

**MELD: A PATTERN SUPPORTED METHODOLOGY
FOR VISUALISATION DESIGN**

by

BARRY WILKINS

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
The University of Birmingham
March 2003

Abstract

The quantity and complexity of data that users are being exposed to is increasing. This is especially evident in domains such as the World Wide Web, software engineering, medical systems, etc. There is a need for the user to be supported in the analysis of this data, which may incorporate tasks such as data exploration, navigation, and manipulation. Visualisation is an area that can provide tools and techniques that support these tasks. However, due to the complexity of these domains, designing effective visualisations is difficult.

Recently user interface designers have started to record their successful design experiences in a structured format known as *patterns*. These patterns help designers develop interfaces that use solutions that have proven to work in past designs.

This thesis describes a methodology that uses patterns, in particular visualisation specific patterns, as a mechanism for providing visualisation designers with appropriate design knowledge at each stage in the development of a visualisation. In addition, we propose the use of visualisation heuristics as a way of evaluating alternative visualisation designs in terms of their usability.

Acknowledgements

I would like to thank my supervisor, Bob Hendley, for his support and encouragement throughout my PhD.

I'd like to thank Claire Macklin, Liz Fricker, Shan Cook, and Mark Channer at QinetiQ for their help designing and conducting the cognitive walkthroughs and empirical trials.

I would also like to thank Jim Beadle for his patience, ideas, confirmations and suggestions during our numerous discussions about class design. I'd also like to thank him for looking over the visualisation patterns and confirming that they at least seem sensible. I also greatly appreciate the time and effort he took proofreading my thesis.

I'd like to thank 'The Chief' a.k.a. Rachel Harris for being the other heckler, our across the office email conversations and always being prepared to have a cuppa tea. I also thank Mr Chu-Ming Du for his words of wisdom.

Thanks also to Rachel Harris and Dave Gurnell for proofreading key chapters.

Finally, I'd like to thank my parents and my brother for putting up with me, knowing I'd get there in the end, and being proud of me no matter what.

Jobs a good 'un,

Barry.

Table of Contents

Chapter 1: INTRODUCTION	1
1.1 Visualisation	1
1.2 Problems faced by Visualisation Designers.....	3
1.3 Aims of this Research	4
1.3.1 Methodology	4
1.3.2 Heuristics	5
1.3.3 Development and use of Patterns.....	5
1.3.4 Design Evaluation.....	6
1.3.5 Novel Visualisations	6
1.3.6 Evaluation	7
1.4 Thesis Structure	7
Chapter 2: VISUALISATION OVERVIEW.....	9
2.1 What is Visualisation?	9
2.2 Why do we need it?	9
2.3 Visualisation in Action	11
2.3.1 The Information Mural	11
2.3.2 CyberGeo Maps	12
2.3.3 Narcissus	13
2.3.4 Business Data Visualisation.....	14
2.3.5 LifeLines	14
2.3.6 Cone Tree.....	15
2.3.7 Physiologic Data Visualisation.....	16
2.3.8 Geographic Visualisation.....	17
2.3.9 Summary.....	18
2.4 Why Visualisations work.....	18
2.5 Visual Perception.....	19
2.5.1 Concepts.....	19
2.5.2 Experiments to Determine Pre-Attentive Features	20
2.5.3 Depth Perception.....	24
2.5.4 Gestalt Laws	26
2.5.5 Colour	28
2.5.6 Examples of Perceptual Guides Applied to Visualisation Design.....	30
2.5.7 Limitations on the use of the perceptual system.....	32
2.6 Data.....	33
2.6.1 Concepts.....	33
2.6.2 Type, Range, Reliability	33
2.6.3 Structures	35
2.7 Tasks	38
2.8 Visualisation Design and Visualisation Techniques.....	39
2.8.1 Using three dimensions.....	40
2.8.2 Overview and Detail	41
2.8.3 Focus and Context	43
2.8.4 Large numbers of items, Visual Clutter and Occlusion.....	45
2.8.5 Small Multiples.....	47
2.8.6 Multiple Linked Views	49

2.8.7	Interaction	50
2.8.8	Visualisation techniques summary	53
2.9	Summary	53
Chapter 3: METHODOLOGIES, MODELS, HEURISTICS AND PATTERNS		55
3.1	Overview	55
3.2	Existing Methodologies	55
3.2.1	Software Engineering	56
3.2.2	Human-Computer Interaction	58
3.2.3	Software Engineering meets Human-Computer Interaction.....	60
3.2.4	Generic Design Methodology	62
3.3	Visualisation Reference Models	63
3.4	Automatic Visualisation Generators	66
3.4.1	History and Evolution	66
3.4.2	Problems with Automatic Visualisation Generators.....	67
3.5	Heuristics	69
3.6	Patterns.....	69
3.6.1	Brief history	69
3.6.2	HCI Patterns.....	70
3.6.3	Benefits of Patterns	73
3.6.4	Pattern Approach to Design.....	74
3.7	Summary	77
Chapter 4: A PATTERN SUPPORTED VISUALISATION DESIGN METHODOLOGY ..		79
4.1	Pattern Supported Methodology	79
4.1.1	Overview	79
4.1.2	Information Gathering	80
4.1.3	Analysis	80
4.1.4	Design	89
4.1.5	Implementation	91
4.1.6	Evaluation	92
4.1.7	Iterative Design.....	93
4.2	Heuristics	95
4.3	Visualisation Patterns	100
4.4	Design Methods	104
4.4.1	Design Construction	105
4.4.2	Design Evaluation.....	106
4.5	Summary	112
Chapter 5: APPLICATION OF THE METHODOLOGY		115
5.1	Motivation.....	115
5.2	Command and Control.....	116
5.2.1	Military Context.....	116
5.2.2	Military Task Requirements	117
5.3	Visualisation Designs	121
5.3.1	Motivation.....	121
5.3.2	Resource Checks.....	122
5.3.3	Parallel Coordinates	123

5.3.4	Route Checks	125
5.3.5	Organix	126
5.3.6	Mission Execution	128
5.3.7	Mission Dependencies	129
5.4	Application of the Design Evaluation Methods	131
5.4.1	Motivation.....	131
5.4.2	Design Evaluation Constants	132
5.4.3	Heuristic Evaluation Tool.....	132
5.4.4	Visualisations.....	133
5.4.5	Summary of Predictions.....	146
Chapter 6: EVALUATION, RESULTS, AND ANALYSIS		148
6.1	Objectives and Approach.....	148
6.2	Cognitive Walkthroughs	149
6.2.1	Procedure	151
6.2.2	Technical Analysis.....	151
6.2.3	Usability Analysis.....	153
6.2.4	Summary	157
6.3	Experiment Design	157
6.3.1	Aims and Measures.....	157
6.3.2	Requirements Analysis	158
6.3.3	General Experiment Design.....	160
6.3.4	Simple Task Experiment Design	161
6.3.5	Complex Task Experiment Design	162
6.3.6	Procedure	164
6.4	Results.....	164
6.4.1	Resource Checks.....	165
6.4.2	Parallel Coordinates	170
6.4.3	Route Checks	174
6.4.4	Ranking of Perceived Usability	179
6.5	Conclusion	179
Chapter 7: CONCLUSIONS AND FUTURE WORK		182
7.1	Summary	182
7.2	Conclusions.....	184
7.3	Future Work.....	187
7.3.1	Methodology	187
7.3.2	Patterns.....	188
7.3.3	Design Evaluation Techniques	188
7.3.4	Visualisation	190
Appendix A: HEURISTICS.....		192
Appendix B: HEURISTIC CLASSIFICATION.....		207
B.1	Heuristic effect on Usability Factors	207
B.2	Rationale	209

Appendix C: VISUALISATION PATTERNS	221
Appendix D: ANALYSIS FACTOR QUESTIONS	254
Appendix E: MILITARY TASK SCENARIOS	258
E.1 Mission Plan Problems	258
E.2 Task Scenarios	259
E.2.1 Scenario 1 (pre-mission assessment)	259
E.2.2 Scenario 2	260
E.2.3 Scenario 3	261
E.2.4 Scenario 4	262
E.2.5 Scenario 5	262
E.2.6 Scenario 6 (post-mission analysis).....	265
E.2.7 Scenario 7 (campaign analysis)	265
List of References	267

List of Figures

Figure 1.1: The power of visual representations.....	2
Figure 2.1: Information Mural.	12
Figure 2.2: CyberGeo Map.	12
Figure 2.3: CyberGeo Map example of website document changes.	13
Figure 2.4: Narcissus.	13
Figure 2.5: Business data visualisation.....	14
Figure 2.6: LifeLines.	15
Figure 2.7: Cone Tree.	16
Figure 2.8: Traditional physiologic data display.	16
Figure 2.9: Real-time physiologic data visualisation.....	17
Figure 2.10: Geographic visualisation.	17
Figure 2.11: Basic model of human visual processing system.	20
Figure 2.12: Response times for pre-attentive and non-pre-attentive targets.	21
Figure 2.13: Target detection using hue.	21
Figure 2.14: Target detection using shape.	21
Figure 2.15: Boundary detection using hue.	22
Figure 2.16: Conjunction search.	22
Figure 2.17: Feature hierarchy.....	23
Figure 2.18: Model of perspective.	24
Figure 2.19: Perspective cues.	25
Figure 2.20: Shading depth cue.	25
Figure 2.21: Focus depth cue.	26
Figure 2.22: Proximity: (a) row dominant, (b) column dominant.	27
Figure 2.23: Continuity.....	27
Figure 2.24: Common fate.....	28
Figure 2.25: RGB Colour Cube.	28
Figure 2.26: Colour Wheel.	29
Figure 2.27: HSB Cone.....	29
Figure 2.28: A set of symbols for a military command and control display.	30
Figure 2.29: Salmon tracking simulation.....	32
Figure 2.30: Mackinlay's (1986) ranking of visual features to data type.....	34
Figure 2.31: MRI data superimposed onto a model of a human head.	36
Figure 2.32: TreeMap visualisation of a file system.	36
Figure 2.33: Network visualisation – Arc Map.	37
Figure 2.34: Example of occlusion in a 3D data space.....	40
Figure 2.35: Reference context. Anchor line and shadow techniques.....	41
Figure 2.36: Overview and detail.	42
Figure 2.37: Hyperbolic browser distorted space focus and context technique.	43
Figure 2.38: FISHEYE view.....	44
Figure 2.39: Fisheye lens distortion.....	44
Figure 2.40: Table Lens collapsed space focus and context technique.	45
Figure 2.41: Visual clutter.	46
Figure 2.42: Viewing detailed information using a temporary 'pop up' window.	46
Figure 2.43: Small multiples in Web Analysis Visualisation Spreadsheet.....	48

Figure 2.44: Small multiples in VisDB.	48
Figure 2.45: Snap-Together Visualisation.	50
Figure 2.46: FilmFinder.	51
Figure 2.47: Bounding box technique used in Microsoft Windows Explorer™.	52
Figure 2.48: Selective Dynamic Manipulation (SDM).	52
Figure 3.1: Software Development Life Cycle.	56
Figure 3.2: Spiral Model.	57
Figure 3.3: Rapid Prototyping.	57
Figure 3.4: Generic design methodology.	62
Figure 3.5: Basic visualisation process.	63
Figure 3.6: Visualisation process.	64
Figure 3.7: Integrated visualisation model.	64
Figure 3.8: Extended visualisation process.	65
Figure 3.9: Mackinlay’s ranking of visual features to data type.	66
Figure 3.10: The Pattern Supported Approach Framework.	75
Figure 4.1: Overview of methodology.	80
Figure 4.2: Information gathering.	80
Figure 4.3: Analysis stage.	81
Figure 4.4: Design stage.	90
Figure 4.5: Implementation stage.	92
Figure 4.6: Evaluation process.	92
Figure 4.7: Pattern Supported Visualisation Methodology.	94
Figure 4.8: Structure pattern language.	103
Figure 4.9: Interaction pattern language.	104
Figure 4.10: Stages of the methodology.	112
Figure 4.11: Stages of the methodology including supporting patterns.	113
Figure 4.12: Methodology including usability engineering life cycle elements.	113
Figure 4.13: The complete pattern supported visualisation design methodology.	114
Figure 5.1: Situation awareness visualisation.	117
Figure 5.2: Resource Checks visualisation.	123
Figure 5.3: Parallel Coordinates visualisation.	124
Figure 5.4: Route Checks visualisation.	126
Figure 5.5: Organix.	128
Figure 5.6: Mission Execution visualisation.	129
Figure 5.7: Mission Dependencies visualisation.	130
Figure 5.8: Heuristic Evaluation tool.	133
Figure 5.9: Resource Checks – wow-vis version.	134
Figure 5.10: Resource Checks – just-a-vis version.	134
Figure 5.11: Power level colour assignments.	135
Figure 5.12: Base resource layout comparison.	136
Figure 5.13: Parallel Coordinates – wow-vis version.	138
Figure 5.14: Parallel Coordinates – just-a-vis version.	138
Figure 5.15: Route Checks – wow-vis version.	142
Figure 5.16: Route Checks – just-a-vis version.	142
Figure 5.17: Legends used in the wow-vis Route Checks visualisation.	143

Figure 6.1: Comparison of intuitiveness to preference.....	157
Figure 6.2: Presentation blocks for simple tasks.	161
Figure 6.3: Presentation blocks for complex tasks.	163
Figure 6.4: Similarity dimension counterbalancing.....	163
Figure 6.5: Resource Checks – simple tasks – reaction time (correct answers only).....	166
Figure 6.6: Resource Checks – simple tasks – reaction time vs. accuracy.....	167
Figure 6.7: Resource Checks – simple tasks – SUMI scores.....	168
Figure 6.9: Resource Checks – complex tasks – SUMI scores.....	169
Figure 6.10: Parallel Coordinates – simple tasks – reaction time.....	171
Figure 6.11: Parallel Coordinates – simple tasks – reaction time vs. accuracy.	172
Figure 6.12: Parallel Coordinates – simple tasks – SUMI scores.....	172
Figure 6.13: Parallel Coordinates – complex tasks – reaction time.....	173
Figure 6.14: Parallel Coordinates – complex tasks – SUMI scores.....	174
Figure 6.15: Route Checks – simple tasks – reaction time.	176
Figure 6.16: Route Checks – simple tasks – reaction time vs. accuracy.	176
Figure 6.17: Route Checks – simple tasks – SUMI scores.	177
Figure 6.18: Route Checks – complex tasks – reaction time.....	178
Figure 6.19: Route Checks – complex tasks – SUMI scores.....	178
Figure C.1: Structure pattern language.	222
Figure C.2: Interaction pattern language.	223

List of Tables

Table 2.1: Example dataset.....	33
Table 2.2: Schema.....	34
Table 2.3: Upper limit of values that can be encoded by various visual features.	35
Table 3.1: Usability Engineering Life Cycle.....	61
Table 3.2: How the generic methodology relates to the usability engineering life cycle.....	62
Table 3.3: Example usability heuristics.....	69
Table 3.4: Pattern comparison.	73
Table 3.5: Borchers' application of patterns to the usability engineering life cycle.	76
Table 4.1: The relationship between patterns and elements of the usability engineering life cycle in the design stage.	91
Table 4.2: Heuristics and sources.	96
Table 4.3: Heuristic format.....	97
Table 4.4: Heuristics matrix.....	98
Table 4.5: Heuristic effect on usability factors.....	99
Table 4.6: Rationale for heuristic 9.....	100
Table 4.7: Overview and Detail structure pattern.....	102
Table 4.8: Example data factor question and solution.....	106
Table 4.9: Comparison between Mackinlay (1986) and Salisbury (2001).	107
Table 4.10: Heuristic score values and descriptions.....	108
Table 4.11: Heuristic score matrix.....	109
Table 4.12: Example usability factor weight assignments.	109
Table 4.13: Example of n-to-1 mapping.....	110
Table 4.14: Dimensional score mapping assignments.....	111
Table 5.1: Typical military scenario used for task analysis.....	118
Table 5.2: Example task scenario.	121
Table 5.3: Usability factor weight assignments for command and control.	132
Table 5.4: Mapping evaluation for visual features used in Resource Checks visualisation...	136
Table 5.5: Heuristic score matrix for Resource Checks visualisation.	136
Table 5.6: Resource Checks – just-a-vis usability scores.....	137
Table 5.7: Resource Checks – wow-vis usability scores.....	137
Table 5.8: Resource Checks visualisation ratings.	137
Table 5.9: Heuristic score matrix for Parallel Coordinates visualisation.	139
Table 5.10: Parallel Coordinates – just-a-vis usability scores.....	139
Table 5.11: Parallel Coordinates – wow-vis usability scores.....	139
Table 5.12: Parallel Coordinates visualisation ratings.....	140
Table 5.13: Colour assignments used in Route Checks visualisations.....	141
Table 5.14: Mapping evaluation for visual features used in Resource Checks visualisation.	144
Table 5.15: Heuristic score matrix for Route Checks visualisation.....	144
Table 5.16: Route Checks – just-a-vis usability scores.....	144
Table 5.17: Route Checks – wow-vis usability scores.....	145
Table 5.18: Route Checks visualisation ratings.....	145
Table 5.19: Summary of ratings using the mapping evaluation.	146

