1	1.00
State Object	0.67
Pure State	0.67
Strategy	0.67
State Member 3	0.56
Exposed State	0.33
Default State	0.33
State Member 2	0.11
State Transitions	0.11
Owner Transitions	0.00

Serial 4 - Utility	
State Member 1	1.00
State Member 2	1.00
State Member 3	1.00
Exposed State	1.00
State Transitions	1.00
Owner Transitions	1.00
Default State	1.00
State Object	0.00
Pure State	0.00
Strategy	0.00

Serial 5 - Balanced	
State Object	1.00
State Member 1	1.00
Pure State	1.00
Strategy	1.00
Default State	0.83
State Member 3	0.67
State Member 2	0.50
Exposed State	0.17
State Transitions	0.17
Owner Transitions	0.00

Serial 6 - Operations	
State Object	1.00
State Member 1	1.00
Pure State	1.00
Strategy	1.00
Default State	0.83
State Member 3	0.67
State Member 2	0.33
Exposed State	0.33
State Transitions	0.17
Owner Transitions	0.00

Serial 7 - Maintenance	
Default State	1.00
State Member 1	0.89
State Member 3	0.78
State Transitions	0.33
State Object	0.11
State Member 2	0.11
Pure State	0.11
Owner Transitions	0.11
Strategy	0.11
Exposed State	0.00

APPENDIX 3 VISITOR PATTERNS TEST

A3.1 Visitor Pattern Overview

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Visitor	Given an object structure containing elements, how can new operations be introduced without changing the element's class?	Encapsulate operations pertaining to each element class in a separate object, entitled visitor. The visitor is passed to elements as the object structure is traversed. As elements "accept" the visitor, the visitor is prompted to execute the appropriate operation.	Extensibility	New operations or algorithm behaviour should be added easily, while it is not expected that new element classes will be added frequently.	By adding visitor objects, new operations over complex objects or algorithmic behaviour can be introduced easily. However, adding new concrete elements entails added effort, as new operations need to be added to every visitor.
			Maintainability	Multiple algorithms apply to the elements of an object structure. Algorithm behaviour should be easily located for maintenance.	By gathering algorithmic behaviour in separate visitor objects isolates maintenance effort to specific visitors, which can be located easily.
			Response Time	Not an issue that would otherwise preclude use of this pattern.	The visitor object, introduces message indirection in order to initiate algorithm specific behaviour.
			Structural Flexibility	An object structure may be composed of elements of different classes. Any traversal of the object structure should be flexible, behaving according to the diversity of the object structure.	The visitor is not restricted by the composition of the object structure.
Acyclic Visitor	Using Visitor introduces dependency cycles. Element depends on Visitor depends on derivative of Element depends on Element.	Using multiple inheritance and dynamic cast the dependency cycle can be broken. Visitor becomes degenerate (no member functions). Create an abstract for each derivative of Element. Concrete visitor implements the specified methods. [36]	Response Time	Not an issue that would otherwise preclude use of this pattern.	Using "dynamic_cast" can be expensive in terms of runtime efficiency. Moreover, its efficiency may vary as the class hierarchy changes. Thus, Acyclic Visitor may be inappropriate in very tight real-time applications where runtime performance must be predictable." [36]

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Default Visitor	One difficulty in applying the Visitor pattern is modifying every concrete visitor whenever a new concrete element is added.	Default Visitor adds another level of inheritance to the Visitor pattern and thus provides a default implementation that takes advantage of the inheritance relationships in a polymorphic hierarchy of elements.	Extensibility	Element classes must be added easily.	"...easier to add new Concrete Element classes (compared to an ordinary visitor)...in cases where polymorphism is used to advantage... the default implementation of the new VisitXxxx() method in DefaultVisitor will often be adequate." [41]
Extrinsic Visitor	Visitor "does not directly provide a mechanism for determining whether a particular combination of concrete visitor and concrete element is meaningful." [41]		Maintainability	Not an issue that would otherwise preclude use of this pattern.	"The default visitor pattern requires more work in writing and maintaining class DefaultVisitor in return for less work in writing and maintaining each ConcreteVisitor class." [41]
			Extensibility	Element classes must be added easily.	"It is much easier to add new concrete classes to the Extrinsic Visitor pattern because only those concrete visitors that operate on object of the new class need to be changed." [41]
			Maintainability	"Visitor has a dependency cycle between each element and the abstract visitor." Hence making a change entails finding and implicates more classes." [41]	"eliminates the need to recompile every concrete element when a new concrete element is added to the hierarchy. ...not all concrete visitors need to be recompiled, only those that are modified to operate on objects of the new class." [41]
			Response Time	Not an issue that would otherwise preclude use of this pattern.	"... implements dynamic binding through a relatively brute-force lookup table that depends on the runtime type of concrete elements.... This approach may be considerably slower at runtime than the use of two virtual function tables in the Standard Visitor pattern." [41]

A3.2 Pattern Title: Visitor

Authors

E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Solution Structure

UML:Class.Visitor

Solution Behaviour

UML:Sequence.Visitor ExpectedBehaviour

Forces in Solution**Tangible**

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	2.50	2.00	4.00	1.67	-
WMC	0.00	1.50	1.17	2.00	0.98	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	2.00
DIT	0.00	0.50	0.50	1.00	0.55	-
NOC	0.00	0.00	0.50	2.00	0.84	-
CBO	0.00	1.00	1.00	2.00	0.89	-
MPC	0.00	1.00	0.83	2.00	0.75	-
DAC	0.00	0.00	0.17	1.00	0.41	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Extensibility	1
Structural Flexibility	1

A3.3 Pattern Title: Acyclic Visitor

Author

R.C. Martin [36]

Solution Structure

UML:Class.AcyclicVisitor

Solution Behaviour

UML:Sequence.AcyclicVisitor ExpectedBehaviour

Forces in Solution**Tangible**

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.50	4.00	1.69	-
WMC	0.00	0.50	0.88	2.00	0.99	-
NCC	-	-	-	-	-	5.00
NAC	-	-	-	-	-	3.00
DIT	0.00	0.00	0.38	1.00	0.52	-
NOC	0.00	0.50	0.63	2.00	0.74	-
CBO	0.00	0.50	0.75	2.00	0.89	-
MPC	0.00	0.50	0.63	2.00	0.74	-
DAC	0.00	0.00	0.13	1.00	0.35	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Extensibility	1
Structural Flexibility	1

A3.4 Pattern Title: Default Visitor

Author

M.E. Nordberg III [41]

Solution Structure

UML:Class.DefaultVisitor

Solution Behaviour

UML:Sequence.DefaultVisitor ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.63	5.00	1.92	-
WMC	0.00	1.50	1.38	2.00	0.74	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	4.00
DIT	0.00	0.50	0.63	2.00	0.74	-
NOC	0.00	0.50	0.50	1.00	0.53	-
CBO	0.00	1.00	1.13	3.00	1.25	-
MPC	0.00	0.50	0.63	2.00	0.74	-
DAC	0.00	0.00	0.25	2.00	0.71	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Extensibility	1
Structural Flexibility	1

A3.5 Pattern Title: Extrinsic Visitor

Author

M.E. Nordberg III [41]

Solution Structure

UML:Class.ExtrinsicVisitor

Solution Behaviour

UML:Sequence.ExtrinsicVisitor ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	2.00	6.00	2.55	-
WMC	1.00	1.00	1.40	2.00	0.55	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	2.00
DIT	0.00	0.00	0.40	1.00	0.55	-
NOC	0.00	0.00	0.40	1.00	0.55	-
CBO	0.00	0.00	0.80	2.00	1.10	-
MPC	0.00	0.00	0.60	2.00	0.89	-
DAC	0.00	0.00	0.20	1.00	0.45	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Extensibility	1
Structural Flexibility	1