Table 5.20: Summary of visualisation ratings using the heuristic evaluation.	146
Table 5.21: Summary of visualisation rankings using each design evaluation technique.....	147
Table 6.1: Tasks used in cognitive walkthroughs.....	150
Table 6.2: Subjective usability of visualisations.	153
Table 6.3: Average intuitiveness ratings for each visualisation.	154
Table 6.4: Visualisations ranked by perceived intuitiveness.	154
Table 6.5: Average preference ratings for each visualisation.....	155
Table 6.6: Visualisations ranked by preference.....	156
Table 6.7: Task categories supported by visualisations.....	159
Table 6.8: Task category tasks.....	160
Table 6.9: Latin Square Design counterbalance for five factors (A-E).	161
Table 6.10: Counterbalanced presentation order for the complex task experiment.	163
Table 6.11: Resource Checks usability ratings using heuristic evaluation.	165
Table 6.12: Resource Checks observed results for simple and complex tasks.....	165
Table 6.13: Parallel Coordinates usability ratings using heuristic evaluation.	170
Table 6.14: Parallel Coordinates observed results for simple and complex tasks.....	170
Table 6.15: Route Checks usability ratings using heuristic evaluation.	174
Table 6.16: Route Checks observed results for simple and complex tasks.	175
Table 6.17: Ranking of perceived usability for simple tasks.....	179
Table 6.18: Ranking of perceived usability for complex tasks.....	179
Table B.1: Heuristic effect on usability factors.	208

Chapter 1: INTRODUCTION

In this thesis we explore ways in which the visualisation designer can be guided to develop visualisations that use well-established techniques. To accomplish this we have reviewed three main areas, visualisation, human computer interaction, and software engineering. This review has focused on the methodologies and techniques used in each of these disciplines, the mechanisms they use to capture and deliver design knowledge, and the methods by which designs can be evaluated. This has led to the development of a methodology that focuses on the factors relevant to visualisation design and that is supported at each stage by appropriate design knowledge.

This chapter briefly introduces the concept of visualisation and why we need it, the problems that visualisation designers are faced with, the aims of this research with respect to those problems, and finally an overview of the structure of the remainder of the thesis.

1.1 Visualisation

Visualisation could be defined as the visual representation of data. This includes traditional representations such as pie charts, bar charts, and scatter plots. However, through the use of computers visualisation provides something more. Software based visualisations are capable of supporting large datasets and, via interaction, give the user the freedom to explore the data, test theories, discover patterns, monitor dynamic events and perform a wide variety of tasks that are simply not possible with traditional static displays.

Visualisations can be created for any domain and, as such, can take many forms. Scientific visualisations are often based on the physical model from which the data is gathered. For example, global ozone-levels can be presented using a model of the Earth, medical imaging data can be displayed using a map of the human body, and so on. In contrast to this, information visualisation tends to use more abstract data such as document collections, software modification logs, file systems, etc., and so has no physical model upon which to base the display. Chart-based representations may still be applicable but often more novel visualisation designs are required to support this abstract data.

There remains the question of why we need visualisation. To answer this we need to ask why we use chart-based displays rather than lists of items. Consider the following example.

Imagine trying to find the highest value in a list of ten two-digit numbers by visual inspection. This is a fairly straightforward serial search task and could be completed quite easily. However, what if the number of items increased from ten to one hundred, or one thousand, or the size of the numbers increased from two to four digits? To attempt this by reading each

1.1 Visualisation

value and comparing it to the highest value held in your short-term memory would be a tedious, time consuming, and in all likelihood, error prone task. Performing the same task using a simple visualisation such as a bar chart or scatter plot is much easier.

Even if we only have a small number of items, an appropriate visual representation still has a number of advantages. Looking at the scatter plots in figure 1.1 it can be seen that there are linear relationships, quadratic relationships, outliers, and so on, information that is almost impossible to detect using the tables alone. In this case the scatter plots act as a tool that can be used to gain insight into the data.

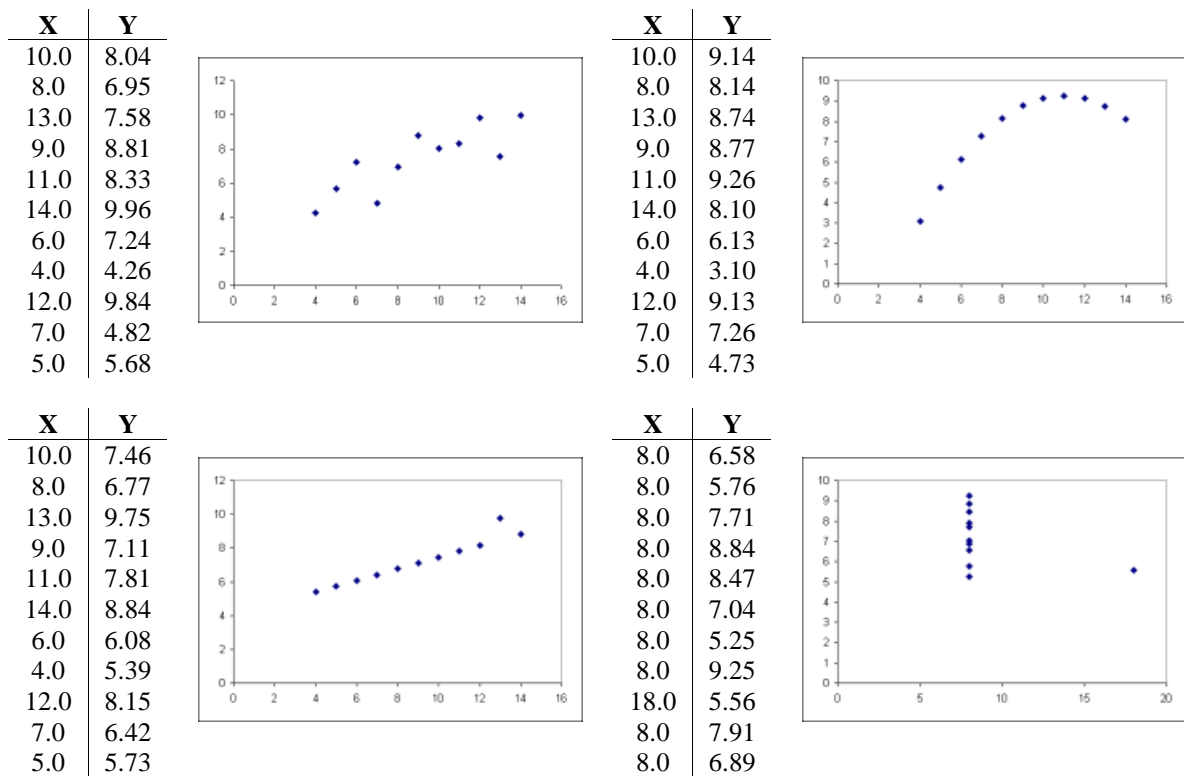


Figure 1.1: The power of visual representations.

(Modified from Tufte (1983))

Card et al. (1999) have identified six ways in which visualisation helps to achieve this insight. These include increasing the memory and processing resources available to the user, reducing the search for information, using visual representations to enhance the detection of patterns, enabling perceptual inference operations, perceptual monitoring, and encoding information in a non-static medium. Essentially, visualisation tries to utilise the strengths of the human visual system and at the same time, overcome the limitations of human memory to reduce overall cognitive load.

Visualisation supports the ever-increasing amount of data we encounter each day; it helps us perform more complex tasks, to make sense of the data, and to find relevant information and

reach conclusions in less time and with greater accuracy. In essence it shifts much of the workload from the cognitive to the perceptual system thus enabling the viewer to more easily gain an insight into data.

1.2 Problems faced by Visualisation Designers

From the discussion above it would appear that visualisation is an ideal mechanism for supporting the user in the analysis of data, but there are problems. The quantity and complexity of the data that users have to deal with is increasing. In addition, the tasks the users must perform are also becoming more involved. It may still be possible to use traditional chart-based displays but it is unlikely that this will be as effective as using a specifically designed visualisation.

Novel visualisations need to be designed to support these more demanding requirements. Researchers such as Mackinlay (1986), Casner (1991), Healey et al. (1998), Chuah (2000), Salisbury (2001), and others, have developed visualisation systems that automatically produce visual representations based on the type of data to be displayed and the tasks to be performed. However, these systems are only capable of producing chart-based displays, which are often ineffective when used with large datasets or for complex tasks. Human designers on the other hand can be much more innovative but need to be guided in their designs.

The data, the task, the domain and the user are just some of the factors that the designer needs to consider. Visualisation also uses techniques from disciplines such as computer graphics, human computer interaction, perceptual psychology, data mining and software development. The complexity of the application domain and the breadth of knowledge that the field of visualisation requires, makes designing effective visualisations difficult.

Ideally, experts from each of the disciplines mentioned would be brought together as part of a visualisation design team. However, this can be expensive in terms of time and money, and is often not possible. One way to overcome this problem is to use guidelines, such as the eight golden rules for interface design proposed by Shneiderman (1998). This has begun to happen with visualisation design. However, as researchers such as Welie et al. (2000) have noted, guidelines are often difficult to select, interpret and apply, they may be too simplistic, and they may even contradict each other.

Recently both software developers and human computer interaction specialists have used the idea of *patterns* to capture and communicate design knowledge. Put simply, patterns are a structured format for presenting proven solutions to recurring design problems. Patterns have a number of benefits over guidelines, but they are difficult to write, and at present, there are no specific visualisation design patterns.

1.3 Aims of this Research

Early software development tended to be intuitive, relying on the software engineer's past experience and imagination. This often resulted in applications that were less than satisfactory. After the introduction of a formal software development process, applications were produced more rapidly, were more robust and reliable, consisted of reusable components, and improved usability.

Reference models for visualisation have been proposed. These models describe a series of stages that convert data into a visual form. However, although these models are useful, they focus more on the visualisation process and not on visualisation design. Visualisation designers are still left with the problems of what techniques to use, which visual structures to employ, the most suitable interaction mechanisms, and so on.

Therefore visualisation designers need a methodology that will incorporate the design knowledge and techniques from various disciplines, overcome the deficiencies of guidelines, and focus the designer on developing a system that is effective, usable, and capable of supporting the ever increasing demands of today's users.

1.3 Aims of this Research

The primary aim of this research is to develop a methodology that can be used by human designers to design novel visualisations that are both effective and usable. This section describes this goal and other related goals in more detail.

1.3.1 Methodology

There have been few methodologies proposed for visualisation design that can be used by human designers. Those that have tend to be lacking in certain design aspects such as data gathering, design evaluation, identifying applicable visualisation techniques, and so on. We attempt to identify the factors relevant to visualisation design, formalise the design process, and support each stage of the design process with well-established techniques and experience-based design knowledge.

In short we propose a formal methodology that covers the main aspects of visualisation design from the gathering of raw data to the evaluation of visualisation designs. This methodology represents a synthesis and extension of previous work, incorporating and combining experience-based heuristics, patterns, software development, human computer interaction techniques and principles from perceptual psychology into a coherent whole that a human designer can use to produce visualisations that are effective and usable.

1.3.2 Heuristics

As well as describing the stages that a designer should follow it is useful for each stage in a methodology to be augmented by related design knowledge. In this way the designer has access to knowledge that can guide them to effective designs. Heuristics, such as those proposed by Nielsen and Molich (1990) and Nielsen (1994), usually consist of one or two sentences describing features that are desirable in a system. As such, heuristics are one way of providing designers with this knowledge.

A number of visualisation researchers have proposed heuristics for visualisation design. Each researcher tends to use their own format and a number of researchers have proposed the same heuristic but under a different title. We propose a formal structure for presenting visualisation heuristics and the compilation of a single catalogue of heuristics from the various disparate sources. This has a number of benefits including making information easier to find, reducing the chances that information will be missed, helping to generate new ideas, helping to identify gaps in knowledge, and providing a useful place where other heuristics can be added in the future. It is the starting point for the development of a single comprehensive library of heuristics that can be added to, updated, and maintained by future researchers.

We propose classifying the heuristics by their effect on various usability factors such as user performance, rate of errors, etc. This allows the designer to focus on the heuristics that are most relevant to their usability design goals.

1.3.3 Development and use of Patterns

Patterns are another technique for capturing and delivering design knowledge, and have been used in disciplines such as architecture, software development, and user interface design. In contrast to heuristics, patterns also record additional meta-information so the designer knows when to apply the heuristic and why it works, as well as examples of how it has been implemented in existing systems. This reduces the chances that the design knowledge will be used inappropriately.

We propose the development of visualisation specific patterns that can be used to produce designs that incorporate well-established techniques. We also feel that patterns can be used to support all stages of the proposed methodology both as a means to producing good designs and as a mechanism for communicating design knowledge between design team members from different disciplines.

1.3.4 Design Evaluation

Visualisation designers are often faced with the problem of how to decide which of several alternative but equivalent designs to develop further. One way of reaching a decision is to analyse the visualisation in terms of its perceptual effectiveness. With respect to low-level visual processing this is an accurate technique, however, it fails to take into account the usability of the visualisation. To try and overcome this problem we aim to use classified visualisation heuristics as part of a design evaluation technique.

Using a single evaluation technique it is often not possible to cover all factors relevant to a design. Therefore we aim to try and gather evidence that supports the use of three different evaluation techniques, each of which is geared towards a different aspect of visualisation design, as a means of corroborating visualisation designs. The three methods we use are as follows:

- *Mapping evaluation*: Mackinlay (1986) and Salisbury (2001) have rated various perceptual encoding techniques used in visualisation designs. We propose that these ratings can be used as a simple measure of the perceptual effectiveness of a visualisation.
- *Visualisation heuristic evaluation*: The classification of the visualisation heuristics allows them to be used as an evaluation tool. Briefly, the designer weights several usability factors based on their relevance to the domain and the designers usability design goals. For each design the designer assigns a score to each heuristic based on how well the designer feels that heuristic has been implemented. The weighted sum of each heuristic's effect on the usability factors is then combined with the scores to produce a final visualisation rating. The ratings for each visualisation design can then be compared. This technique is described in detail in chapters four and five. It should be noted that although we have referred to this as an heuristic evaluation, it does not strictly meet Nielsen and Molich's (1990) definition.
- *Metric evaluation*: Brath (1997a, 1997b, 1999) has proposed a number of visualisation metrics that can be used to compare visualisation designs. However the metrics have not been used formally in other research. We adopt these metrics as a means to corroborate effective visualisation designs and as way of determining how applicable they are to visualisation design evaluation.

1.3.5 Novel Visualisations

We have chosen to develop novel visualisations for military command and control. This is primarily due to our connections with QinetiQ, previously known as the Defence Evaluation Research Agency (DERA). With respect to QinetiQ's requirements, Macklin and Dudfield (2001) have identified the need to develop systems that support command teams in situation

assessment and decision-making. This research will try to find evidence to support the hypothesis that visualisations are an appropriate tool that can fulfil this need.

It should be noted that although a specific domain has been chosen, the proposed methodology can be applied to any domain and it is hoped that some of the visualisations developed are generic enough to apply to multiple domains. The lessons learnt while developing new visualisations may provide valuable information to any future work.

1.3.6 Evaluation

We aim to perform an empirical evaluation of each of the visualisations implemented in order to gather evidence that supports their use as effective decision-making tools in military command and control.

The empirical evaluation will also attempt to measure the usability of the visualisations. The data collected will then be compared with the predictions made by each of the design evaluation techniques. Since the heuristics form the basis for many of the patterns, and the patterns are an integral part of the methodology, we are particularly interested in the comparison between observed results and the visualisation heuristic evaluation.

1.4 Thesis Structure

Chapter two begins with a more detailed discussion of what visualisation is and why we need it. There then follows a review of various visualisation designs to illustrate the range of domains and tasks that visualisations can support. From this review we discuss why visualisation works, concentrating in particular on human visual perception¹ and how perceptual experiments have led to the development of specific principles that can be used to design perceptually effective visualisations. The influence of both the data and tasks on visualisation design is also discussed. Visualisation design issues are then introduced together with the development of techniques commonly used in visualisations to overcome recurring design problems. A summary of the chapter and the reasons why a methodology for visualisation design is required are then presented.