A3.6 Testing Results

Decision Context

Preferences		Test Serial Preference Weights						
Force								
		1	2	3	4	5	6	7
Performance		1	0	0	0	5	7	3
Maintainability		0	1	0	0	5	3	7
Reliability		0	0	1	0	5	7	3
Utility		0	0	0	1	5	3	7

1 – Performance 5 – Balanced
2 – Maintainability 6 – Operations
3 – Reliability 7 – Maintenance
4 – Utility

Thresholds	
Name	Threshold Value
$\hat{c}_1$	0.70
$\hat{c}_2$	0.50
d	0.10

Candidates

Candidate	Hierarchy of Forces													
	Performance(+)				Maintainability(+)				Reliability(+)				Utility(+)	
	Response Time (-)				Size(-)				Fault Proneness(-)				Extensibility	
					Population(+)				Coupling(+)					
	RFC(+)				WMC (+)				Inheritance(+)				RFC (+)	
Visitor	2.00	1.17	4.00	2.00	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.17	2.00
Acyclic Visitor	1.50	0.88	5.00	3.00	0.38	0.63	0.38	0.63	0.75	0.63	0.63	0.63	0.13	1.50
Default Visitor	1.63	1.38	4.00	4.00	0.63	0.50	0.63	0.50	1.13	0.63	0.50	1.63	0.25	1.63
Extrinsic Visitor	2.00	1.40	3.00	2.00	0.40	0.40	0.40	0.40	0.80	0.60	0.40	2.00	0.20	2.00

Decision Results

Serial 1 - Performance	
Acyclic Visitor	1.00
Default Visitor	0.67
Visitor	0.00
Extrinsic Visitor	0.00

Serial 2 - Maintainability	
Visitor	1.00
Acyclic Visitor	0.67
Default Visitor	0.33
Extrinsic Visitor	0.00

Serial 3 - Reliability	
Extrinsic Visitor	1.00
Acyclic Visitor	0.67
Visitor	0.33
Default Visitor	0.00

Serial 4 - Utility	
Visitor	1.00
Acyclic Visitor	1.00
Default Visitor	1.00
Extrinsic Visitor	1.00

Serial 5 - Balanced	
Acyclic Visitor	1.00
Visitor	0.67
Default Visitor	0.00
Extrinsic Visitor	0.00

Serial 6 - Operations	
Acyclic Visitor	1.00
Extrinsic Visitor	0.67
Visitor	0.33
Default Visitor	0.00

Serial 7 - Maintenance	
Visitor	1.00
Acyclic Visitor	1.00
Default Visitor	0.00
Extrinsic Visitor	0.00

APPENDIX 4 PROXY PATTERNS TEST

A4.1 Summary of Pattern Information

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Proxy	How can a client's access to an object be controlled?	Introduce an intermediate object, entitled proxy that interacts with the real object. It appears to a client like the real object, but controls the client's access.	Response Time	Accessing the real object should be run time efficient.	The proxy limits the message indirection required to gain access to the real object.
Decorator	How to attach additional responsibilities to an object dynamically rather than to an entire class.	Subclass an abstract Decorator to an abstract Component. For a particular operation, the concrete Decorator invokes the concrete Component's operation with added behaviour implemented as required.	Transparency	The access control mechanism should appear like the real object to the client. From another point of view, the client should not have to change its behaviour when dealing with the access control mechanism.	The proxy, by inheriting from the same abstract class as the real object, will appear like the real object to any perspective client.
			Behavioural Flexibility	Requirement "to add responsibilities to individual objects dynamically..." [26]	"Decorator provides a more flexible way to add responsibilities to objects than can be had with static (multiple) inheritance. With decorators, responsibilities can be added and removed at run-time simply by attaching and detaching them." [26]
			Maintainability	Not an issue that would otherwise preclude use of this pattern.	"Results in systems composed of lots of little objects that all look alike. ...Although these systems are easy to customize by those who understand them, they can be hard to learn and debug." [26]
Null Object	"Sometimes an object that requires a collaborator does not need the collaborator to do anything. However, the object wishes to treat a collaborator that does nothing the same way it treats one that actually provides behavior" [57]	Introduce a Null object, which "... encapsulates the implementation decisions of how to do nothing and hides those details from its collaborators." [57]	Transparency	Requirement "to add responsibilities to individual objects... transparently, that is, without affecting other objects." [26]	"A decorator acts as a transparent enclosure for the concrete Component" [26]
			Reusability		"Multiple clients which all need their collaborators to do nothing can reuse the NullObject class so that they all do nothing the same way." [47]

Pattern Title	Problem	Solution	Force Name	Force In Problem	Force In Solution
Distributed Proxy	<p>"Constructing a design solution for distributed object communication is hard due to the complexity inherent to distributed communication... Furthermore, distributed communication spans several nodes with different name spaces such that names may not have the same meaning in each of the nodes." [47]</p>	<p>"The solution defines three layers: functional, logical, and physical. Functional layer contains the application functionalities and interactions that are normal object-oriented invocations. At the logical layer, proxy objects are introduced between distributed objects to convert object references into distributed names and vice-versa. ... Finally, the physical layer implements the distributed communication using the underlying communication mechanisms." [47]</p>	Distribution	<p>"Object references may be transparently passed between distributed nodes" [47]</p>	<p>"Object distribution is achieved because proxy objects convert names into references and vice versa." [47]</p>
Broker	<p>"... when distributed components communicate with each other, some means of inter-process communication is required." [13]</p>	<p>"Introduce a <i>broker</i> component to achieve better decoupling of clients and servers. Servers register themselves with the broker, and make their services available to clients through method interfaces. Clients access the functionality of servers by sending requests via the broker." [13]</p>	Response Time	<p>"The problem and respective solution is complex. Several aspects must be dealt with: the specificities of the underlying communication mechanism; and the diverse name spaces." [47]</p>	<p>"Complexity is managed by layered separation of problems. Logical layer supports name spaces and physical layer implements the underlying communication mechanisms."</p>
			Structural Flexibility	<p>"The resulting applications should be flexible in the incorporation and change of distribution issues. The underlying communication mechanisms should be isolated and it should be possible to provide different implementations." [47]</p>	<p>"Flexibility is achieved since the physical layer, which contains the underlying communication mechanisms particularities, is decoupled from logical layer." [47]</p>
			Transparency	<p>"The incorporation of distributed communication should be transparent for functional classes by preserving the interaction model, object-oriented interaction, and confining the number of changes necessary in functionality code." [47]</p>	<p>"... is achieved since logical and physical layers are decoupled from the functional layer. Functionality code uses transparently the logical layer..." [47]</p>
			Distribution	<p>"Components should be able to access services provided by others through remote, ... service invocations." [13]</p>	<p>"By using the Broker pattern, an application can access distributed services simply by sending message calls to the appropriate object, instead of focusing on low-level inter-process communication." [13]</p>
			Structural Flexibility	<p>"You need to exchange, add, or remove components at run-time" [13]</p>	<p>"... the Broker architecture is flexible, in that it allows dynamic change, addition, deletion, and relocation of objects." [13]</p>
			Transparency	<p>"Components should be able to access services provided by others through ..., location-transparent service invocations." [13]</p>	<p>"... introducing an object model in which distributed services are encapsulated within objects." [13]</p>

A4.2 Pattern Title: Proxy

Authors

Original: E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Adapted By: F. Buschmann, R. Meunier, H. Ronhert, P. Sommerlad, and M. Stal. [13]