Chapter three focuses on the relationship between software engineering, human computer interaction, and visualisation. In particular, the integration of various methodologies and techniques from each discipline is described, as is the use of patterns as a means of capturing design knowledge and facilitating cross-discipline communication. Together with chapter two this provides the setting for the visualisation design methodology presented in chapter four.

¹ In the remainder of this thesis, unless otherwise stated, perception should be read as visual perception.

In chapter four, each stage of the methodology is described together with a rationale for its inclusion. Specific components relevant to the design stage are then described in detail. This includes details about the heuristics that have been collected, the development of visualisation patterns, and the proposed visualisation design evaluation methods.

Chapter five describes the development and implementation of a set of visualisations to be used later in an empirical evaluation. After a brief introduction to the command and control domain the process of determining a suitable set of military tasks is described. A number of visualisations are then presented together with the predictions made by the proposed design evaluation techniques.

An account of the motivations behind and process involved in the evaluation of the visualisations described in chapter five is given in chapter six. Also included is an analysis of the data collected.

Chapter seven discusses whether or not the contributions outlined in the introduction have been achieved, the successful and unsuccessful aspects of the research and finally how the work might be adapted or extended in the future.

A number of appendices are also included which provide details about the heuristics, visualisation patterns, and military task scenarios.

Chapter 2: VISUALISATION OVERVIEW

This chapter starts with a discussion of what visualisation is and why it is necessary. There then follows a review of visualisation designs. This is used to try to establish why visualisation works and the types of data and tasks that visualisations support. The problems that visualisation designers face and the techniques they use to overcome them are also included. A series of open questions identified during the review are then presented at the end of the chapter.

2.1 *What is Visualisation?*

Visualisation can be defined as the use of visual representations to gain an understanding about data. Hence, by using these visual representations, the process of acquiring knowledge, and consequently the use of that knowledge, is enhanced. This definition takes into account representations such as tables, pie charts, bar charts, scatter plots, etc., but also other, more novel representations.

Traditionally, visual representations of data such as tables, bar charts, scatter plots, etc., have been drawn on paper. However, visualisations are more commonly associated with computers, which are more convenient, more flexible, can process large quantities of data quickly, and perhaps most significant of all, allow the data to be interacted with. This has led Card et al. (1999) to define visualisation as:

“The use of computer-supported, interactive, visual representations of data to amplify cognition.”

Using this definition two main forms of visualisation can be identified based on the data source. In *scientific visualisation* the data is usually gathered from some physical source; therefore the visualisation already has a model upon which to base the representation. For example, sensor data gathered during a magnetic resonance imaging (MRI) scan can be overlaid onto a model of the organ scanned, ozone levels can be shown using a model of the world, and so on. In contrast to this the data used in *information visualisation* tends to be more abstract with no corresponding physical model upon which to base the display. Examples of abstract data include document collections, financial data, company records, software systems, etc. Since there is no physical model underlying the data, information visualisations can be much more difficult to design. In this case metaphors are often used to represent data concepts.

2.2 *Why do we need it?*

The amount of data being recorded is increasing each day, as is its complexity. To gain a competitive advantage in the marketplace or confirm some hypothesis, users need to analyse

2.2 Why do we need it?

and make sense of this data. Due to the quantity and complexity of the data and the fact that the majority of data is stored on computer, using pen and paper techniques is no longer viable. However, even software-based versions of tables, bar charts, scatter plots, etc., have difficulty supporting the huge quantities of data that users now have to cope with.

The quantity of data is not the only issue. The types of data source and the relationships between different data sources are becoming more complex. For example, magnetic resonance imaging data collected for a patient may need to be linked to the patient's medical records, which may include references to other scans and possibly to other patients with similar conditions, this may then lead to demographic information for all patients in the same region, and so on. These complex interrelationships can be difficult for users to follow.

Users are also becoming more demanding. They no longer simply want to view the data, they want to explore and manipulate it. They want to be able to try out 'what if' scenarios, hide data that is not relevant, follow paths through the data, view the same data in different ways, and in general perform fast and accurate analyses.

It could be argued that several of these issues can be addressed using algorithms or machine learning techniques such as decision tree learning, instance based learning, rule based learning, inductive learning, analytical learning, reinforcement learning, etc. In some cases this is true. If the task to be supported is well defined then it is nearly always better to use a mechanical approach. For example, it would more effective to write an algorithm that finds the largest file on a file system, than to have a user interpret a visualisation. Even more complex tasks such as pattern detection can be achieved using machine learning techniques.

However machine learning is less appropriate for other types of task. For example, when making a decision an expert human user combines reasoning with past experience and other non-quantifiable factors. This human model is very difficult to capture and reproduce, even when multiple machine learning techniques are used in combination. In contrast, visualisation simply presents the data relevant to the task, together with mechanisms for exploring the data, and lets the human user do their job, i.e. interpret the data. Data interpretation is something that humans are good at, and this ability can be enhanced by the use of a well designed visualisation. Of course, once the process a human user goes through to solve a task can be clearly defined, it may then be possible to automate it.

Of particular relevance is the fact that most of the machine learning techniques are only capable of solving one or two specific tasks whereas a visualisation can be constructed to solve multiple diverse tasks. For example, a simple scatter plot allows a user to see clusters, outliers, minimum and maximum values, and so on. To achieve the same level of task support with machine learning techniques would require combining, and possibly training, several

different techniques. In this case a very simple visualisation can achieve the same results as a very complex set of machine learning techniques.

In short, we need visualisations so that users can analyse and make sense of increasingly large quantities of complex data. This is something that cannot be satisfactorily achieved using traditional static chart-based representations or machine learning techniques.

2.3 Visualisation in Action

This section presents a review of different visualisation designs. This review is not intended to be exhaustive; it is simply a way of clarifying what a visualisation is, reinforcing the reasons why visualisations are needed, and highlighting the domains that visualisation has been applied to. In addition, an analysis of these designs will help to determine why visualisations work, the elements common to all visualisations, and the techniques that visualisation designers use to overcome common visualisation design problems. Additional designs will also be presented throughout the remainder of this chapter.

2.3.1 The Information Mural

The Information Mural developed by Jerding and Stasko (1997) represents large information spaces in small graphical spaces by mapping elements in the information space to pixels in the graphical space. As each information element is mapped to a pixel the greyscale or colour intensity of that pixel is altered to reflect the type of information element and the number of ‘hits’ that pixel has received.

An example of the Information Mural technique applied to a large text document is shown in figure 2.1. The mural is the coloured bar at the left hand side of the figure. Each colour represents a different section of the document with the intensity of the colour reflecting the density of the text (i.e. number of characters) at that point in the document. The small red rectangle in the upper third of the mural indicates the section of the document currently viewed in the main display.

One of the main advantages of the Information Mural technique is that it allows the user to browse the entire information space and focus quickly on items of interest. It also makes maximum use of the limited display space available. The Information Mural has been used to visualise software execution, sun spot activity data, population density distribution, river flow data, geographic information, and text documents.



Figure 2.1: Information Mural.

2.3.2 CyberGeo Maps

In order to view changes in the structure and content of websites over time, Holmquist et al. (1998) developed the CyberGeo Map visualisation. CyberGeo Maps represent each document in a website as a dot, with the index document at the centre of the map. The size of the dot is used to represent the document size and the greyscale value of the dot indicates the age of the document, with darker dots representing older documents. The dots are then placed at a distance from the centre proportional to the associated document’s depth in the file structure and distributed radially based on the name of the directory the document is in and the document file name. This is done in such a way that documents in the same sub-directory are placed in approximately the same area. Finally each dot is given a random amount of ‘jitter’ to reduce overlap. An example CyberGeo Map is shown in figure 2.2.

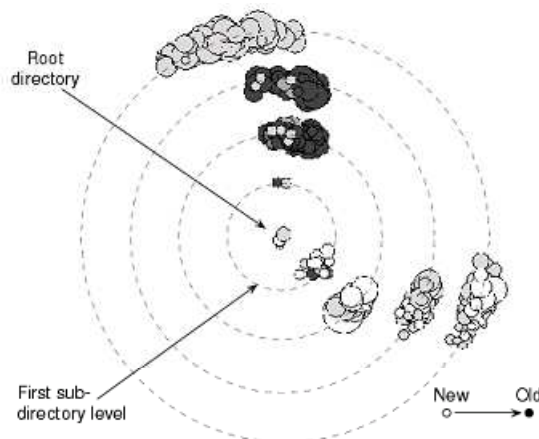


Figure 2.2: CyberGeo Map.

Scanning the documents of a website regularly over a period of time and representing each scanned set as a CyberGeo Map provides insight into how that website has changed over time. For example in figure 2.3 it can be seen that several documents have been added to the root directory and a large document has appeared in a 2nd level sub-directory. Also, due to changes in the positions of the dots, it is evident that several documents or directories have been moved or renamed in both the 2nd and 3rd level sub-directories, and in addition the 3rd level sub-directory has had one large document removed and several smaller documents moved or added, again with different names or in a different sub-directory.

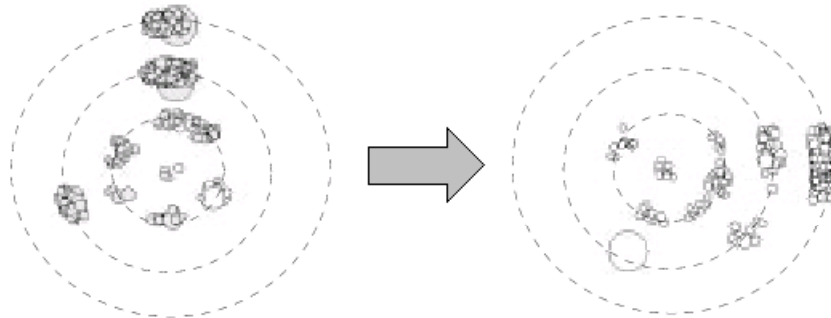


Figure 2.3: CyberGeo Map example of website document changes.

2.3.3 *Narcissus*

Narcissus, developed by Hendley et al. (1995), is designed to help users understand and navigate complex structures such as websites and program code. Spheres are used to represent objects such as web pages or software classes and links between them indicate relationships. Initially the spheres are placed randomly in 3D space. A novel layout algorithm is then applied which causes the spheres to attract and repel each other based on the semantic similarity of their associated data object. After a number of iterations equilibrium is reached and the structure of the system is revealed. For example in figure 2.4, which is a visualisation of the classes in a software system, it is apparent that there are two main classes from which most of the subclasses inherit multiply and only a few singly.



Figure 2.4: Narcissus.

Narcissus allows the user to move through the virtual space, turn links on and off, and select individual objects or classes of objects. Hendley et al. (1995) claim that during informal tests these facilities, together with the visual representations used, are useful. However, an empirical evaluation of Narcissus has yet to be completed.

2.3.4 Business Data Visualisation

Over many years Brath (1999) has developed dozens of three-dimensional visualisations for the business domain. As can be seen in figure 2.5 many of these visualisations are derived from chart-based representations. However, as well as using three dimensions, interaction and animation, Brath also extends these more traditional representations by linking them together and combining them in a single display as shown in figure 2.5.d. This gives the user access to more information in a smaller space and allows them to identify patterns, trends, relationships, etc., more easily.

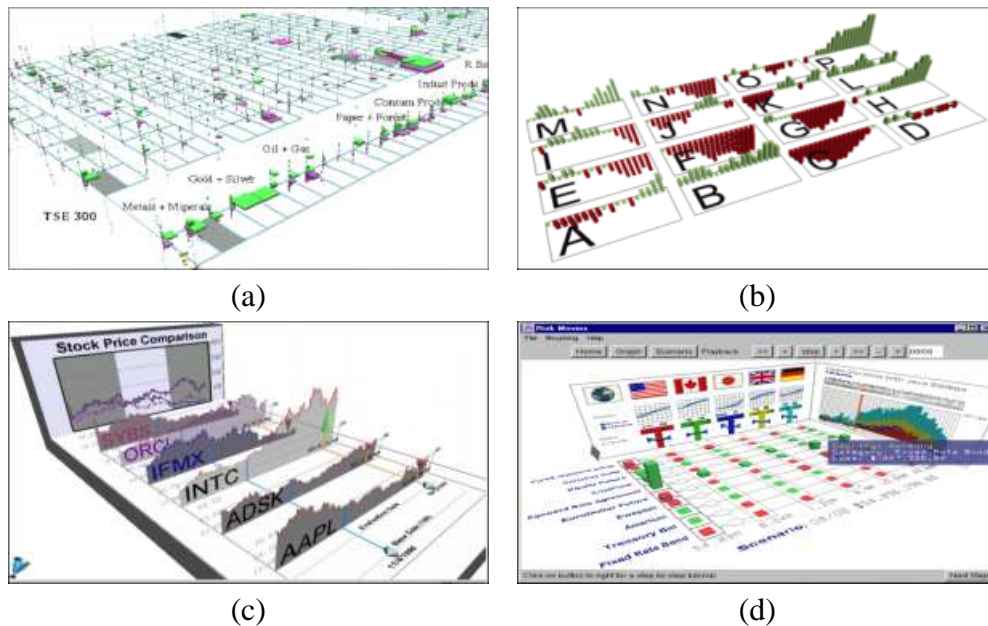


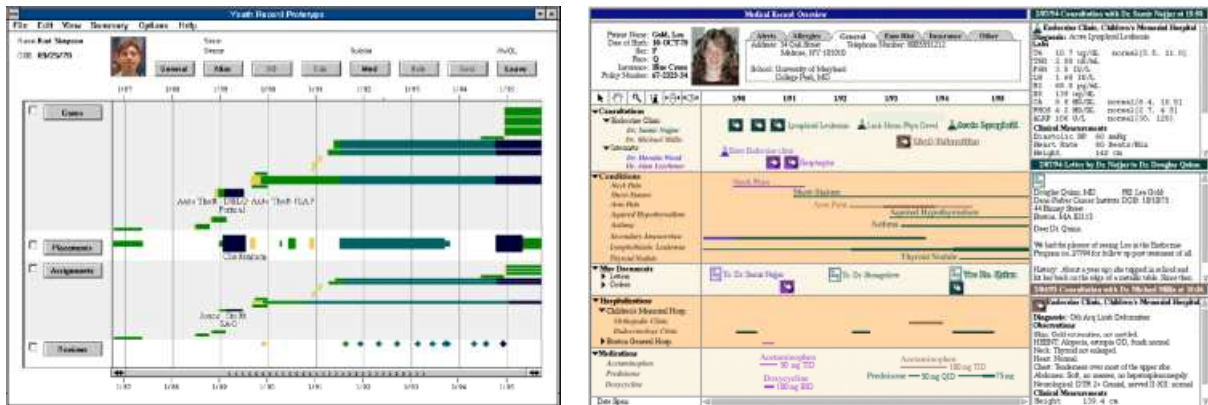
Figure 2.5: Business data visualisation.

Brath (1999) states that these visualisations have been used successfully by a number of commercial organisations.

2.3.5 LifeLines

Plaisant et al. (1996) found that although the amount of personal data e.g. medical records, legal case histories, etc., being recorded was increasing, adequate tools for the presentation and exploration of that data did not exist. In the LifeLines visualisation personal histories are represented as horizontal lines placed at appropriate points along a time axis. The colour and thickness of these lines are used to represent various aspects of the data such as the severity of a crime or the doctor a patient has seen. In addition icons are used to represent discrete events such as a consultation. Examples of how LifeLines has been used to visualise young offender records and medical records are shown in figure 2.6.

2.3 Visualisation in Action



(a) Young offender record.

(b) Medical record with detailed view.

Figure 2.6: LifeLines.

Initially the LifeLines display shows an overview of an entire personal history record, users can then filter out unwanted information, zoom in on items of interest, and click on the icons to access detailed information. Thus, the user can spot trends and anomalies using the overview and can confirm information as required.

2.3.6 Cone Tree

As the use of computers increases so does the number of files that users have to organise and maintain. This is particularly difficult for network administrators who have to deal with literally thousands of files. Traditional viewers, such as Windows Explorer™, show only a small part of the file system at any one time, making it difficult to determine the overall structure or locate items of interest. The Cone Tree visualisation developed by Robertson et al. (1993) tries to overcome these problems by using three dimensions. However its application is not limited to file systems; the Cone Tree can be used with any data that has a parent-child relationship, i.e. a tree structure.

A Cone Tree is built such that the children of a particular item are distributed evenly around the circumference of the base of a cone with the parent item at the top of the cone. Each child item is then treated as a parent with its own cone and the process is repeated. Unfortunately as can be seen in figure 2.7, although the structure of the system is apparent, large numbers of individual items are obscured. To overcome this the Cone tree uses transparency and allows the user to rotate the cones so that all the items are accessible.

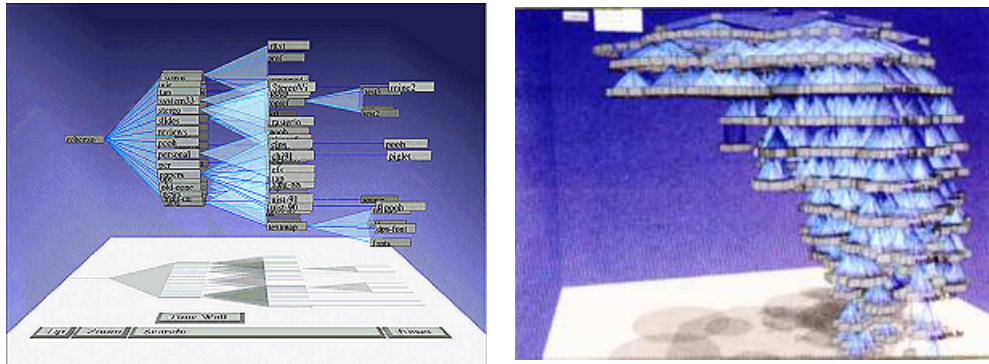


Figure 2.7: Cone Tree.

Robertson et al. (1993) provide analytical evidence of the benefits of using a 3D layout rather than a 2D layout in order to maximise the effective use of screen space and enable visualisation of the entire hierarchy.

2.3.7 *Physiologic Data Visualisation*

Traditional physiologic data displays, as shown in figure 2.8, such as those found in anaesthesiology use discrete, non-interactive data representations. To help anaesthetists detect, diagnose, and treat physiologic conditions more easily and more accurately Bermudez et al. (2000) developed the visualisation shown in figure 2.9. Four complementary interactive displays are used. In each display cardiac and respiratory data is mapped onto 3D objects that match the anaesthetist's mental model of the processes involved. For example the red spheres change shape with each heartbeat. This helps them to diagnose problems. Reference lines, shapes, colours and spacing are all used to indicate 'normal' conditions. Deviation of any of the objects from these norms helps the anaesthetists to detect problems.



Figure 2.8: Traditional physiologic data display.

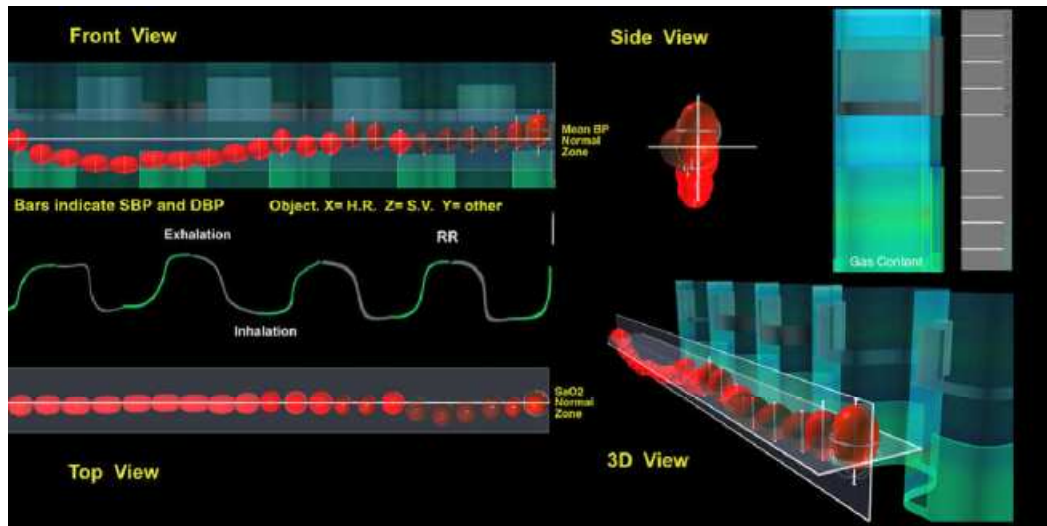


Figure 2.9: Real-time physiologic data visualisation.

Bermudez et al. (2000) compared the new visualisation to the traditional display using test data generated by a body simulator system. The results showed that participants' reaction times to critical events were statistically significantly faster using the visualisation.

2.3.8 Geographic Visualisation

The visualisation of population statistics, health statistics, socio-economic statistics, urban development, climate data, etc., comes under the general heading of *geographic visualisation*. In each case the data has some reference to a geographic location which is referred to by MacEachren et al. (1998) as *georeferenced information*. In many cases geographic visualisations take the form of maps since this is a natural model upon which to overlay the data. Examples of geographic visualisations are shown in figure 2.10.

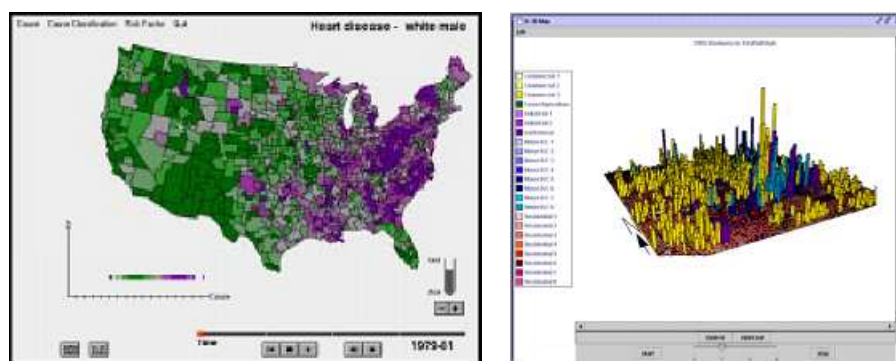


Figure 2.10: Geographic visualisation.

Examples from MacEachren et al. (1998) and Salisbury (2001).

Typically geographic visualisations are used to help users perform spatial and temporal analysis tasks such as finding and comparing patterns, discovering relationships between

different regions, highlighting low and high values, identifying clusters, etc. Due to the large quantities of data involved, interaction and animation techniques are used to assist the user in performing these tasks.

The results of an empirical evaluation conducted by MacEachren et al. (1998) showed that geographic visualisation can successfully support domain experts during typical data exploration tasks such as finding information, discovering patterns and comparing trends.

2.3.9 Summary

This review was not intended to be a comprehensive analysis of visualisation but instead has presented a number of successful visualisation designs as a way of illustrating the following:

- The diverse range of domains for which visualisations have been developed and the types of tasks that they support, e.g. website data, document collections, software, networks, business data, personal data, georeferenced data, file systems, etc.
- The range of presentation and interaction techniques that visualisation designers have developed.
- The variety of designs that visualisation designers have produced. In particular the differences between designs that can be applied to the same domain e.g. CyberGeo Maps, Narcissus, and Cone Trees can all represent website data.
- The factors that influence visualisation design e.g. data, tasks, space limitations, etc.

Further analysis of these and other designs during the remainder of this chapter will help to determine why visualisations work, how they relate to human users, the elements common to all visualisation designs, the factors that designers must consider and the techniques they can use to overcome common visualisation design problems.

2.4 Why Visualisations work

To understand why visualisations work we need to look at how they enhance the capabilities of the humans who use them. There are three aspects of humans that are particularly relevant to visualisation; these are *visual perception*, *memory*, and *reasoning*.

Humans are extremely good at processing and interpreting visual information. We can easily spot a familiar face in a crowd, judge people's emotions from facial expressions, estimate distances, and so on. Throughout our daily lives we are constantly interpreting visual information and yet for the most part we are unaware of the process. This is because the human visual system has evolved highly specialised sub-systems that can interpret vast quantities of visual stimuli rapidly and accurately. Visualisations work because they are

designed to take advantage of these fast visual processing systems, which in turn allows the human user to process information more rapidly.

Visualisations also tend to group and organise related items together in a relatively small visual area. This reduces the search space and may enhance patterns. This is particularly obvious in the Information Mural, which is capable of using very small visual areas. Visualisations can also support perceptual inferences, e.g. the child-parent relationships in the Cone Tree and Narcissus, comparison of visual objects, and so on. In addition, visualisations often use abstraction to present high-level overviews of data. Each of these features enables the user to more easily gain an insight into the data as was illustrated in figure 1.1.

In the context of this thesis memory can be broadly split into two parts. Short-term memory (STM) is limited in capacity and can be easily overloaded. Long-term memory (LTM) on the other hand is capable of storing large amounts of information but requires more effort to do so.

Visualisation acts as an external memory aid that in effect expands short-term memory. This means that more complex tasks involving more information can be handled because the visualisation is acting as a larger and more permanent storage area for the data and any intermediate results involved in those tasks. Consequently, short-term memory is less likely to be overloaded and the human user is able to cope with more data than would be possible without the visualisation. For example, the LifeLines visualisation allows the user to correlate detailed information regarding a patient's medical records.

Shifting the workload from the memory and reasoning systems to the visual system and providing mechanisms to explore and manipulate the data via interaction are all reasons why visualisation works.

2.5 Visual Perception

Visual perception is obviously significant to visualisation. Therefore it is useful to study how the human visual system processes information. The results of these studies could then be used to develop guidelines for visualisation design. This section describes the basic concepts of human visual perception and how they relate to visualisation and is followed by a brief discussion as to why visualisation designs cannot rely on visual perception alone.

2.5.1 Concepts

Any scene can be described in terms of the properties of the objects within the scene and the properties of the scene that act on those objects. These properties are called *visual features*.

For example, imagine describing a single tree in a field on a sunny day. You may describe the length, thickness and colour of the trunk, the texture of the bark, the overall shape of the tree, the colour and shape of the leaves, the angle of the branches, the length and direction of the shadow, the swaying motion of the branches, and so on. All of these properties may be considered visual features of the scene.

A basic model of how information in a scene is processed by the human visual system, as described by Ware (1999), is shown in figure 2.11. In the first stage visual features in the scene are extracted and processed in parallel without any conscious effort. This rapid parallel processing is referred to as *pre-attentive* processing. The second stage is broadly composed of object recognition and environment interaction. The object recognition system tries to match the visual features processed in stage one with objects stored in long-term memory. This is a slow serial process that requires focused attention, referred to as *attentive* processing. An example of attentive processing is reading the text in a document. The environment interaction system relates to tasks involving eye-hand coordination and manipulation of objects in the environment.

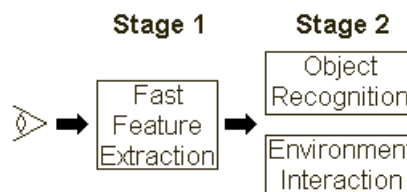


Figure 2.11: Basic model of human visual processing system.
(Modified from Ware (1999))

A visualisation can be thought of as a visual representation of data, i.e. a data scene. One of the aims of visualisation is to make use of the fast pre-attentive processing abilities of the visual system in order to allow rapid interpretation of that scene. However, before this aim can be achieved it is necessary to determine which visual features can be processed pre-attentively, the conditions that allow pre-attentive processing to occur, and how objects within the scene should be organised.

2.5.2 Experiments to Determine Pre-Attentive Features

The most common method used to determine if a visual feature can be processed pre-attentively is to conduct a *target detection* task. A pre-attentively distinct target object that possesses the visual feature in question is placed in a field of *distracter* objects. The task is to determine the presence or absence of the target object. To be classed as pre-attentive the task must be completed in a fixed time independent of the number of distracter objects. Ware (1999) defines the amount of fixed time as 10msec per item. Ideally this would result in a graph similar to that shown in figure 2.12 with a near constant response time for targets with unique pre-attentive features.

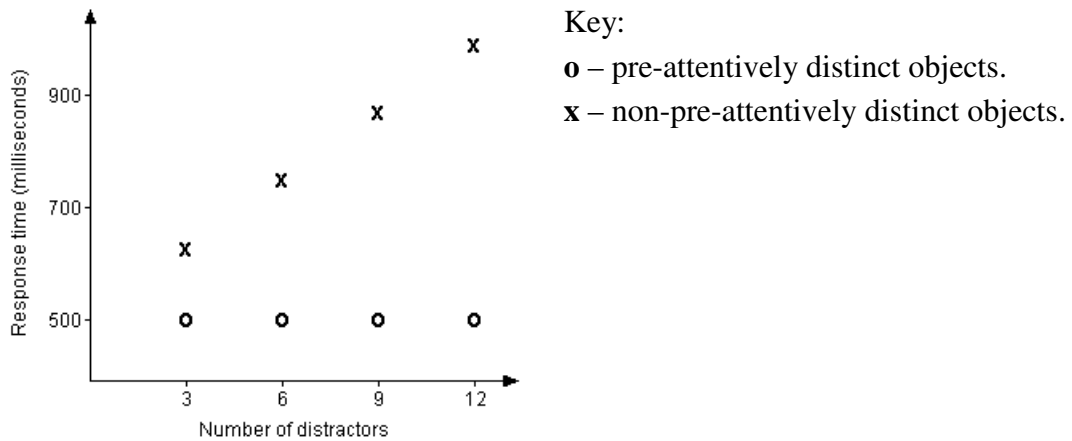
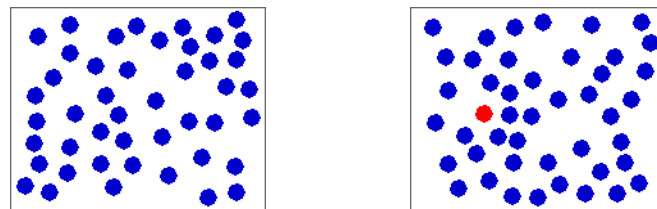


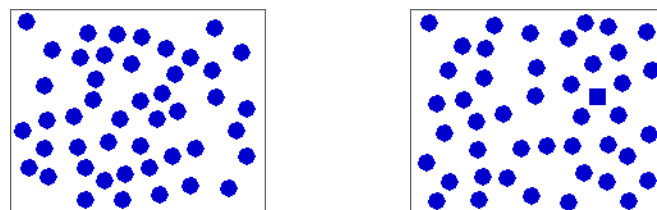
Figure 2.12: Response times for pre-attentive and non-pre-attentive targets. (Ware (1999))

The following example illustrates how a target detection task would be used to determine if the visual feature *hue* could be processed pre-attentively. A participant would be shown a series of displays such as figures 2.13.a and 2.13.b for a short fixed time period, e.g. less than 200 msec. If the participant can successfully identify the presence or absence of the target object, in this case a red circle, within the fixed time, the visual feature is classed as pre-attentive. In this case hue is a pre-attentive feature. A similar experiment could be conducted for *shape* as shown in figures 2.14.a and 2.14.b.



(a) Target object absent. (b) Target object present.

Figure 2.13: Target detection using hue.



(a) Target object absent. (b) Target object present.

Figure 2.14: Target detection using shape.

The results of experiments such as those described above have established which visual features can be processed pre-attentively. Although this is not an exhaustive list, Ware (1999) has categorised these pre-attentive features as follows:

2.5 Visual Perception

Form		
Line orientation	Line collinearity	Spatial grouping
Line length	Size	Added marks
Line width	Curvature	Numerosity
Colour		
Hue	Intensity	
Motion		
Flicker	Direction of motion	
Spatial Position		
2D Position	Stereoscopic depth	
Convex/concave shape from shading		

In addition to target detection, Healey et al. (1993) have also identified pre-attentive features using *boundary detection*, *group tracking* and *counting* experiments. An example of boundary detection using hue is shown in figure 2.15.

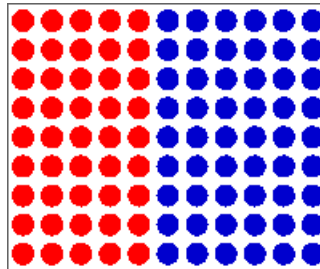


Figure 2.15: Boundary detection using hue.

Unfortunately visualisations often combine a number of visual features, which results in a scene that is far more complicated than those used in standard perceptual experiments. If visualisation is to try to make use of the perceptual abilities of humans then the effect of combining visual features in this way must be determined.

In the example in figure 2.16, the target object, a red circle, has no unique features. Instead it includes features from both type of distracter, the hue (red) from the squares and the shape from the circles. This conjunction of features in the target object inhibits pre-attentive processing. The only way the target can be detected is by performing a slow serial search. This is known as a *conjunction search*.