Solution Structure

Solution Behaviour

UML:Sequence.Proxy ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.00	2.00	1.00	-
WMC	0.00	1.00	0.67	1.00	0.58	-
NCC	-	-	-	-	-	2.00
NAC	-	-	-	-	-	1.00
DIT	0.00	1.00	0.67	1.00	0.58	-
NOC	0.00	0.00	0.67	2.00	1.15	-
CBO	0.00	0.00	0.33	1.00	0.58	-
MPC	0.00	0.00	0.33	1.00	0.58	-
DAC	0.00	0.00	0.33	1.00	0.58	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Behavioural Flexibility	0
Distribution	0
Structural Flexibility	0
Transparency	1

A4.3 Pattern Title: Decorator

Authors

E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Solution Structure

Solution Behaviour

UML:Sequence.Decorator ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	0.50	1.00	3.00	1.41	-
WMC	0.00	0.50	0.75	2.00	0.96	-
NCC	-	-	-	-	-	2.00
NAC	-	-	-	-	-	2.00
DIT	0.00	1.00	1.00	2.00	0.82	-
NOC	0.00	0.50	0.75	2.00	0.96	-
CBO	0.00	0.50	0.75	2.00	0.96	-
MPC	0.00	0.00	0.25	1.00	0.50	-
DAC	0.00	0.00	0.25	1.00	0.50	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Behavioural Flexibility	1
Distribution	0
Structural Flexibility	0
Transparency	1

A4.4 Pattern Title: Null Object

Authors

E. Gamma, R. Helm, R. Johnson, and J. Vlissides [26]

Solution Structure

Solution Behaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	1.00	1.00	2.00	0.82	-
WMC	0.00	1.00	0.75	1.00	0.50	-
NCC	-	-	-	-	-	3.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.50	0.50	1.00	0.58	-
NOC	0.00	0.00	0.50	2.00	1.00	-
CBO	0.00	0.00	0.25	1.00	0.50	-
MPC	0.00	0.00	0.25	1.00	0.50	-
DAC	0.00	0.00	0.25	1.00	0.50	-

Intangible

Intangible Force	Supported
Behavioural Flexibility	0
Distribution	0
Structural Flexibility	0
Transparency	1

A4.5 Pattern Title: Distributed Proxy

Authors

A.R. Silva, F.A. Rosa, T. Gonçalves, M. Antunes [47]

Solution Structure

UML:Class.DistributedProxy

Solution Behaviour

UML:Sequence.DistributedProxy ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	4.00	3.13	6.00	2.47	-
WMC	0.00	3.00	2.13	4.00	1.55	-
NCC	-	-	-	-	-	6.00
NAC	-	-	-	-	-	2.00
DIT	0.00	0.50	0.50	1.00	0.53	-
NOC	0.00	0.00	0.50	2.00	0.93	-
CBO	0.00	0.50	0.88	2.00	0.99	-
MPC	0.00	0.50	1.25	4.00	1.58	-
DAC	0.00	0.50	0.88	2.00	0.99	-
LCOM	0.00	0.00	0.25	1.00	0.46	-

Intangible

Intangible Force	Supported
Behavioural Flexibility	0
Distribution	1
Structural Flexibility	1
Transparency	1

A4.6 Pattern Title: Broker

Authors

F. Bushmann, R. Meunier, H. Ronhert, P. Sommerlad, and M. Stal. [13]

Solution Structure

UML:Class.Broker

Solution Behaviour

UML:Sequence.Broker ExpectedBehaviour

Forces in Solution

Tangible

Metric	Min	Median	Mean	Max	Std Dev	Total
RFC	0.00	5.00	3.80	6.00	2.68	-
WMC	0.00	4.00	2.60	4.00	1.95	-
NCC	-	-	-	-	-	4.00
NAC	-	-	-	-	-	1.00
DIT	0.00	0.00	0.40	1.00	0.55	-
NOC	0.00	0.00	0.40	1.00	0.55	-
CBO	0.00	2.00	2.00	4.00	1.41	-
MPC	0.00	2.00	1.40	2.00	0.89	-
DAC	0.00	1.00	1.60	4.00	1.52	-
LCOM	0.00	0.00	0.00	0.00	0.00	-

Intangible

Intangible Force	Supported
Behavioural Flexibility	0
Distribution	1
Structural Flexibility	1
Transparency	1

A4.7 Testing Results

Decision Context

Preferences									
Force	Test Serial Preference Weights								
	1	2	3	4	5	6	7		
Performance	1	0	0	0	5	7	3		
Maintenance Effort	0	1	0	0	5	3	7		
Reliability	0	0	1	0	5	7	3		
Utility	0	0	0	1	5	3	7		

1 – Performance
2 – Maintainability
3 – Reliability
4 – Utility
5 – Balance
6 – Operations
7 – Maintenance

Thresholds	
Name	Threshold Value
$\hat{c}_1$	0.70
$\hat{c}_2$	0.50
d	0.10

Candidates

	Hierarchy of Forces																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
	Performance(+)				Maintainability(+)				Reliability(+)				Utility(+)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
	Response Time (-)				Size(-)		Reusability(+)		Fault Proneness(-)				Behavioural Flexibility		Distribution		Structural Flexibility		Transparency																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
					Population(+)		Inheritance(+)		Inheritance(+)		Coupling(+)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																															
	RFC(+)				WMC (+)		NCC (+)		NAC(+)		DIT (+)		NOC (+)		DIT (+)		NOC (+)		RFC (+)		DAC (+)		MPC (+)		CBO (+)																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
Candidate																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										

Decision Results

Serial 1 - Performance	
Proxy	1.00
Decorator	1.00
Null Object	1.00
Distributed Proxy	0.50
Broker	0.00

Serial 2 - Maintainability	
Decorator	1.00
Proxy	0.75
Null Object	0.25
Distributed Proxy	0.25
Broker	0.00

Serial 3 - Reliability	
Null Object	1.00
Broker	0.75
Proxy	0.00
Decorator	0.00
Distributed Proxy	0.00

Serial 4 - Utility	
Distributed Proxy	1.00
Broker	1.00
Decorator	0.67
Proxy	0.00
Null Object	0.00

Serial 5 - Balanced	
Decorator	1.00
Null Object	0.75
Proxy	0.50
Distributed Proxy	0.00
Broker	0.00

Serial 6 - Operations	
Null Object	1.00
Decorator	0.75
Proxy	0.50
Broker	0.25
Distributed Proxy	0.00

Serial 7 - Maintenance	
Decorator	1.00
Distributed Proxy	0.75
Proxy	0.25
Broker	0.25
Null Object	0.00

APPENDIX 5 XML AND PATTERN DECISION AID PERSISTENCE

The eXtensible Markup Language (XML) has been applied in this thesis to extend an existing pattern related XML structure[25], which defines the pattern document. This appendix introduces sufficient information to read the XML documents developed as part of the pattern decision aid persistence strategy. Although, not comprehensive, links have been provided to related sources where more information is available.

```
<!--comment syntax --!>
<!--file declaration --!>
<?xml version='1.0' encoding='us-ascii'?>

<!-- A main tag defined to include a sequence of (subElements) --!>
<!ELEMENT mainElement (elementContents, element2?, element3*, element4+)>
<!-- ", " implies 'and/preceded by', whereas "|" implies 'or/optional' --!>
<!-- The content elements are suffixed with a cardinality of:
      "blank" one and only one
      "?" zero to one
      "*" zero to many
      "+" one to many--!>
<!--elementContents defined to not have any subElements "EMPTY" --!>
<!ELEMENT elementContents EMPTY>

<!--the attribute list for elementContents --!>
<!ATTLIST elementContents

      <!-- "ID" must be unique in file, "IDREF" refers to unique "ID" --!>
      <!--"#REQUIRED" implies the attribute must have a value to be a valid element--!>
      name ID #REQUIRED

      <!--A choice of values with a "option2" specified as a default --!>
      attribute1 (option1 | option2 ) "option2"

      <!--Character data, can be "PCDATA" for Parsed Character DATA --!>
      attribute2 CDATA #IMPLIED
>
<!ELEMENT element2 EMPTY>
<!ATTLIST element2
      name IDREF #IMPLIED
>...
```