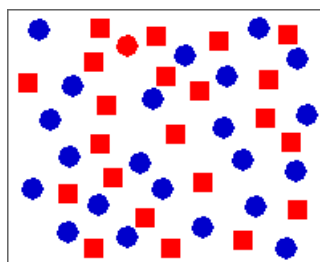
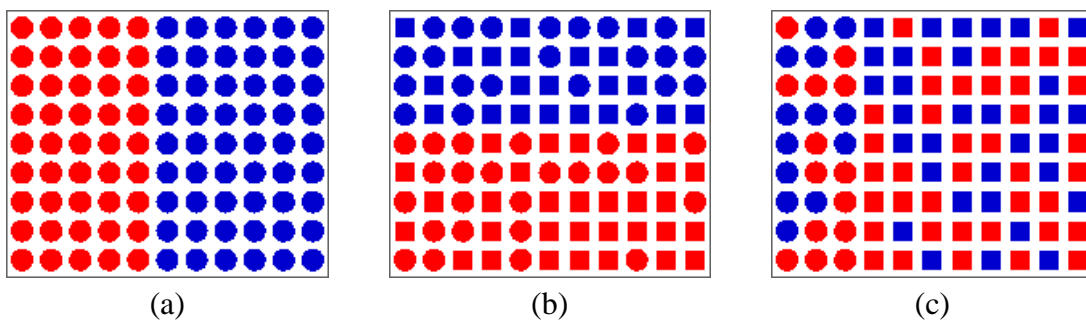


Figure 2.16: Conjunction search.

The target object, the red circle, has no unique features.
A serial search must be performed to find the target object.

The problem of conjunction search is significant to visualisation design. The combination of several visual features obviously affects the rate at which the information presented can be searched, processed, and interpreted. Thus performance is reduced. Depending on the type of task they are trying to support, visualisation designers must carefully consider the visual features that they use and, if necessary, try to ensure that each visual object has some unique visual feature.

There is evidence that the visual system processes the visual features of a scene in a particular order, giving some visual features a higher priority than others. This is known as *feature hierarchy*. An example of this prioritising effect is shown in figures 2.17.a to 2.17.c. In figure 2.17.a the hue boundary is easily detected, in figure 2.17.b it can be seen that variation in shape has no effect on hue boundary detection whereas in figure 2.17.c the variation in hue does interfere with shape boundary detection. From this evidence Callaghan (1989,1990) concluded that the visual system ranked hue as more important than shape during boundary detection.



(a)

(b)

(c)

(a) Vertical hue boundary pre-attentively detected.

(b) Horizontal hue boundary pre-attentively detected despite variation in shape.

(c) Vertical shape boundary is “masked” by random hue.

Figure 2.17: Feature hierarchy.

The ranking imposed by the visual system would suggest that visualisations should make use of the most dominant visual features first. However, Ware (1999), who cites Callaghan (1989), states that this is not possible because the strength of the visual feature and its context have a significant influence. For example, using colour depends on the saturation, the size of the colour patch, and the degree of difference from the surrounding colours. Rheingans and Landreth (1995) list several other ways in which colour can be affected.

Ensuring that objects within a visualisation are visually distinct is not enough to guarantee fast pre-attentive processing. Issues such as the choice and combination of visual features, conjunction search, feature hierarchy, number of distracter objects, and so on, all affect performance and must each be considered during visualisation design. Unfortunately, the complex nature of visualisation often means that one or more of these effects will be

encountered. However, pre-attentive processing is never used in isolation, but instead forms part of a more complex model that includes both memory and reasoning. For example, the first time figure 2.17.c is viewed it is difficult to detect the boundary, but for each subsequent view the boundary is detected immediately because we remember where to look and what to look for. The relationships between the various systems that make up the human brain mean that although low-level visual perception is an important part of visualisation design other human abilities also need to be taken into account.

For an in depth review of various perception experiments the interested reader should refer to Healey et al. (1993, 1995, 1996).

2.5.3 Depth Perception

Many of the visualisation designs reviewed in section 2.3 use three-dimensional views. Therefore it is important that visualisation designers understand how the human visual system processes 3D space and the factors that are relevant to 3D visual perception.

A *depth cue* is a mechanism that provides information about 3D space, in particular the relative locations and sizes of objects in that space. A number of depth cues are related to *perspective* and result from the way in which the 3D world is projected onto the 2D retina of the eye. If we ignore the fact that the retina is a curved surface, we can model perspective by tracing a line from each point in the 3D world through a flat plane perpendicular to the line of sight to a fixed point. Using this model we can see that the apparent size of an object is inversely proportional to its distance from the plane. This is illustrated in figure 2.18. As can be seen in figure 2.19 placing objects of known size in a scene can help to determine the size of adjacent objects. The significance of perspective in determining the relative distances and sizes of objects in a scene is relevant to 3D visualisation designs. This is especially true when the distance and size of the object is related to some aspect of the data. Without these perspective depth cues data could easily be misinterpreted.

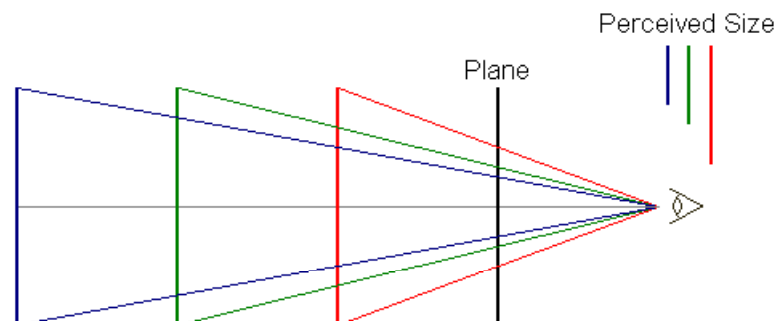


Figure 2.18: Model of perspective.

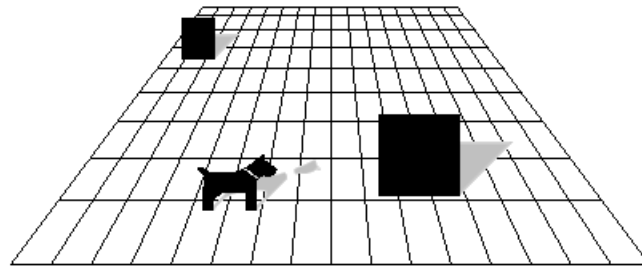


Figure 2.19: Perspective cues.
(Modified from Ware (1999))

Shadows allow the location of an object to be determined relative to some nearby surface. Other depth cues linked to the surface, such as perspective, can then be used to estimate the distance of the object from the observer. Thus the shadows of objects projected onto background planes in a 3D visualisation enable the user to determine the relative distances of those objects.

The intensity of light reflected from objects in the real world is usually greatest at the point where the light hits the surface of the object and then gradually decreases as the surface curves away from the light source. This variation in intensity, or *shading*, helps to identify the shape of the object. In figure 2.20 the objects on the left appear as ‘bumps’ whereas those on the right appear as ‘dents’. If you turn this figure upside down the opposite is true. It seems as though the human visual system is biased towards assuming that light comes from above. This is a reasonable assumption since in the natural world we are used to light coming from the sky. Therefore, a visualisation designer should also make this assumption; to do otherwise may lead to misinterpretation of the scene.

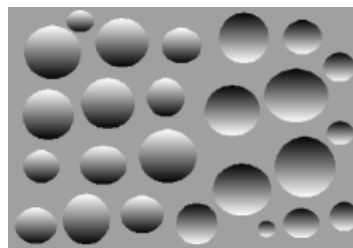


Figure 2.20: Shading depth cue.

When one object overlaps another object it is judged as being closer to the observer. This is known as *occlusion*. Although occlusion is a very strong depth cue it can only provide information about the relative ordering of objects and not about their distance. However, occlusion is often undesirable in visualisation because the occluded data cannot be read, although using semi-transparent objects can help to overcome this problem.

When we attend to an object of interest that object comes into sharp focus. At the same time other objects in the foreground and background become blurred. This would be an excellent

mechanism for highlighting objects of interest in a visualisation but as Ware (1999) points out this technique is computationally expensive and therefore limited in its application.

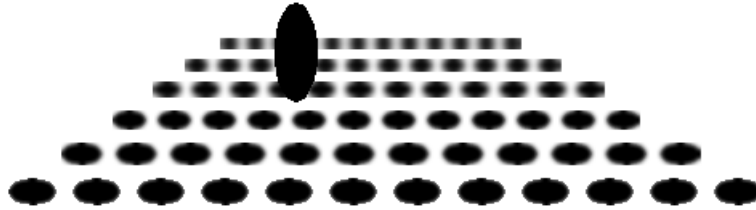


Figure 2.21: Focus depth cue.
(Modified from Ware (1999))

Motion is also an important depth cue. *Motion parallax* is an effect we commonly experience whereby objects in the foreground appear to move more rapidly than objects in the background. For example, when you look out of a car window as it travels along a road the lampposts nearby move past quickly, but trees in the distance move by relatively slowly. Rotation is another form of motion. If we rotate a 2D line on a computer screen we perceive a 3D object. This is known as the *kinetic depth effect*. These two forms of motion can be used in a visualisation to provide information about the shapes of objects and their relative locations.

Stereoscopic depth perception is an effect caused by the displacement of the eyes sending two slightly different images to the brain. Replicating stereoscopic depth perception requires specialised equipment that is not normally available to visualisation researchers and is hardly ever available to the average user. It is also not the focus of this research. For these reasons it will not be reviewed here.

As well as combining visual features, visualisations that use three dimensions often use several depth cues simultaneously. Ware (1999) notes that it would be useful to have information about the relative values of different depth cues when used in combination, but that unfortunately there is no accepted theory of space perception. However, Ware (1999) does cite evidence that suggests that the human visual system applies different rules for depth cue combinations based on the task requirements. Although further studies are needed to determine what these rules are, if used with care visualisation designers can combine depth cues to enhance visualisations.

For a more detailed description of depth perception, the interested reader should refer to Bruce et al. (1996), Eysenck and Keane (1995), and Ware (1999).

2.5.4 Gestalt Laws

The discussion so far has concentrated on the identification of pre-attentive features that can be used to encode data in visual objects and the limitations of the human visual system in

interpreting those objects. We still need to know the different ways in which these objects can be organised to form a complete scene. The Gestalt Laws, based on the work of Koffka (1935), describe a set of basic principles that define how visual patterns are organised and perceived. The most commonly used Gestalt Laws are *proximity*, *similarity*, *continuity*, *connectedness*, *closure*, and *common fate*. Each of these organising principles has a number of applications in visualisation. By studying these laws visualisation designers can utilise the strengths of the human visual system and so reduce cognitive load.

The law of proximity states that objects that are close together will tend to be perceived as a group. This is demonstrated in figure 2.22 in which small differences in horizontal and vertical distance result in objects being grouped by either rows or columns. Likewise, the similarity law states that objects of similar shape tend to be grouped together.

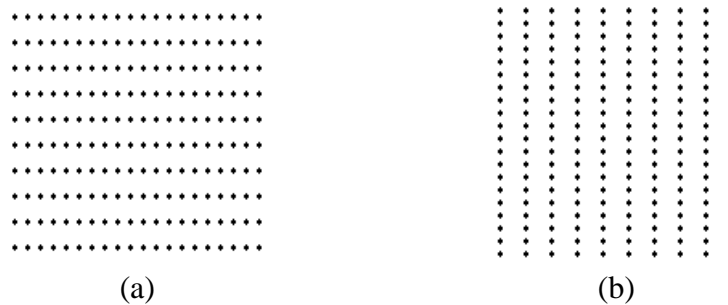


Figure 2.22: Proximity: (a) row dominant, (b) column dominant.

The principle of continuity states that continuous forms are more likely to dominate a scene in comparison to forms that have abrupt changes in direction. Evidence for this can be seen in the node-link diagrams in figure 2.23, which demonstrates that it is much easier to follow the smooth paths in figure 2.23.a than it is those in figure 2.23.b. Connectedness is a form of continuity which states that connected objects are perceived as groups.

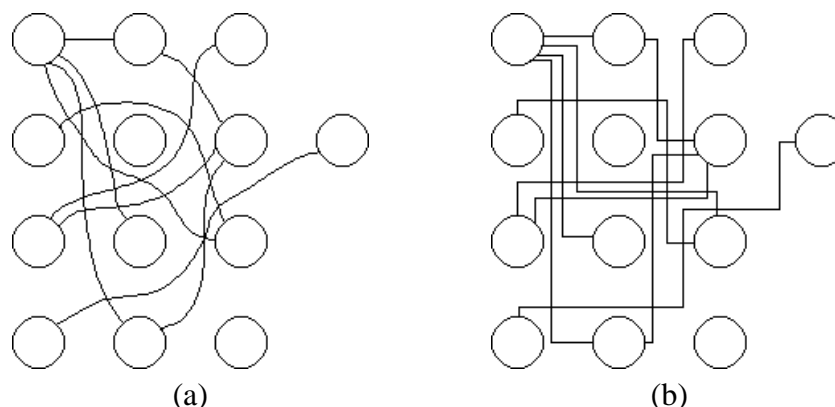


Figure 2.23: Continuity.
(Modified from Ware (1999))

Common region, which is a form of closure, is used in most graphical user interfaces. In this

case the common regions are the ‘windows’ used to divide the screen space into related tasks. The location of objects within a window can be judged relative to the enclosing contour, thus the window also provides a frame of reference.

Objects that have the same orientation or motion are also grouped together. This is known as common fate and is illustrated in figure 2.24.

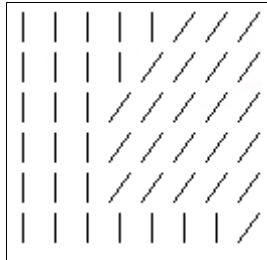


Figure 2.24: Common fate.

Depending on the requirements of the visualisation the Gestalt Laws can be used effectively to partition the visual space, group objects together, enhance patterns, and highlight relationships between objects such as is-a, part-of, belongs-to and so on. This makes them a powerful perceptual tool for visualisation design.

2.5.5 Colour

Colour plays an important role in nearly all visualisations and should therefore be used carefully and consistently throughout the design. Understanding how colour is modelled can help to achieve this. A *colour model* is used to define a colour in terms of the intensity of one or more colour dimensions. The two most commonly used colour models are the RGB colour cube and the HSB cone.

The RGB colour cube defines colours in terms of the intensities of the three primary colours red, green and blue. This is shown in figure 2.25, with the primary colours along each axis, black at the origin (0,0,0), white at the opposite corner (1,1,1) and the secondary colours at the remaining vertices.

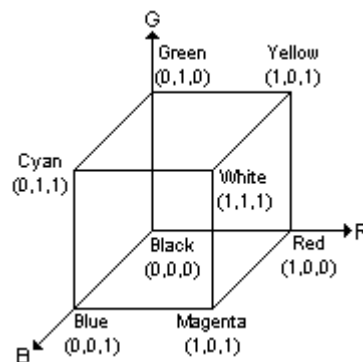


Figure 2.25: RGB Colour Cube.

2.5 Visual Perception

The HSB cone defines colours in terms of *hue*, *saturation* and *brightness*. Hue can be thought of as the main colour; for example, red, yellow, green, blue, cyan, and magenta are all different hues. Hue is represented as an angle on a *colour wheel* as shown in figure 2.26. The saturation indicates how far the hue is from a neutral grey. The higher the saturation value is the closer the colour is to a pure hue at a given illumination. The brightness, sometimes referred to as *value* or *intensity*, describes the luminance. A brightness of zero means the colour is black, a maximum brightness means the colour is at its brightest. The HSB cone model is shown in figure 2.27.

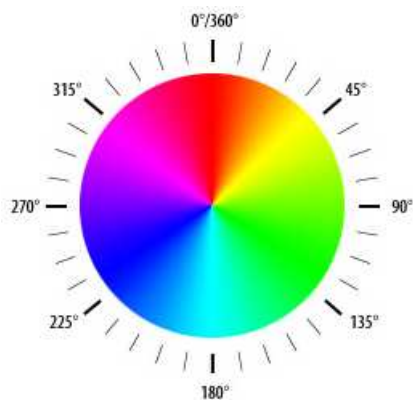


Figure 2.26: Colour Wheel.

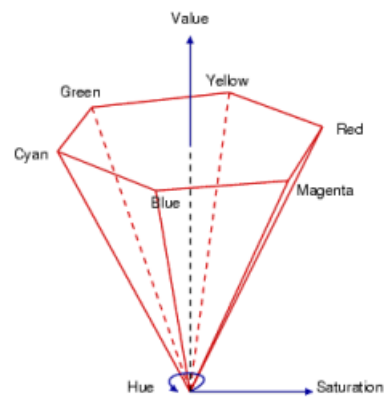


Figure 2.27: HSB Cone.

The main advantage of the HSB model is that it is more intuitive to use than the RGB model allowing colours to be more easily mixed. For example, it is much easier to think of pink as a light red rather than as a combination of full red, half-green and half-blue. The HSB model also makes it easier to define a set of matching colours by simply altering the brightness.

Colour has many uses in visualisation design. Colour can be used to highlight significant data, distinguish between different categories, group items, represent scales, and so on. It is important to use colour carefully as the incorrect use of colours may lead to the misinterpretation of data. Understanding the colour models can help to avoid these problems.

Colour is a complex visual feature that has been studied for many decades. It is important that colour be used carefully in visualisation designs. However, it is not the focus of this research to describe in detail when, where and how to use colour effectively, nonetheless certain common issues pertaining to the use of colour should be mentioned. Briefly the issues to consider include:

- Limiting the number of colours that are used to represent discrete data. Too many colours overload the user, making it difficult to remember and interpret colour codes correctly.
- Avoiding specific colour combinations such as blue on a black background. This is especially true for small objects.

- Being aware of different cultural interpretations of colour. For example, in Western cultures red means stop or danger, and green means go or safe, but in some Eastern cultures the opposite is true.
- Considering the needs of colour-blind users. Red-green colour blindness is common therefore these colours may need to be avoided.
- Using colour consistently.
- Ensuring that the colour coding supports the task.

2.5.6 Examples of Perceptual Guides Applied to Visualisation Design

In order to develop effective guidelines it is necessary to conduct perceptual experiments. However, the application of those guidelines in a real-world setting introduces complexity that was not present in the experimental environment, and can reveal unforeseen problems. Therefore it is useful to analyse a few specific instances of their application.

2.5.6.1 Military Campaign

In the first example Ware (1999) designs a set of visual objects to be used as part of a military map. In this case each visual object represents a different type of real-world object such as aircraft, tanks, buildings, etc. The task is to distinguish the type and owner of each object displayed. Ware uses shape to make each visual object type pre-attentively distinct and different greys to highlight the difference between friendly and enemy targets. In addition the map must display both natural and manmade features such as rivers, lakes, roads, cities, etc. The resulting design is shown in figure 2.28.

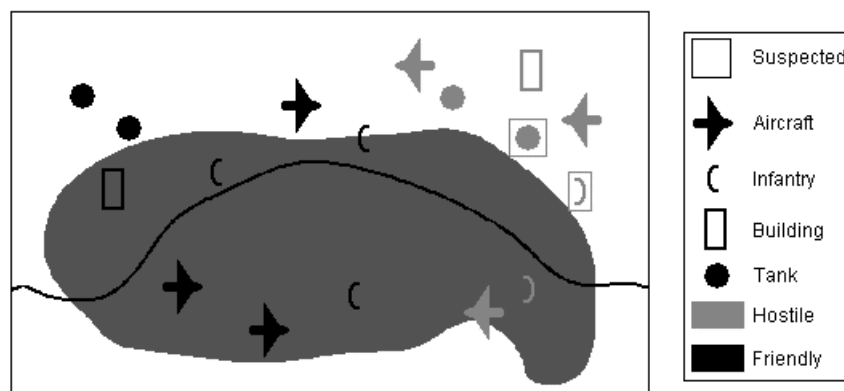


Figure 2.28: A set of symbols for a military command and control display.

Although Ware admits that this is a simplified version of what would normally be a far more complicated display, it does highlight several problems.

In real military scenarios friendly and enemy objects tend to cluster around certain locations such as bases, transport routes, battle lines, and so on. These clusters of objects often lead to

visual clutter and the large number of potentially overlapping distracter objects can make it difficult to locate individual items.

A less serious problem arises from the use of grey values. Many cultures think of safe, good, or friendly objects as green and dangerous, bad, or enemy objects as red, although this is not always the case. This means that using grey values, although “culture safe”, may be considered as non-connotative. However, changing the encoding to use red and green rather than grey could have serious consequences. Visual objects may simply “disappear” depending on the background terrain or may lose their pre-attentive properties. There are methods to overcome these problems but this example illustrates how a slight change in encoding can radically alter the effectiveness of a design.

Despite these problems the overall design allows pre-attentive identification of target objects based on shape and pre-attentive classification of friendly/enemy objects by grey value. A similar design was used in the experimental trials conducted as part of this research, details of which can be found in section 5.3.4.

2.5.6.2 *Salmon Tracking*

The results of visual perception research by Healey et al. (1995) were applied to a salmon tracking simulation. The aim of the application was to determine how ocean temperatures affect salmon migration routes. The encoding used in the simulation was as follows:

- Presence or absence of salmon → Hue
 - Red indicates the presence of salmon at that location.
 - Blue indicates the absence of salmon at that location.
- Temperature of the ocean → Shape
 - Circle indicates a region of cold water.
 - Square indicates a region of warm water.

This mapping does not seem intuitive. A more connotative mapping for most cultures would be to map temperature to hue, using red as hot and blue as cold. However, this was the mapping predicted by the research. According to Healey et al. (1995) reversing the mapping to the seemingly more intuitive version made the patterns of the ocean temperatures more prominent, therefore tracking the salmon, which was the aim of the visualisation, became much more difficult.

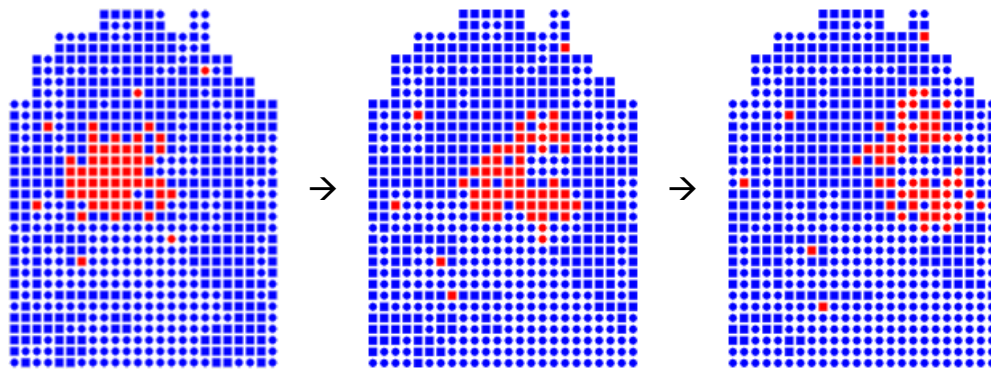


Figure 2.29: Salmon tracking simulation.

(Recreated from Healey et al. (1995))

2.5.6.3 Summary

The military campaign visualisation, the salmon tracking visualisation, and the visualisations presented in section 2.3, all demonstrate typical ways that the Gestalt laws are used to segregate information, highlight similarities, and convey relationships between objects. However, these examples also reveal some of the difficulties that designers face when trying to develop visualisations that can be easily and accurately interpreted.

In general designing a visualisation to support more tasks and display more information may result in a degradation of the degree to which various aspects of the display can be processed. For example, adding information such as flight paths and danger levels to Ware's map, or using techniques such as animation, all increase the complexity of the display, making pre-attentive processing less likely and increasing the cognitive workload. However, these requirements are something that visualisation designers often have to cope with and so have developed techniques that try to maintain the effectiveness of a visualisation. A more comprehensive set of examples together with the techniques visualisation designers use is described in section 2.8.

2.5.7 Limitations on the use of the perceptual system

The perceptual system is capable of processing large amounts of information rapidly. It is clear therefore that visualisations should be designed to make use of this system. However, visualisation designers cannot rely on the perceptual system alone. Due to the complex nature of the data and tasks that visualisations support, it is often not possible to meet the conditions required for ideal perceptual processing. Effects such as cognitive search, feature hierarchy, colour conflicts, depth perception issues, etc., will be encountered. In addition, although our knowledge of the perceptual system is improving, it is still incomplete. Consequently the use of perception is limited and it is necessary for designers to develop specific techniques that

can overcome these problems and so allow them to build effective visualisations. Examples of these techniques can be found in section 2.8.

2.6 Data

Visualisation is the visual representation of data and, as we have seen in section 2.3, this data can come in various forms depending on the domain. It is necessary to study this data in a domain-independent manner so that we can define various generic data characteristics and understand how those characteristics influence visualisation design.

2.6.1 Concepts

The datasets that a visualisation tries to represent consist of a number of *data items* (or *tuples*). Each item is made up of a number of *attributes* (or *fields*) each of which represents a *dimension* of the data. A *schema* describes the name, type and possibly other meta-data about each attribute. Table 2.1 shows a dataset for a buyers guide to off-road capable cars. The top row defines the attribute names, each of the remaining rows in the table represents a single data item, each column represents an attribute and each cell contains the value of an attribute for a particular item. This dataset will be used in examples in the proceeding sections.

Manufacturer	Model	Colour	Price £	Off-Road Ability
Nissan	Terrano II	Blue	17,000	Good
Land Rover	Freelander	Green	18,000	Good
...
Suzuki	Vitara	Black	12,000	Excellent
Toyota	Landcruiser	Silver	24,000	Good

Table 2.1: Example dataset.

2.6.2 Type, Range, Reliability

To design perceptually effective visualisations it is essential to know the type of data that is to be displayed. This is because the data type will affect the choice of visual feature. There are three basic data types:

- *Nominal* (N): a set that has no order. Identifying labels attached to items may be considered nominal data, e.g. names of countries, names of people, colour names, etc.
- *Ordinal* (O): an ordered set. It is possible to say that a particular item in the set comes before or after another item. For example, <small, medium, large>, days of the week, etc.
- *Quantitative* (Q): a numerical set, e.g. 2.5, 100, -3, etc. Quantitative data can be split into:
 - *Interval* (I): using an interval scale it is possible to calculate the difference between data values. For example aircraft arrival and departure times can be defined using an interval scale.

- *Ratio (R)*: items that can be described in terms of ratios. For example item X can be described as being 10 times faster than item Y. Ratio data includes height, weight, money, etc.

Referring to table 2.1 it can be seen that the manufacturer, model and colour of the car are all nominal data types, the price is quantitative and the off-road ability is ordinal. This defines a simple schema for the dataset as shown in table 2.2.

Attribute Name	Attribute Type
Manufacturer	N
Model	N
Colour	N
Price	Q
Off-Road Ability	O

Table 2.2: Schema.

Mackinlay (1986) mapped and ranked visual features to quantitative, ordinal and nominal data types based on the accuracy with which the data could be interpreted. This ranking is shown in figure 2.30 and helps visualisation designers to choose the most appropriate visual feature.

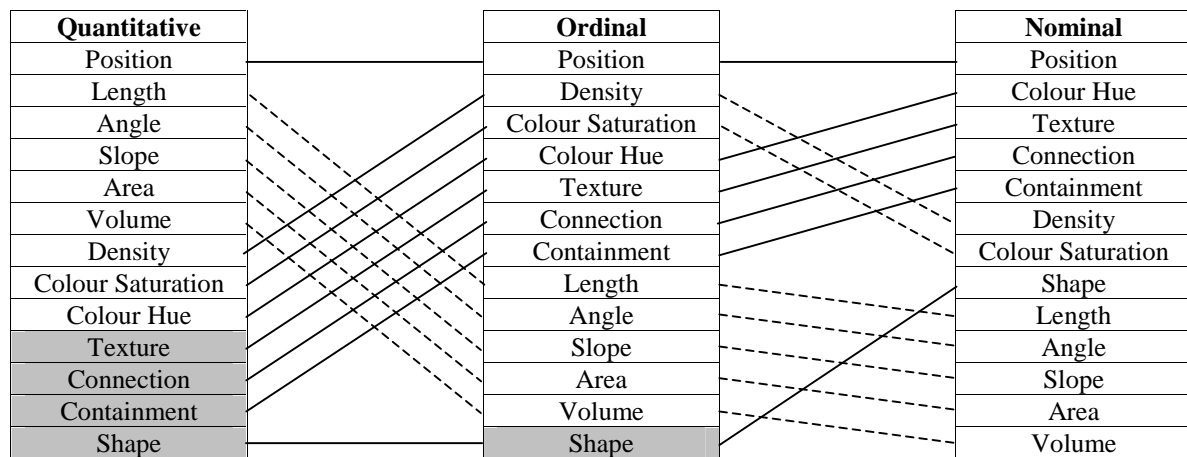


Figure 2.30: Mackinlay's (1986) ranking of visual features to data type.

Note: The features in the grey boxes are not relevant to these types of data.

The granularity and range of data values are also important when deciding which visual features to use. Values can be continuous or discrete. However, only certain visual features are capable of supporting a continuous range of values (e.g. altitudes), or large numbers of discrete values (e.g. names). Casner (1991) has suggested an upper limit for the number of distinct values that can be practically encoded using various visual features. These limits are listed in table 2.3. Using modern computer graphics techniques we could argue that some of these limits are no longer valid.

2.6 Data

Visual Feature	Limit	Visual Feature	Limit	Visual Feature	Limit
Horizontal Position	100	Area	10	Line Thickness	3
Vertical Position	100	Shading	4	Line Dashing	2
Height	50	Connectivity	8	Shape	5
Width	50	Colour	12	Visibility	2
Line Length	50	Labels	∞	Tabular	∞

Table 2.3: Upper limit of values that can be encoded by various visual features.

Data of one type can sometimes be transformed into data of a different type. For example, car prices ranging from £10000-£30000, i.e. quantitative data, can be split into categories such as low price, medium price, and high price, i.e. ordinal data. Obviously some information is lost but this strategy of transforming the data from one type to another can help to reduce the granularity of the data and may be useful when a visualisation is required to show an overview of the entire dataset.

The reliability of the data must also be considered since erroneous data may lead to misinterpretation of information. In fact, some visualisations have been developed that help users identify unreliable or incomplete data.

2.6.3 Structures

The structure of the data can influence the way in which it is displayed. Shneiderman (1996) has defined seven broad data structures including 1-dimensional, 2-dimensional, 3-dimensional, multidimensional, hierarchical, network and temporal. Based on Shneiderman's definitions each of these data structures is briefly described below.

The most common form of 1-dimensional data is text. This text may come from many sources. For example it may form part of a document or be lines of code in a software module. Depending upon the source, additional attributes can be associated with each text term, a term being a word, line, or paragraph. For example in the SeeSoft system (Eick et al. 1992) lines of code can have an associated programmer, number of fixed errors, age, and so on. It is important to note that although the data may be classed as 1-dimensional, any view of that data has to use at least two dimensions.

Two-dimensional data often contains some form of spatial information e.g. geographic maps, urban development maps, organisational floor plans, etc. Again additional attributes such as area type, location, altitude, etc., can be encoded as required. Examples of this can be seen in visualisations such as those developed by Brath (1999) (section 2.3.4), MacEachren et al. (1998) (section 2.3.8) and Salisbury (2001) (section 2.3.8).

Shneiderman (1996) defines 3-dimensional data as being linked to real-world objects such as molecules, the human body, etc. Data of this form is usually gathered using sensors and is

more commonly associated with scientific visualisation. In this case the data source can often provide an appropriate model on which to overlay the data values, as described in section 2.1.

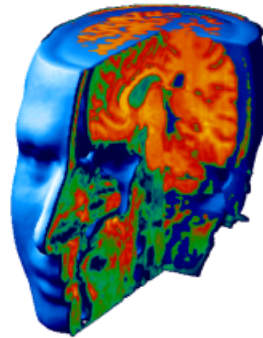


Figure 2.31: MRI data superimposed onto a model of a human head.

Multidimensional data is perhaps the most common data type. Consider the simple example of a personnel record kept in a company database. Information kept in this record could include employee name, age, phone number, position in company, salary, period of employment, the list goes on. Representing all of the dimensions is a difficult task, especially if there is also a large quantity of data.

If the data has within it a parent-child relationship it is said to have a hierarchical or tree structure. The structure of the company mentioned above could contain such a relationship. For example, the managing director is in charge of all departments, each department may have its own manager, each department may consist of several project areas each with their own team leader, and so on. A common hierarchical data source is the file system of a computer. In this case files can be thought of as children of directories and sub-directories can be thought of as children of the parent directory. The Cone Tree described in section 2.3.6 and the TreeMap visualisation developed by Johnson and Shneiderman (1991) are both examples of hierarchical data visualisation.