Listing 2 - Example of DTD Syntax

A5.1 Guide to Document Type Definition (dtd) Syntax

Listing 2 provides a sample of the common syntax used in specifying a document type definition (dtd). Other than the necessary declarations, the main requirement in specifying a dtd is defining elements and attributes. Defining elements is equivalent to defining data entities, and corresponds to the main tags encountered within a XML file. Elements can also form a hierarchy of tags. Alternatively, attributes will contain

information typically associated with a specific entity. A typical decision is whether to represent a particular structure with elements or attributes. Certain items may be included as individual tags to represent sub-elements of a particular entity, with an entity's attributes defined in the appropriate attribute list `<!ATTLIST>`. In the `force.dtd` developed, separate sub-elements are used for associative entities or variables that are specific to certain types of an entity. For more details concerning the syntax of the XML dtd refer to: <http://www.informatik.tu-darmstadt.de/DVS1/staff/bouret/xml/xmlldtd.html>.

A5.2 Guide to XML Syntax

The `.xml` file represents a particular instance complying with the dtd. The name of a dtd must be consistent with the main tag applicable to a XML file. Listing 3 provides an example of the definition of the main element. It should be noted that the element contents in brackets have been defined such that each element must exist in the implemented XML file for the file to be valid. For an annotated version of the XML specification refer to: <http://www.xml.com/axml/axml.html>

```
<?xml version='1.0' encoding='us-ascii'?>

<!--Declare dtd to be used. If using an internal dtd it is declared here --!>
<!DOCTYPE mainElement SYSTEM "../path/./mainElement.dtd">
<!-- opening tag --!>
<mainElement>
    <!--subElements --!>
    <elementContents name="UniqueName" attribute2="uses default"/>
    <element2 name="referenceToName"/>
<!--closing tag --!>
</mainElement>
```

Listing 3 - Example of Resulting XML Syntax

A5.3 Using XML as Part of the Pattern Decision Aid Persistence Strategy

In the context of the pattern repository a Pattern dtd is currently used to specify the structure for a pattern document. As XML can also be useful for representing Entity Relationship Data Models, and to remain consistent with the current repository, XML is applied for persistence of candidate pattern information relevant to the pattern decision

aid. This decision is further warranted by the static nature of pattern information. That is, once a pattern's structure and behaviour have been specified in accordance with the guidelines of section 9.1 , subsequent measurement will produce metric values that are not likely to change.

```
<?xml version='1.0' encoding='us-ascii'?>
<!--ELEMENT forces (force+,metrics)-->
<!--ELEMENT force (influence*,measure*,intangibleIndicator*)-->
<!--ATTLIST force
      name ID #REQUIRED
      type (general | tangible | intangible) "tangible"
      isRoot (true | false) "false"
-->
<!--ELEMENT intangibleIndicator EMPTY-->
<!--ATTLIST supported (true | false) "false"
-->
<!--ELEMENT influence EMPTY-->
<!--ATTLIST influence
      target IDREF #REQUIRED
      impact (negative | neutral | positive) "positive"
-->
<!--ELEMENT measure EMPTY-->
<!--ATTLIST measure
      metricID IDREF #REQUIRED
-->
<!--ELEMENT metrics (metric+)-->
<!--ELEMENT metric (influence*)-->
<!--ATTLIST metric
      id ID #REQUIRED
      name CDATA #IMPLIED
      minValue CDATA #IMPLIED
      meanValue CDATA #IMPLIED
      medianValue CDATA #IMPLIED
      maxValue CDATA #IMPLIED
      stdDevValue CDATA #IMPLIED
      totalValue CDATA #IMPLIED
-->
```

Listing 4 - forces.dtd

Based on this decision the developed dtd related to force hierarchies is included in Listing 4. Recall from Chapter 10 a force shadow separates the decision analysis information from the actual pattern document structure. Two force shadows have been used in the pattern decision aid design:

1. The Common Force Shadow contains information concerning the specific layout of the force hierarchy; and

2. The Pattern Force Shadow contains information for each candidate related to the metric values measured.

Table 30 contains a description of each of the elements defined, including a note concerning which of the force shadow xml files the element will appear in.

Table 30 - forces.dtd Element Descriptions

Element Name: forces	Applicable To: - Common Force Shadow - Pattern Force Shadow
Description: A collection of data related to each force as well as those metrics applicable to either the force hierarchy or the individual pattern.	
Element Name: force	Applicable To: - Common Force Shadow - Pattern Force Shadow
Description: - <u>Common Force Shadow</u> . The relative position of a force in the force hierarchy is conveyed by references to metrics that account for it and the other forces the particular force influences. - <u>Pattern Force Shadow</u> . Specific information pertaining to the candidate pattern regarding each force is dependent on the force type. An intangible must have an indication whether it is supported. A tangible force will refer to those metrics that measure the particular force. Finally, in the case of a general force, the element will only have contents if a metric directly measures it.	
Element Name: influence	Applicable To: - Common Force Shadow
Description: In a force hierarchy relationships exist between metrics and forces as well inter-force relationships. In essence as a metric value or force changes it will influence another force. The direction of change based on this influence is known as an impact and is used to determine whether the metric is a cost or benefit criterion.	
Element Name: intangibleIndicator	Applicable To: - Pattern Force Shadow
Description: The intangible force does not have a metric, but rather an indication of whether the pattern is supported or not. It is expected that a simple Boolean is sufficient ("0" or "1")	
Element Name: measure	Applicable To: - Pattern Force Shadow
Description: A metric can indicate different forces just as a force is measured by many metrics. Consequently the measure is an associative entity that separates the many-to-many relationship.	
Element Name: metrics	Applicable To: - Common Force Shadow - Pattern Force Shadow
Description: Specific metric related information is contained in this collection.	
Element Name: metric	Applicable To: - Common Force Shadow - Pattern Force Shadow
Description: - <u>Common Force Shadow</u> . The forces it influences define the orientation of a metric. This information is useful to determining whether a metric is a cost or benefit criterion. - <u>Pattern Force Shadow</u> . The actual measured values pertaining to a candidate pattern are included as attributes to the metric.	