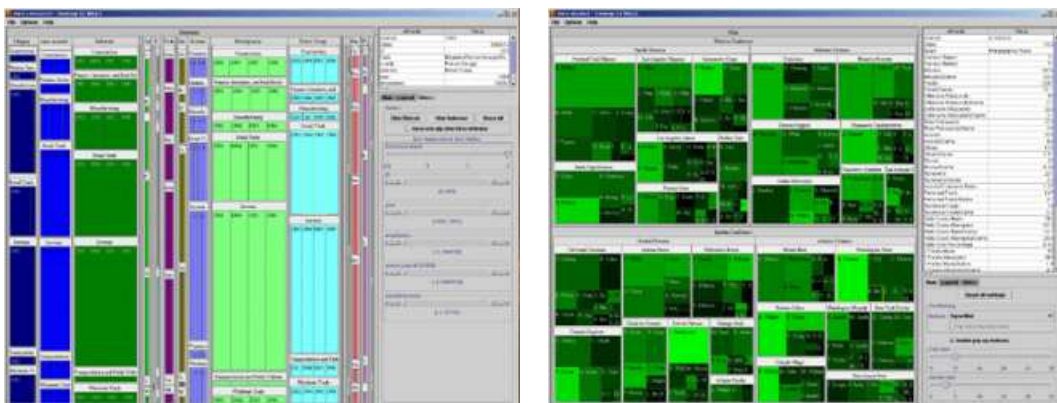


Figure 2.32: TreeMap visualisation of a file system.

Networks are similar to hierarchical structures but allow multiple child-parent relations. This can form loops within the structure. The Internet, websites, and computer networks in general are all good examples of data that has an underlying network structure. These structures are often highly interconnected which means that visualisation designers are faced with the problem of how to represent a large number of connections and at the same time minimise occlusion.

It should be noted that for both hierarchical and network data additional attributes can be associated with the nodes and the links connecting nodes. In some cases the user may be more interested in the non-structure attributes than in the structure. However, even if this is the case, the underlying network structure does provide a ‘natural’, and therefore convenient, form on which to overlay the relevant data. The Arc Map visualisation developed by Cox et al. (1996) shows world-wide Internet traffic, with the height, colour and thickness of the arc indicating the amount of traffic over a two hour period.

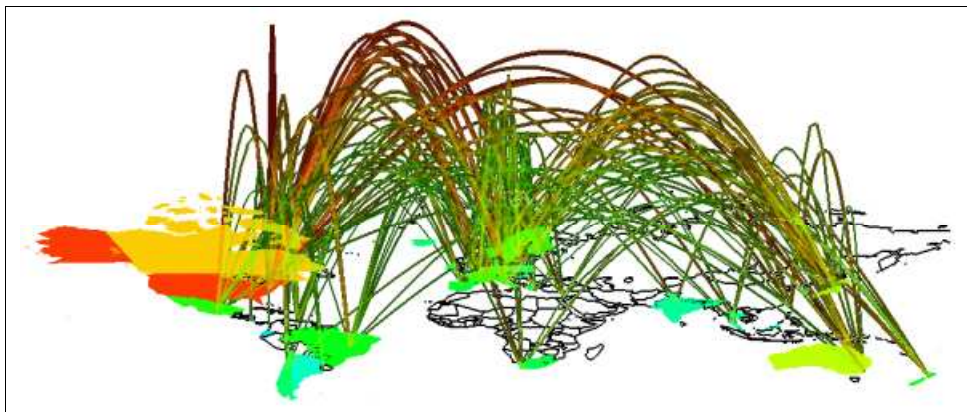


Figure 2.33: Network visualisation – Arc Map.
(Modified from Cox et al. (1996))

Temporal data has the added property of being time dependent. Shneiderman (1996) defines temporal data items as having a start and a finish time and that the times for individual items may overlap. Any data collected over a period of time can be considered temporal data, e.g. medical records, stock market prices, sunspot activity data, etc. Temporal data can be thought of as a series of ‘snap-shots’ or *frames*. Each frame can be represented as a single view in a visualisation, or by using animation, the frames can be made to run in sequence. The LifeLines, CyberGeo Maps, geographic visualisations, business data visualisations, and others, all use temporal data.

It can be seen that the data structures and the concepts that the data represents can help form the foundations of the visual structures to be used in the visualisation. These ‘natural’ structures may more closely match the users cognitive model, or help form an appropriate one, which in turn may amplify cognition and improve the usability of the visualisation.

2.7 Tasks

Visualisation has been applied to a variety of domains so that users can perform domain-specific tasks more effectively and more efficiently and at the same time gain insight into the underlying data. However, as with the data analysis, it is useful to discuss these tasks independently of domain. In this way a generic set of tasks can be identified and their influence on visualisation design understood.

To determine the types of tasks that a visualisation may be required to support it is useful to work through a typical scenario. Imagine a visualisation that shows information relating to residents in a country such as their age, annual income, marital status, district they reside in, type of car they drive, the type of accommodation they live in, etc. Initially the user may wish to find all residents over the age of 25 that have an annual income greater than £12,000. They may then wish to compare those residents that match to see if any relationships exist between them. To do this they may first of all decide to group the residents together based on some criteria such as age e.g. 25-29, 30-34, 35-39, etc. Having grouped the residents the user may then like to know the number of residents in each group, their average annual income, etc., thus calculations must be performed. Finally, if the data has been collected over a period of time the user may wish to discover trends in the data.

Using the hypothetical scenario above, together with the visualisations reviewed in section 2.3, the following tasks, which will be expanded upon in chapter four, have been identified:

- finding items,
- looking at item details,
- comparing items,
- discovering relationships between items,
- grouping or aggregating items,
- performing calculations,
- and identifying trends.

Some of these tasks could be considered compound tasks. For example, it could be argued that comparing items requires the user to look up individual item values. Similarly, high-level tasks such as anomaly detection could be considered a comparison task since you are comparing each item to see if any seem 'out of place'.

The visualisations in section 2.3 are all capable of simultaneously supporting a number of different tasks. For example, the CyberGeo Maps not only reveal changes in the structure and content of websites but also by using the Gestalt law of proximity allows users to visually group files. In addition, the user can visually determine the size and age of files and so is able

to detect files that could be considered anomalous e.g. if a fairly new large file is in a group of small old files then perhaps it has been placed in the wrong directory. Thus relationship, comparison, aggregation, find and look-up tasks are all supported.

Interestingly, the Cone Tree supports the same set of tasks as CyberGeo Maps but is obviously a very different design. Although there are a number of reasons for the differences between the two visualisations, one of the main reasons is due to the primary task that each visualisation was designed to support. In the case of CyberGeo Map this is to reveal changes in the content, and to a lesser extent the structure, of websites over time. The Cone Tree on the other hand is designed to allow the user to find and manipulate files. Thus the type of task has a significant influence on the design and therefore must be considered by the visualisation designer.

2.8 Visualisation Design and Visualisation Techniques

In a way the perceptual issues described in section 2.5 can be thought of as a set of ‘scientific principles’ upon which visualisation design is based. The visualisation designer should be aware of these principles and must try to use them as much as possible. However, as also described in section 2.5 the perceptual system does have limitations. This section examines a variety of techniques that visualisation designers have developed to overcome these limitations.

The basic visualisation process can be described as the transformation of data into a set of visual objects that are then arranged to form a complete scene. To do this, the designer must look at the data items required to support each task, and represent each relevant data attribute of those items as a visual feature. For example, the data item ‘car’ may have attributes such as ‘price’, ‘colour’, ‘# of owners’, etc., which can then be mapped to different features of a visual object. So if the visual object is a circle, the car’s price can be mapped to the circle’s position, the car’s colour to the circle’s colour, the number of owners to the circle’s size, and so on. However, the designer must also organise these visual objects in the view using some suitable structure.

Using the principles of perception the designer can make an informed decision as to which data attribute to visual feature mappings to use, although intuition often plays a part. However, the designer is still faced with the problems of how to display a large quantity of items clearly and how to encode several data attributes effectively. The problem of how to display a large number of data items is compounded by the fact that there is a limited amount of screen space available. Leading on from these problems are a number of separate issues such as how to allow the user access to detailed information, how to avoid overloading the user with too much information, and how to let the user navigate the data space. Fortunately

visualisation designers have developed numerous techniques to help overcome these problems.

2.8.1 Using three dimensions

One approach to the problem of limited screen space is to use a three-dimensional (3D) display. The debate about whether or not two-dimensional (2D) displays are better than 3D displays is ongoing. Both have advantages and disadvantages. Perhaps one of the biggest advantages of using three dimensions is that it allows access to a much larger volume of space, which means that more data can be displayed.

Unfortunately placing objects in a 3D space can lead to problems such as occlusion, which is when objects in the foreground obscure objects in the background, although this also occurs in 2D displays. One technique to overcome occlusion is to use transparency, unfortunately this can make discerning individual data objects more difficult.

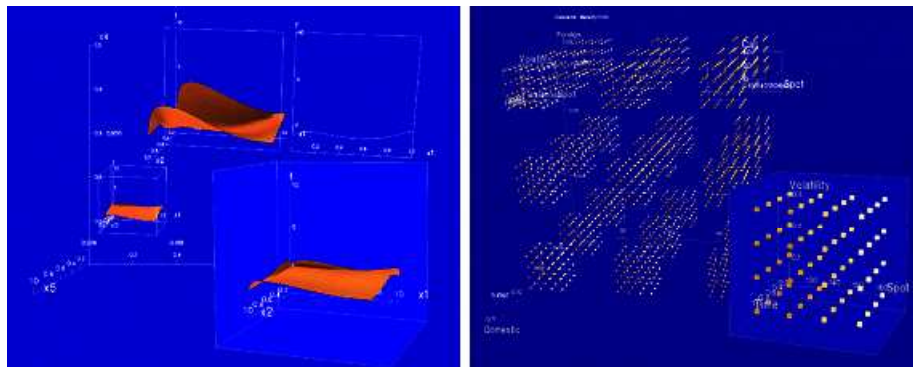


Figure 2.34: Example of occlusion in a 3D data space.

Worlds-within-Worlds (Feiner and Beshers (1990)).

Depth perception is also a problem. The display viewed on a computer monitor is actually a 2D projection of a 3D space. This means that it can be difficult to determine the relative spatial locations of different objects. Is an object small but close to the viewer or large and far away? Depth cues such as those described in section 2.5.3 can help to overcome some of the problems associated with 3D displays. For example, techniques such as shadows, anchor lines, grids, etc., have been used to provide a *reference context* that helps users identify the locations and sizes of objects in a scene. Unfortunately including a reference context adds more ‘ink’ to the display which may obscure data.

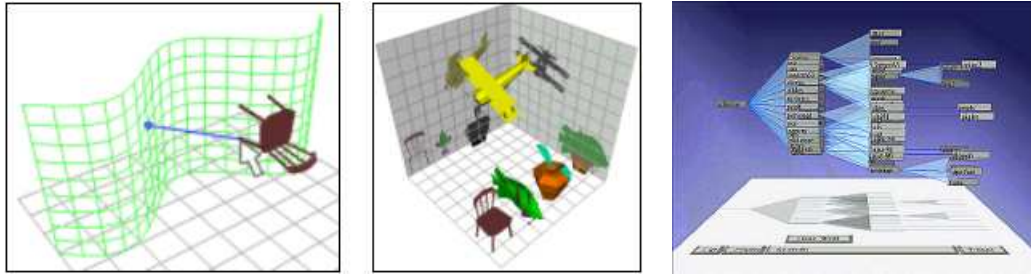


Figure 2.35: Reference context. Anchor line and shadow techniques.

Navigation in 3D space must also be considered. Allowing the user to alter the viewpoint can help to overcome occlusion and zoom in on items of interest. The typical controls available to a desktop PC user are the mouse and keyboard. These devices are not ideal for 3D navigation. Brath (1999) provides a comprehensive account of the problems associated with various navigation techniques. He concludes that a navigation model should always keep the scene on screen; prevent the scene from rolling, in other words use a “steady-cam” model; provide consistent interaction; provide fast feedback i.e. rapid scene updates; provide alternatives to mouse navigation; and finally provide a mechanism that allows the user to quickly and easily return to a previous viewpoint. These are important points that should be adhered to by visualisation designers.

Another important aspect of 3D navigation is that of travelling through a 3D space. This can be done ‘smoothly’, where the user progresses at a fixed rate towards the destination, or the user can be ‘teleported’ instantly for one location to another. However, providing a smooth transition allows the user to maintain their context in the data space, which helps prevent them from getting ‘lost’.

It is evident from the number of existing visualisation designs that use three dimensions, that this is a popular approach. However from the brief discussion above it can be seen that doing so introduces a number of problems. Brath (1999) recommends that 3D should be avoided if possible, especially if the number of data items is low. Visualisation designers should carefully consider the pros and cons of using 3D before committing to this type of display.

2.8.2 Overview and Detail

A different approach to the problem of a lack of screen space is to combine an overview of the entire data set with a detailed view of a subset of the data. This is commonly referred to as an *overview and detail* technique. It should be noted that the overview does not have to be a scaled version of the detailed view; it can be some other form of reduced representation.

This technique has a number of advantages. Using a high level overview of the data means that more data can be displayed using less screen space, it also reduces the space the user has

to search and allows the user to perform visual pattern detection. In addition, by being able to see all the data at once the user can more easily decide which area of the data to look at in more detail, this acts as an aid to navigation.

Having chosen an area on the overview to investigate, the user effectively zooms in to get a detailed view of the data, which is presented in the main window. There are two types of zoom that can be performed. A *spatial zoom* produces an enlarged version of a selected area of the overview, whereas a *semantic zoom* generates a view with a different appearance to that of the overview. This means that the overview and the detailed view do not have to use the same visual structure.

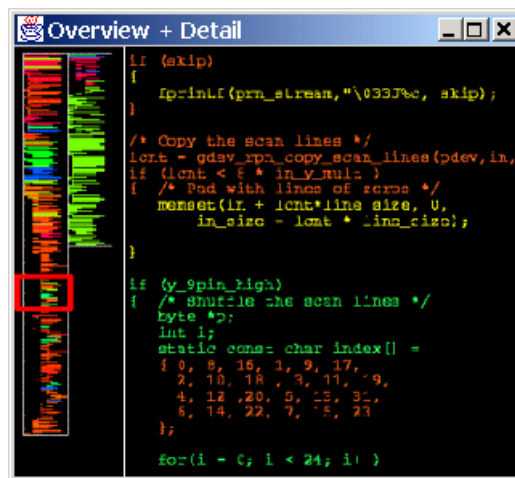


Figure 2.36: Overview and detail.
(Modified from Eick et al. (1992))

Obviously the overview and detailed view need to be linked in some way. The data displayed in the detailed view needs to be highlighted in the overview and actions in the overview must cause the detailed view to be updated. Interaction in the overview usually consists of either clicking on items of interest or manipulating a *navigation box* embedded in the display. Dragging the navigation box to a different area of the overview causes the detailed view to display the corresponding data area. Similarly, if the detailed view is updated by some other mechanism then the navigation box in the overview should be adjusted accordingly. It is exactly this approach that was implemented in one of the visualisation designs used in the experimental trials described in chapters five and six.

This overview and detail technique is a common solution to the problems presented by large datasets and a lack of screen space and has been used in a variety of visualisations such as the FilmFinder (Ahlberg and Shneiderman 1994), LifeLines (Plaisant, et al. 1996), SeeSoft (Eick, et al. 1992), The Information Mural (Jerding and Stasko 1997), and others.

2.8 Visualisation Design and Visualisation Techniques

```

1 #define DIG 40
2 #include <stdio.h>
3
4 main()
5 {
6     int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
7
8     while((c=getchar()) != EOF){
9         if(c >= '0' && c <= '9'){
10            x[0] = 10 * x[0] + (c - '0');
11            for(i=1; i<k; i++){
12                x[i] = 10 * x[i]
13                + x[i-1]/10000;
14                x[i-1] %= 10000;
15            }
16        } else {
17            switch(c){
18                case '+':
19                    t[0] = (t[0] + x[0]);
20                    for(i=1; i<k; i++){
21                        t[i] = t[i] + x[i]
22                        + t[i-1]/10000;
23                        t[i-1] %= 10000;
24                    }
25                    t[k-1] %= 10000;
26                    break;
27                case '-':
28                    t[0] = (t[0] + 10000)
29                    - x[0];
30                    for(i=1; i<k; i++){
31                        t[i] = (t[i] + 10000)
32                        - x[i]
33                        - (1 - t[i-1])/10000;
34                        t[i-1] %= 10000;
35                    }
36                    t[k-1] %= 10000;
37                    break;
38                case 'e':
39                    for(i=0; i<k; i++) t[i] = x[i];
40                    break;
41                case 'q':
42                    exit(0);
43                default:
44                    noprint = 1;
45                    break;
46            }
47            if(!noprint){
48                for(i=k-1; t[i] <= 0 && i > 0; i--){
49                    printf("%d", t[i]);
50                    if(i > 0) {
51                        for(i--; i >= 0; i--){
52                            printf("%04d", t[i]);
53                        }
54                        putchar('\n');
55                        for(i=0; i > k; i++) x[i] = 0;
56                    }
57                    noprint = 0;
58                }
59            }
60        }
61    }

```

(a) Full view of a C program.