In order to illustrate the resulting files required by the pattern decision, Listings 5 and 6 represent samples of the common force shadow and pattern force shadow respectively.

```
<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE forces SYSTEM "forces.dtd">
<forces>
  <force name="Total_Satisfaction" isRoot="true" type="general"></force>
  <force name="Performance" type="general">
    <influence target="Total_Satisfaction"/>
  </force>
  <force name="Message_Indirection">
    <influence target="Performance" impact="negative"/>
    <measure metricID="RFC"/>
  </force>
  <force name="Maintainability_Effort" type="general">
    <influence target="Total_Satisfaction" impact="negative"/>
  </force>
  <force name="Size" type="general">
    <influence target="Maintainability_Effort"/>
  </force>
  <force name="Population">
    <influence target="Size"/>
    <measure metricID="WMC"/>
    <measure metricID="NCC"/>
  </force>
  <force name="Reusability">
    <influence target="Maintainability" impact="negative"/>
    <measure metricID="NAC"/>
  </force>
  <force name="Inheritance">
    <influence target="Reusability"/>
    <influence target="Fault_Proneness" impact="negative"/>
    <measure metricID="DIT"/>
    <measure name="NOC"/>
  </force>
  <force name="Reliability">
    <influence target="Total_Satisfaction"/>
  </force>
  <force name="Fault_Proneness">
    <influence target="Reliability" impact="negative"/>
  </force>
  <force name="Coupling">
    <influence target="Fault_Proneness"/>
    <measure metricID="MPC"/>
    <measure metricID="RFC"/>
    <measure metricID="CBO"/>
    <measure metricID="DAC"/>
  </force>
  <force name="Utility">
    <influence target="Total_Satisfaction"/>
  </force>
  <force name="Data_Sharing" type="intangible">
    <influence target="Utility" impact="neutral"/>
  </force>
  <force name="Dynamic_Transitions" type="intangible">
    <influence target="Utility" impact="neutral"/>
  </force>
  <force name="Transparency" type="intangible">
    <influence target="Utility" impact="neutral"/>
  </force>
  <force name="Distribution" type="intangible">
    <influence target="Utility" impact="neutral"/>
  </force>
  <metrics>
    <metric id="MPC" name="Message Passing Coupling">
      <influence name="Message_Indirection"/>
      <influence name="Coupling"/>
    </metric>
  </metrics>
</forces>
```

```

</metric>
<metric id="WMC" name="Weighted Methods Per Class">
  <influence name="Population"/>
</metric>
<metric id="NKC" name="Number of Key Classes">
  <influence name="Population"/>
</metric>
<metric id="NSC" name="Number of Support Classes">
  <influence name="Population"/>
</metric>
<metric id="NAC" name="Number of Abstract Classes">
  <influence name="Reusability"/>
</metric>
<metric id="DIT" name="Depth in Inheritance Tree">
  <influence name="Inheritance"/>
</metric>
<metric id="NOC" name="Number of Children">
  <influence name="Inheritance"/>
</metric>
<metric id="RFC" name="Response For Call">
  <influence name="Coupling"/>
</metric>
<metric id="CBO" name="Coupling Between Object Classes">
  <influence name="Coupling"/>
</metric>
<metric id="DAC" name="Data Abstraction Coupling">
  <influence name="Coupling"/>
</metric>
<metric id="Data_Sharing" name="Data Sharing">
  <influence name="Utility" impact="neutral"/>
</metric>
<metric id="Dynamic_Transitions" name="Dynamic Transitions" type="intangible">
  <influence name="Utility" impact="neutral"/>
</metric>
<metric id="Transparency" name="Transparency" type="intangible">
  <influence name="Utility" impact="neutral"/>
</metric>
<metric id="Distribution" name="Distribution" type="intangible">
  <influence name="Utility" impact="neutral"/>
</metric>
</metrics>
</forces>

```

Listing 5 - XML Sample of Common Pattern Force Shadow

In Listing 6 there exist certain data elements that are redundant to the common force shadow. The redundancy is intended to promote a data structure consistent with the expected candidate dictionary into which force measures will be parsed as part of the pattern decision aid. Although, at a minimum each candidate must have separate files for metric values, it may be reasonable to reduce the redundancy as the pattern decision aid design is further refined. In particular this applies to the repetition of measure references and force attributes.

```

<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE forces SYSTEM "forces.dtd">
<forces>
  <force name="Total_Satisfaction" type="general" isRoot="true"></force>
  <force name="Performance" type="general"></force>
  <force name="Message_Indirection">
    <measure metricID="RFC"/>
  </force>
  <force name="Maintainability" type="general"></force>
  <force name="Size"></force>
  <force name="Population">
    <measure metricID="WMC"/>
    <measure metricID="NCC"/>
  </force>
  <force name="Reusability" type="general">
    <measure metricID="NAC"/>
  </force>
  <force name="Inheritance">
    <measure metricID="DIT"/>
    <measure name="NOC"/>
  </force>
  <force name="Reliability" type="general"></force>
  <force name="Fault_Proneness"></force>
  <force name="Coupling">
    <measure metricID="MPC"/>
    <measure metricID="RFC"/>
    <measure metricID="CBO"/>
    <measure metricID="DAC"/>
  </force>
  <force name="Utility" type="general"></force>
  <force name="Data_Sharing" type="intangible">
    <intangibleIndicator/>
  </force>
  <force name="Dynamic_Transitions" type="intangible"/>
    <intangibleIndicator/>
  </force>
  <force name="Transparency" type="intangible"/>
    <intangibleIndicator/>
  </force>
  <force name="Distribution" type="intangible"/>
    <intangibleIndicator/>
  </force>
  <metrics>
    <metric id="RFC" meanValue="1.00"></metric>
    <metric id="MPC" meanValue="0.25"></metric>
    <metric id="WMC" meanValue="0.75"></metric>
    <metric id="NCC" totalValue="3.00"></metric>
    <metric id="NAC" totalValue="1.00"></metric>
    <metric id="DIT" meanValue="0.50"></metric>
    <metric id="NOC" meanValue="0.50"></metric>
    <metric id="CBO" meanValue="0.75"></metric>
    <metric id="DAC" meanValue="0.25"></metric>
  </metrics>
</forces>

```

Listing 6 - Sample of a Candidate Pattern's Force Shadow

APPENDIX 6 PATTERN DECISION AID DESIGN REPORT

This appendix represents the state of the pattern decision aid design at the submission of this thesis. It is intended to provide sufficient information concerning the envisaged pattern decision aid, that subsequent work to this thesis can be applied to modifying the design or continuing relevant development effort.

This report will not reintroduce concepts concerning the approach to the design, previously introduced in Chapter 8. It is therefore recommended that this chapter be used in conjunction with Chapter 8. It should be noted that not all elements of the design have been completely refined. To note these elements the tag **<not included in this iteration>** will be used. Furthermore, there may exist the potential for future enhancements that now fall outside the current scope, likewise **<outside current scope>** will denote these items.

A6.1 System Objective

The objective of developing a pattern decision aid is to allow a pattern user to compare two or more selected patterns based on multiple forces.

A6.2 Report Structure

Rather than organizing the appendix according to the construction activity responsible for a particular artifact, the structure of this appendix is logically organized according to scenarios once the Requirements Catalogue, Domain Structure and Use Case diagrams have been specified. Accordingly, each scenario section will have a Scenario Textual Description and UML Sequence Diagram associated to it. In summary the following is a breakdown of section elements and associated artifacts that comprise this appendix:

1. Requirements Catalogue. Includes a table, as outlined in Chapter 8, that captures assumptions and requirements identified, that pertain to the pattern decision aid.

2. Use Case Diagram. Includes a UML Use Case diagram distinguishing between main scenarios that comprise the workload of the pattern decision aid, along with interactions with external entities or actors.
3. Domain Structure. Associated artifacts include a UML class diagram representing main domain classes and relevant associations. This diagram is accompanied by a set of Class, Responsibility, Collaboration tables consistent with the format specified in Chapter 8.
4. Pattern Decision Aid Behaviour. This section is subdivided according to the scenarios identified in the Use Case diagram. Each subsection is comprised of a Scenario Textual Description (STD) and related UML Sequence Diagram. The STD tables are structured in accordance with the specification introduced in Chapter 8. Two conventions have been used in the UML Sequence diagrams. In order to promote traceability, a reference to the relevant responsibility sequence number from the corresponding STD has been annotated on each diagram. Furthermore, where an message typically implies invoking a simple query method, an explicit return arrow is not drawn. UML 1.3 supports the latter convention.

A6.3 Requirements Catalogue

ID	Title	Description	Source Reference
000-A	Assumptions	The 000 – 099 series represent the assumptions made during the course of this project.	
001-A	Consistent Pattern Forces	The force shadows: applicable to candidate patterns is based on a consistent hierarchy.	June 7, 2000
002-A	Consistent Pattern Metrics	The metrics used to measure candidate patterns is based on a consistent hierarchy.	June 7, 2000
003-A	Force influenced by Duplicate Tangible Forces	If a force influences two different forces (typically Tangible Force) then duplicate entries will exist in the candidate's Pattern Force Shadow.	June 9, 2000
004-A	Influences among forces at the same	For the purposes of considering the decision analysis problem, influences among forces at the	June 15, 2000

2. Use Case Diagram. Includes a UML Use Case diagram distinguishing between main scenarios that comprise the workload of the pattern decision aid, along with interactions with external entities or actors.
3. Domain Structure. Associated artifacts include a UML class diagram representing main domain classes and relevant associations. This diagram is accompanied by a set of Class, Responsibility, Collaboration tables consistent with the format specified in Chapter 8.
4. Pattern Decision Aid Behaviour. This section is subdivided according to the scenarios identified in the Use Case diagram. Each subsection is comprised of a Scenario Textual Description (STD) and related UML Sequence Diagram. The STD tables are structured in accordance with the specification introduced in Chapter 8. Two conventions have been used in the UML Sequence diagrams. In order to promote traceability, a reference to the relevant responsibility sequence number from the corresponding STD has been annotated on each diagram. Furthermore, where an message typically implies invoking a simple query method, an explicit return arrow is not drawn. UML 1.3 supports the latter convention.

A6.3 Requirements Catalogue

ID	Title	Description	Source Reference
000-A	Assumptions	The 000 – 099 series represent the assumptions made during the course of this project.	
001-A	Consistent Pattern Forces	The force shadows applicable to candidate patterns is based on a consistent hierarchy.	June 7, 2000
002-A	Consistent Pattern Metrics	The metrics used to measure candidate patterns is based on a consistent hierarchy.	June 7, 2000
003-A	Force influenced by Duplicate Tangible Forces	If a force influences two different forces (typically Tangible Force) then duplicate entries will exist in the candidate's Pattern Force Shadow.	June 9, 2000
004-A	Influences among forces at the same	For the purposes of considering the decision analysis problem, influences among forces at the	June 15, 2000

ID	Title	Description	Source Reference
	hierarchy level.	same level in the hierarchy (peers) are not considered.	
100-F[NF]	System	The 100 – 199 series represent high-level system requirements of either the functional [F] or non functional variety. [NF]	
101-F	System Objective	Must develop a system that allows a pattern user to compare two or more selected patterns based on multiple forces.	June 7, 2000
102-NF	Distributed Environment	The Pattern Decision Aid is to be made available over the WWW.	June 8, 2000
103-NF	Implementation Environment	The Pattern Decision Aid must be implemented in an environment consistent with the existing Carleton School of Computer Science pattern repository. That is java servlet platform JRun, XML, and XML parser based on Sun's ProjectX (jaxp).	Thesis Motivation
200-F[NF]	Decision Analysis	The 200-299 series represent requirements that are related to the specification of the decision analysis approach.	
201-F	Decision Analysis Approach	The software is to support an outranking method for decision analysis. In particular the comparison approach is to be consistent with the Electre II method.	Meeting Thesis Advisor [24 May 2000]
202-F	Criteria Hierarchy	Based on the nature of the forces a hierarchy of criteria is to be applied with user preferences to be assigned only to the highest level.	June 8, 2000
203-F	Pairwise Concordance and Discordance	During a comparison of two candidate patterns, pairwise concordance values are calculated to indicate the amount of agreement that candidate <i>a</i> is preferred to candidate <i>b</i> . As well a discordance value is calculated to account for the amount of disagreement that candidate <i>a</i> is preferred to candidate <i>b</i> . The reverse of these indices must also be calculated.	- Chapter 6 - Pattern decision analysis method step 1.2
204-F	Out-flow and In-flow Indices	Based on tested concordance and discordance values, a strong and weak out-flow as well as in-flow, representing the net preference of candidate <i>a</i> over <i>b</i> is calculated.	- Chapter 6 - Pattern decision analysis method step 1.3
205-F	Number of Outranks	Based on analyzed out and in flows the number of candidates that a particular candidate outranks is calculated. In the end this value is used to sort candidates from most preferred to least preferred.	- Chapter 6 - Pattern decision analysis method step 2.1 and output.
300-F[NF]	Forces	The 300-399 series represent requirements that are related to the specification of the treatment of pattern forces.	
301-F	Force Hierarchy	Forces are inter-related in the context of a force hierarchy, consistent with the criteria hierarchy. Moving up the force hierarchy acts to determine the decision results.	- Chapter 6 - Pattern decision analysis method
302-F	Tangible Forces	A force that can be measured. Comparing candidates based on tangible forces entails comparing the variety of metric values used to indicate the presence of the force.	- Chapter 4 - Chapter 6 – Pattern decision analysis method

ID	Title	Description	Source Reference
303-F	Intangible Forces	A force that cannot be readily accounted by a metric value. Typically such forces contribute to the Utility of an associated pattern. Comparing candidates based on intangible forces entails comparing whether or not the pattern supports the intangible and whether the decision maker has requested the intangible.	- Chapter 4 - Chapter 6 – Pattern decision analysis method
400-F[NF]	Metrics		
401-F	Metric Role	The tangible forces are indicated by Object-Oriented Metrics. A comparison of tangible forces entails comparing all metric values.	- Chapter 4 - Chapter 5
402-F	Metric Influences	For a force hierarchy to be meaningful, there must be an indication to the extent that a particular force is present in the pattern solution. The metric fulfills this role. However, the metric in turn may have different impacts on the force that it actually indicates.	- Chapter 5
403-F	Normalizing Metrics	Beginning with the metrics in the updated force hierarchy, metric values will be normalized as explained in the extended ELECTRE II method covered in section 2.4.	- Chapter 6 - Pattern decision analysis method step 1.1
404-F	Number of Outranks	A particular visit to a force node in the force hierarchy is completed by generating the number of outranks a candidate possesses which indicates that the candidate in question has a more dominant strong out-flow and lower strong in-flow than the other candidates. As move up the hierarchy these are normalized as benefit criteria.	- Chapter 6 - Pattern decision analysis method step 2.1
500-F[NF]	Candidates	The 400-499 series represent requirements that are related to the specification of the candidates.	
501-F	Candidate Definition	The candidates are those patterns to be evaluated during the decision procedure. They are represented by measured values corresponding to specified forces.	- Chapter 6 - Pattern decision analysis input
600-F[NF]	Context	The 500-599 series represent requirements that are related to the specification of the context of the decision maker's preferences.	
601-F	Thresholds	Two concordance and one discordance thresholds must be input by the pattern user.	- Chapter 6 - Pattern decision analysis input
602-F	Preferences	The pattern user will assign preference to the highest level of forces in the related hierarchy.	- Chapter 6 - Pattern decision analysis input
900-F[NF]	Miscellaneous	The 900-999 series represent requirements that are considered miscellaneous and do not fit into the other classifications.	

A6.4 Use Case Diagram

The decision aid has been designed to work in conjunction with the existing pattern repository search capabilities. It is expected that the user will have conducted a search and displayed output will be suffixed with a check box indicating: “ Select to Compare”. At the bottom of the list will be a button entitled: “Compare selected candidates”. Upon depressing the button the decision aid is initiated. Subsequently a number of steps are required to set up the actual pattern decision problem. This set up will require user input concerning preference for forces and intangibles. Once the pattern decision problem has

Use Case: pattern.decisionAidServlet

been constructed the pattern decision analysis is executed by traversing the force hierarchy in a post-order manner with each visit comprising of a comparison and analysis stage.

Class Responsibilities and Collaborations

Class Name: Candidate	
Description: Each candidate pattern is represented by a profile of metrics. This class serves to keep associated data organized.	Collaborations: - contains many ForceKey/Force pairs
Responsibilities: - Has local knowledge of the measured forces pertaining to a candidate pattern. - A candidate is assigned the comparison with another candidate. - Initiates comparison with assigned candidate.	

Class Name: Context	
Description: A particular decision maker operates in a specific context, tempered by their preferences for certain criteria/forces and thresholds concerning the agreement and disagreement that candidate <i>a</i> is preferred to candidate <i>b</i> .	Collaborations: - contains many ForceKey/Preference pairs
Responsibilities: <ul style="list-style-type: none">- Is used to capture the decision maker's input.- Has local knowledge concerning a decision maker's preferences.- Tests whether a given assertion that candidate <i>a</i> is preferred to candidate <i>b</i> meets specified thresholds.	

Class Name: DecisionAid	
Description: A class that coordinates the input, running pattern decision analysis, and output of results.	Collaborations: <ul style="list-style-type: none">- Parser- contains aForceHierarchy- contains many Candidate- contains aContext- Pattern.DecisionAidViewer
Responsibilities: <ul style="list-style-type: none">- Handles object creation and central knowledge of main domain classes.- Triggers input of Context (decision preferences/thresholds)- Triggers building Candidate profiles.- Triggers comparison.- Triggers output of results.	

Class Name: Force	
Description: A force represents a decision criteria that can be used to represent a particular point of view concerning a pattern decision problem.	Collaborations: - must have a forceType
Responsibilities: - Records result from ongoing comparisons.	

Class Name: ForceHierarchy	
Description: A basic description of the class.	Collaborations: - extends Hierarchy - organizes forceNodes
Responsibilities: - Has knowledge about the structure of the hierarchy applicable to the pattern decision problem. - Coordinates the traversal of the hierarchy during the actual decision method.	

Class Name: ForceKey	
Description: A unique identifier of a force.	Collaborations: - Implements java.Comparable.
Responsibilities: - acts as a key in all TreeMaps for the force – preference mappings and force-force mappings.	

Class Name: ForceNode	
Description: Represents a force's position within a force hierarchy.	Collaborations: - assigns comparison to a candidate - contained in a forceHierarchy
Responsibilities: - Coordinates overall pairwise comparison and analysis of candidates during a triggered visit. - Given a set of candidates, it coordinates the assigning of comparisons to candidates during a visit. - Given a set of compared candidates, the assigning of analysis to candidates during a visit is coordinated.	

Class Name: ForceType	
Description: An abstract class declaring methods to be implemented by concrete sub-classes.	Collaborations: - nil
Responsibilities: - see implementing classes Tangible, General, and Intangible	

Class Name: General	
Description: A concrete Force type that be measured and/or have child Forces influence it.	Collaborations: - extends Tangible. - queries/updates a given General.
Responsibilities: - Handles the actual comparison with another candidate's General.	

Class Name: <<utility>> Hierarchy	
Description: A general tree data structure.	Collaborations:
Responsibilities: - Basic responsibilities typical of a general tree data structure.	

Class Name: Influence	
Description: Associative object possessing data concerning impact of a metric or force on another force.	Collaborations: - relates a Metric to a Force; xor - relates a Force to a Force
Responsibilities: - Has local knowledge concerning metric-force xor force-force associations and impact of the association.	

Class Name: Intangible	
Description: A concrete Force type that cannot be measured, but whose presence can be indicated.	Collaborations: - implements ForceType. - queries/updates a given Intangible.
Responsibilities: - Handles the actual comparison with another candidate's Intangible.	

Class Name: Metric	
Description: A structure can be measured according to certain metrics. The associated values indicate how much a particular tangible or general force is present in a OO structure.	Collaborations: - nil
Responsibilities: - Given a metricNorm or alternatively the high and low candidate values for the metric, calculates a normalized value.	

Class Name: MetricNorm	
Description: To calculate normalized metric values, the metric norm records high and low metric values.	Collaborations: - nil
Responsibilities: - Has local knowledge of the highest and lowest metric values among all candidates.	

Class Name: Preference	
Description: Each force will be assigned a preference weight. Some weights will be assigned by the decision maker (level 1 in forceHierarchy) and others will be internally assigned a common weight (1)	Collaborations: - nil
Responsibilities: - Has local knowledge of a decision maker's preference for a particular force.	

Class Name: Tangible	
Description: A concrete Force type that can be measured.	Collaborations: - implements ForceType. - queries/updates a given Tangible.
Responsibilities: - Handles the actual comparison with another candidate's Tangible.	

A6.6 Pattern Decision Aid Behaviour

Scenario: Initiate Decision Aid

STD Identifier: Initiate Decision Aid	Requirement Reference
Description: Create the initial representation of the candidate patterns to be compared.	
External Actors: Pattern User	
Precondition: It is envisaged that the Pattern User has searched and selected candidate patterns from the pattern repository. This implies the output from the searcher needs to include a "select for comparison" selection box beside each pattern name in the search results and a "compare selected Patterns" button.	
Triggering Event: The pattern user presses the compare selected patterns option.	
Sequence of Responsibilities: R1. (R1.1) A collection of pattern string names that have been selected forms the request for service that is sent to pattern.decisionAidServlet. (R1.2)The pattern.decisionAidViewer is created along with the (R1.3) pattern.decisionAid. The pattern.decisionAidServlet initializes the decisionAid with the requested collection of patternNames. <not included in this iteration> R2. (R2.1)As part of initialization the decisionAid creates a context a candidate object (see comments). (R2.2)The decisionAid then sequences through the collection of requested patternNames and (R2.3) creates a respective candidate, initialized with the associated name. (R2.4)The created candidate is then added to the decisionAid's collection of candidates. R3. (R3.1)Upon reaching the end of the requested patternNames check the size of the decisionAid's collection of candidates. (R3.2)If there are two or more candidates the decisionAidServlet signals the decisionAid to buildContext().	103-NF
Postcondition: The decisionAid collection of candidates contains two or more Candidate objects initialized with a pattern name.	
Resulting Event: The decisionAid responds to buildContext() signal.	
Alternatives: Alt1. If candidates are empty indicate to user that no patterns were selected and should be before launching the pattern.decisionAidServlet. If there is only one candidate, indicate that there are not sufficient candidate patterns selected to enable a warranted comparison.	
Nonfunctional Requirements: Enter a set of nonfunctional requirements that apply to the scenario (optional)	
Comments: R2.2 The candidate object is entitled norm in order to collect the normalization factors to be used in normalizing candidate metrics. The metric values stored include the global max and global minimum encountered (refer to Build Candidates Profile).	

UML:Sequence Initiate Decision Aid

Scenario: Build Force Hierarchy

STD Identifier: Build Force Hierarchy <not included in this iteration>	Requirement Reference
Description: To extract the force hierarchy to be used for the purposes of the comparison.	
External Actors: Common Pattern Force Shadow	
Precondition: The decisionAid must be initialized.	
Triggering Event: The decisionAid receives a readForceHierarchy().	
Sequence of Responsibilities: R1. (R1.1)The decisionAid retrieves a DocumentObjectModel (DOM) of the CommonPatternForceShadow (using Parser). (R1.2)The DOM is then read with (R1.3)ForceKeys created. (R1.4)The associated ForceNode instance is created. (R1.5) While reading the hierarchy sub-tree the influence node in the Document will be queried. The associated Influence instance will be initialized and the forceNode will be annotated with its orientation. (R1.6)Once a ForceNode has been initialized it is added to its parent node in the forceHierarchy. R2. As ForceNodes are created the associated ForceKey/Force pair is copied into a :TreeMap(set as the key with values set to a null Force). As <metric> tags are read the :Metric is created, added to the :ForceTypes metric collection, and related influences are added to the metric in question. R3. Once the last of the nodes in the DOM is reached (R3.1) the decisionAid then sends a setPreferences(:TreeMap) to the context. The (R3.2) context cycles through the received TreeMap and instantiates empty preference objects to be associated to each forceKey in a preferences collection. R4. (R4.1)The decisionAid then sequences through its candidates and sending setForceMeasures(:TreeMap). (R4.2)A candidate will then set its forceMeasures to the received TreeMap. (R4.3) Upon return the decisionAid similarly sends the norm candidate a setMeasuredForces(TreeMap) signal with similar effect.	103-NF 301-F 402-F 402-F 602-F 501-F
Postcondition: The decisionAid has a forceHierarchy which is not empty.	
Resulting Event: The DecisionAid continues with the remainder of the Build Context Profile scenario.	
Alternatives: Alt1. (Alt1.1)If a <influence> tag is encountered then the force in the force nodes preference is queried. As well the parent node is queried for its force and is to be assigned to the influence's target. (Alt1.2)The particular node assigns a created Influence object with the calculated force assigned to the influence's origin. Impact is assigned appropriately.	
Nonfunctional Requirements:	
Comments: By keeping track of the number of negative influences we can determine whether a particular criteria is a cost or benefit criteria. This will be used in calculating the appropriate normalizeCost or normalizeBenefit required from a metric. The requirement to explicitly create Influence instances between forces may be revisited in future iterations.	

UML:Sequence Build Force Hierarchy

Scenario: Build Candidates Profile

STD Identifier: Build Candidates Profile <not included in this iteration>	Requirement Reference
Description: The metrics associated with a particular candidate pattern are read from each candidate Pattern Force Shadows.	
External Actors: Pattern Force Shadow	
Precondition: There must be Candidate objects in the DecisionAid candidates collection.	
Triggering Event: The DecisionAid receives a readCandidates() signal.	
Sequence of Responsibilities: R1. (R1.1)The decisionAid gets a Parser and (R1.2) then sequences through the list of candidates signaling to each Candidate to readMetrics(parser, norm). (R1.3)The Candidate then initiates a search of the Pattern Force Shadows for the applicable pattern name and gets the Parser to return a document (DOM) object. R2. (R2.1)The Candidate then reads through the returned document, adding created Metric objects to the forceType in the :Force corresponding to the forceKey within the forceMeasures collection. (R2.2)The candidate then sends the norm the Metric to updateNormMetric(forceKey, metric). In turn the norm compares values received with those contained in the high and low values contained in the corresponding normMetric. R3. Once the last node of the DOM has been read the Candidate returns control to the decisionAid.	501-F 103-NF
Postcondition: The candidates held by the DecisionAid now contain metric values for each forceKey.	
Resulting Event: The DecisionAid sends a recursive signal entitled compare().	
Alternatives:	
Nonfunctional Requirements:	
Comments:	

UML:Sequence Build Candidates Profile

Scenario: Pattern Decision Analysis

STD Identifier: Pattern Decision Analysis	Requirement Reference
Description: To coordinate pairwise comparison of the candidates and subsequently display analyzed results to the pattern user.	201-F
External Actors: Pattern User	
Precondition: The Candidate objects must have metrics available for comparison. The orientation of each forceNode in the forceHierarchy has been previously determined and annotated in the node.	
Triggering Event: The DecisionAid receives a recursive compare() signal.	
Sequence of Responsibilities: R1. (R1.1)The decisionAid invokes a compare() operation in the forceHierarchy passing the candidates, normalized candidate and the context for the input pattern decision problem. (R1.2) Essentially the forceHierarchy conducts a post-order traversal of the Force Nodes in the hierarchy. (R1.3) Each visit is made up of two scenarios, the Compare Candidate Pattern Profiles and Analyze Candidate Pattern Profiles. At the end of these scenarios the candidates will have been updated. R2. As the flow of control is returned to the decisionAid, it will sort the candidates collection based on the number of outranks associated to the Total Satisfaction force of each candidate. R3. The pattern.decisionAidViewer builds an HTML page using the context object and candidates collection. In essence this should be a table with a row for each candidate with the metric values (not including calculated indexes as part of the Comparison scenario). Once the returns have been processed, the program is essentially finished and appropriate deconstruction actions should be taken. <not included in this iteration>	202-F 301-F 205-F 102-NF
Postcondition: Pattern User's browser updated with a HTML page of ranked pattern candidates.	102-NF 101-F
Resulting Event: Control is returned to the Pattern User.	
Alternatives:	
Nonfunctional Requirements:	
Comments: R3. <outside current scope> The context object and candidates collection represent a particular instance of a decision. If made persistent over time a knowledge base can be developed of pattern decision results (trends can be examined in future).	

UML:Sequence Pattern Decision Analysis

UML:Sequence Compare Visit

UML:Sequence Tangible Force Comparison (steps 1.1 and 1.2)

UML:Sequence General Force Comparison (steps 1.1 and 1.2)

UML:Sequence Intangible Force Comparison (steps 1.1 and 1.2)

UML:Sequence Force Comparison (step 1.3)

Scenario: Analyze Comparison Results

STD Identifier: Analyze Comparison Results	Requirement Reference
Description: To pairwise analyze candidates based on the strength of the out-flow and in-flow of the visited force. In so doing the number of candidates a particular candidate outranks is calculated.	201-F
External Actors: None	
Precondition: The force type must have values for the strong and weak out-flows as well as in-flows.	
Triggering Event: The forceNode being visited sends a recursive message, invoking the analyze() method.	
Sequence of Responsibilities: R1. (R1.1)The forceNode initiates the pairwise assessment of outranking by assigning a particular candidate1 another candidate 2 to be assessed against. (R1.2) The two forces under scrutiny then compare the strong out-flow and in-flow values first with weak flows compared in the case of ties. (R1.3) The candidate for whom preference can be established increments its value for number of outranks in the force object. R2. Once all pairwise assessments have been completed the norm candidate is updated with a search for the highest and lowest values for number of outranks.	205-F 602-F 404-F
Postcondition: The norm candidate has a value for hi and low number of outranks.	
Resulting Event: control is returned to the force hierarchy, which continues traversing the hierarchy.	
Alternatives:	
Nonfunctional Requirements:	
Comments: (R1.1) A candidate is not analyzed against it, as well the candidate will handle assigned opposite analysis (that is c1 vs c2 and c2 vs c1). Hence the candidate's collection only needs to be passed through completely once. Basic concept is to have two nested loops the first x =1 to n-1 and the second y = x+1 to n. (R1.2) The comparison of flows is aimed at establishing which candidate has the higher out-flow and lower in-flow.	

UML:Sequence Analyze (Step 2.1)