Box shows lines in a typical “flat” view, as shown in (b). Underline shows lines in a FISHEYE view, as shown in (c).

```

28 t[0] = (t[0] + 10000)
29 - x[0];
30 for(i=1; i<k; i++){
31     t[i] = (t[i] + 10000)
32     - x[i]
33     - (1 - t[i-1])/10000;
34     t[i-1] %= 10000;
35 }
36 t[k-1] %= 10000;
37 break;
38 case 'e':
39     for(i=0; i<k; i++) t[i] = x[i];
40     break;
41 case 'q':
42     exit(0);
43 default:
44     noprint = 1;
45     break;
46 }
47 if(!noprint){
48     for(i=k-1; t[i] <= 0 && i > 0; i--){
49         printf("%d", t[i]);
50         if(i > 0) {
51             for(i--; i >= 0; i--){
52                 printf("%04d", t[i]);
53             }
54             putchar('\n');
55             for(i=0; i > k; i++) x[i] = 0;
56         }
57         noprint = 0;
58     }
59 }

```

(b) “Flat” view.
 >> indicates current line

```

1 #define DIG 40
2 #include <stdio.h>
... 4 main()
5 {
6     int c, i, x[DIG/4], t[DIG/4], k = DIG/4, noprint = 0;
... 8     while((c=getchar()) != EOF){
9         if(c >= '0' && c <= '9'){
... 16         } else {
17             switch(c){
18                 case '+':
19                 case '-':
... 27                 case 'e':
28                 case 'q':
29                 default:
... 38                 for(i=0; i<k; i++) t[i] = x[i];
39                 break;
40                 case 'q':
41                 default:
... 46                 noprint = 1;
47                 break;
48             }
49             if(!noprint){
50                 for(i=k-1; t[i] <= 0 && i > 0; i--){
51                 }
52                 printf("%04d", t[i]);
53             }
54             putchar('\n');
55             for(i=0; i > k; i++) x[i] = 0;
56         }
57         noprint = 0;
58     }
59 }
60 }
61 }

```

(c) FISHEYE view.
 >> indicates current line.
 ... indicates missing lines.

Figure 2.38: FISHEYE view.

It should be noted that although technically correct the name FISHEYE is slightly confusing since most people normally think of a fisheye lens which causes the space distortion effect shown in figure 2.39.



Figure 2.39: Fisheye lens distortion.

2.8 Visualisation Design and Visualisation Techniques

The Table Lens by Rao and Card (1994) uses the collapsed space focus and context technique to visualise tabular data. Text label information is viewed in the focused areas and other representations such as bars are viewed elsewhere. This can be seen in figure 2.40, with the areas in focus showing the actual data values.

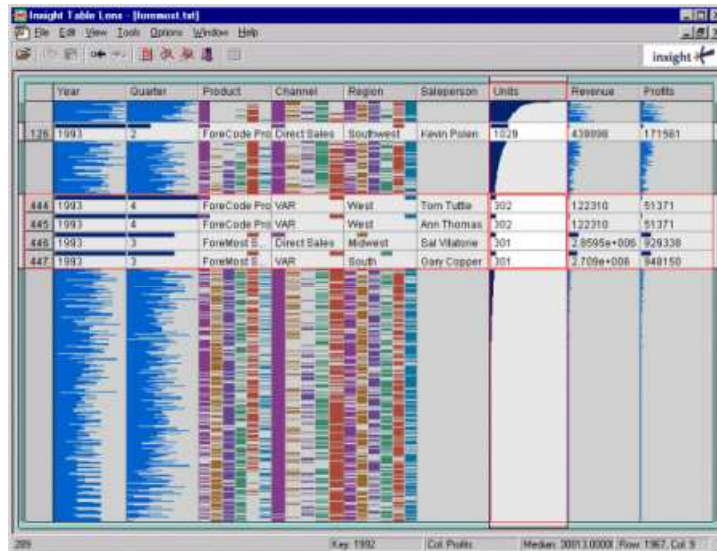


Figure 2.40: Table Lens collapsed space focus and context technique.

2.8.4 Large numbers of items, Visual Clutter and Occlusion

A visualisation that shows large numbers of data items, together with other structure specific items such as connecting lines, reference contexts, etc., often results in visual clutter and occlusion. This is especially true when the data items are displayed in close proximity to each other. It is important to note that visual clutter can occur without occlusion. In such cases, while each individual data item may be identifiable, the proximity of the surrounding items distracts the user, making task execution more difficult.

A common example of visual clutter occurs when trying to display detailed information about data items using text labels. In other words, explicitly displaying the values as text so that they can be read. An example of this is shown in figure 2.41.

2.8 Visualisation Design and Visualisation Techniques

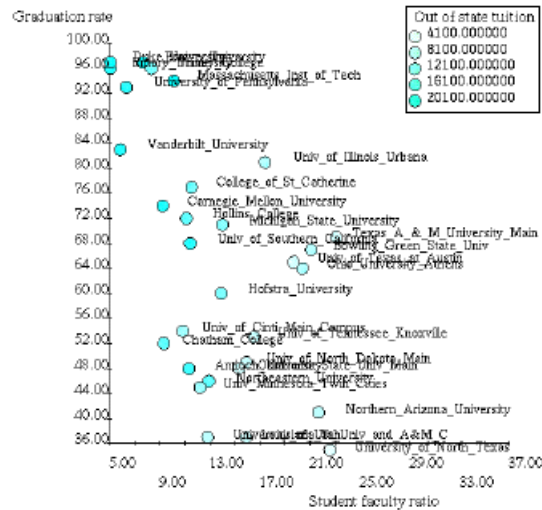


Figure 2.41: Visual clutter.
(Modified from Chuah (2000))

Explicit details are often required by users in order to confirm items of interest. Brath (1999) states that users may also require detailed information in order to confirm their mental model of the visualisation. A technique is required that can supply the user with detailed information when they require it but hide that information when it is not necessary, this would mean the labels could be removed, thereby reducing visual clutter. This can be achieved by ‘popping up’ a temporary window that displays detailed information when the user places the mouse pointer over an item of interest. When the user moves the pointer off the item, the window disappears. This technique is similar to the way tooltips are used in graphical user interfaces to display additional information about toolbar buttons or other controls. We shall refer to this technique as *datatips*.

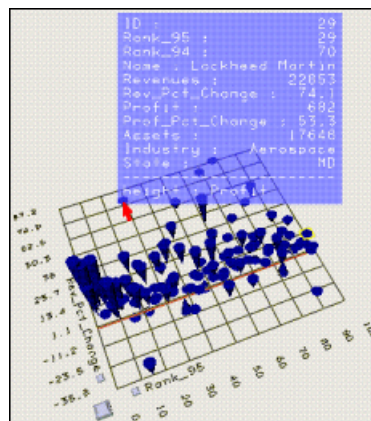


Figure 2.42: Viewing detailed information using a temporary ‘pop up’ window.
(Modified from Brath (1999))

Perhaps the simplest approach to reducing visual clutter is to filter out data that is not relevant to the current task. In many cases, the user is only interested in a particular subset of the data. Providing the user with controls that allow them to filter out unwanted data items helps to

reduce visual clutter. It also helps them to focus on the appropriate information rapidly and accurately. Filtering is one several forms of interaction, which are described in more detail in section 2.8.7. An alternative approach that can also reduce visual clutter is to use abstraction.

Animation can also be used to display a large number of data items and at the same time reduce visual clutter. A large dataset can be divided into a series of “data frames” based on some pre-determined criteria. These frames can then be played in sequence giving the user access to potentially hundreds of thousands of data items. This technique is often used when the data has some temporal component e.g. has been collected over a period of time, and is a useful method of detecting trends and anomalies in the data. As with 3D navigation it is important that the animation is smooth so that the context of each frame is maintained. When using animation in this way, the user should have access to controls that allow them to pause, reverse direction, and step through the animation a frame at a time.

2.8.5 *Small Multiples*

A small number of data dimensions can be encoded relatively easily. For example a standard scatter plot can easily represent four or five data dimensions. Two dimensions can be encoded as positions on the x and y-axes, one as the shape of the mark, one as the mark colour and finally one as the size of the mark. A sixth dimension can be encoded on the z-axis if a 3D scatter plot is used. What happens if we continue to increase the number of dimensions? How can these extra dimensions be encoded given the limited number of visual features available and the limitations on the effective combinations of visual features?

A *small multiple* is a combination of visual features, each one representing a data dimension, packaged together and designed to function as a complete unit. So each ‘mark’ in the example above is a small multiple that uses position, shape, colour, and size to encode different data dimensions. However, if multiple scatter plots are used to represent either different data dimensions of a single dataset or the same data dimensions of multiple datasets, then each scatter plot can also be thought of as a small multiple. In this way several data dimensions or several datasets can be viewed simultaneously.

Chi and Card (1999) use Cone Trees as small multiples to visualise how website content and usage changes over time. In this case the small multiples are embedded in a spreadsheet with each column representing a discrete time and each row being used to threshold the frequency of usage. This is shown in figure 2.43 with red representing high usage and blue low usage.

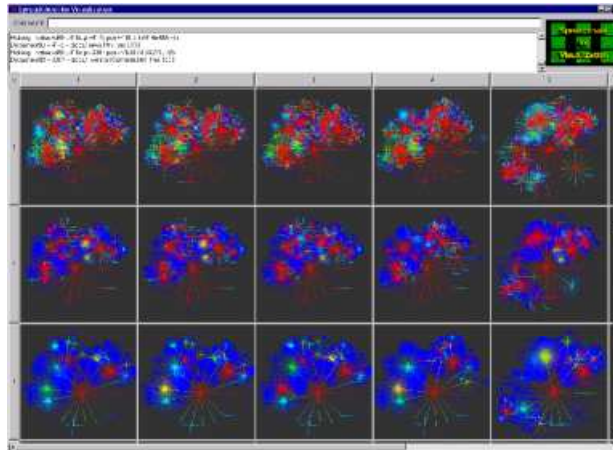


Figure 2.43: Small multiples in Web Analysis Visualisation Spreadsheet.

Another interesting example of the small multiple technique is the VisDB visualisation shown in figure 2.44. Developed by Keim and Kriegel (1993, 1994), and later extended by Kiem et al. (1995), it attempts to overcome the problems of large quantities of multidimensional data and to maximise the use of available screen space by representing each data item as a single pixel in a view, with each view representing a different data dimension.

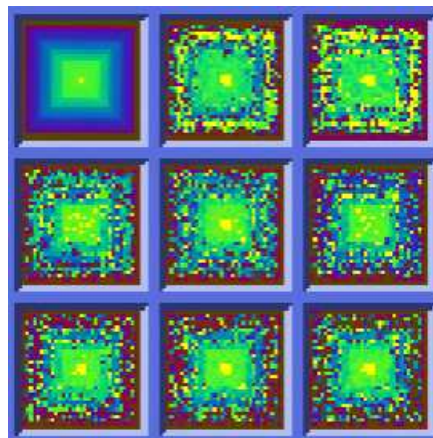


Figure 2.44: Small multiples in VisDB.

As well as presenting multiple data dimensions small multiples can also provide rapid access to large quantities of data. They are also a useful way of comparing datasets or visualising temporal data, especially if presented using a regular structured layout. However, small multiples can occupy relatively large amounts of space and designing effective small multiples can be difficult. Brath (1999) describes a serious case where the mapping of data dimension to visual feature was poorly chosen making it difficult for the user to accurately interpret the values associated with each small multiple.

2.8.6 Multiple Linked Views

In one sense small multiples could be thought of as multiple views. However, whereas small multiples all use the same visual features and are all constructed in the same way, the views in multiple view visualisations, sometimes referred to as *compound visualisations*, are different but complementary. The business data visualisations created by Brath (1999) and the LifeLines visualisation developed by Plaisant et al. (1996) both incorporate multiple views.

Multiple views are an excellent mechanism for allowing the user to find relationships between data items or datasets. However, one of the disadvantages of multiple views is that it is difficult for users to integrate information between views. This is known as *context switching*. For example, if the user has identified a set of interesting data items in one view it may be difficult to locate them in another view and once located it may be difficult to correlate the information. To minimise this problem the views need to be linked. These *multiple linked views* assist the user in finding and integrating the information contained in separate views. Although context switching still occurs, its negative impact is reduced.

Brushing is any mechanism that allows the user to interactively select subsets of the data for further manipulation. In multiple linked views, actions in one view may be propagated to the other views so that, for example, when the user selects items in one view the same items are selected in all the other views. This is also known as *linked-brushing*. In some cases it may also be possible to generate new linked views based on actions in existing views. The overview and detail technique described earlier is an example of a multiple linked view.

North and Shneiderman (1999a, 1999b, 2000) have used the idea of multiple linked views to develop the *Snap Together Visualisation (STV)*. They regard each individual visualisation as a component, which is then linked to other components to solve a more complex task. This is a logical 'divide and conquer' type approach, with the advantage that each visualisation can be specifically designed to solve a small subset of tasks and new visualisations can be built and simply 'plugged-in' as required. The only drawbacks with this approach, in addition to the multiple linked view problems already mentioned, are the need to establish a standard communication protocol between the visualisations and the extra learning effort required by the user to understand and use the connected views.

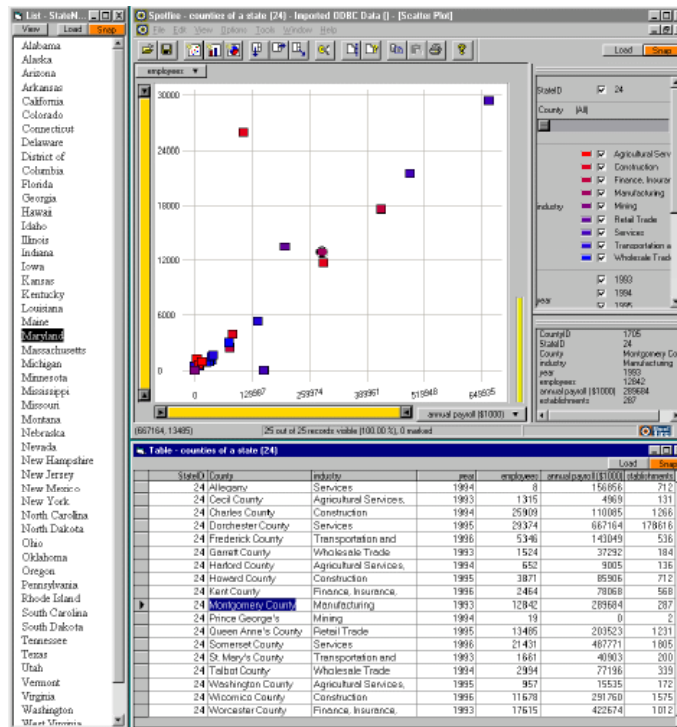


Figure 2.45: Snap-Together Visualisation.

Multiple linked views still need to be laid out, organised, and manipulated, all of which increase the users workload. Laying out views so that they do not overlap is especially difficult, as is identifying the relationship between views. To help solve these problems Kandogan and Shneiderman (1996) have developed a novel approach called *elastic windows*, which provides automatic layout facilities and rapid multi-view operations.

Baldonado et al. (2000) have developed a comprehensive set of guidelines that describe when and how to use multiple linked views and their impact on learning, memory, performance and display space. Guides such as this are useful in assisting the designer in making important design decisions and maintaining the usability of the visualisation.

2.8.7 Interaction

The visualisation presented to the user consists of two parts, a view of the data and a graphical user interface (GUI) associated with the view. The view is a representation of the data that is derived from various data features such as those described in section 2.6, e.g. dimensions, quantity, range, etc., and the task requirements. The GUI augments the view and usually consists of standard graphical components such as menus, buttons, sliders, list boxes, etc. Interaction can be defined as actions taken in both the view and the GUI that achieve user goals.

2.8 Visualisation Design and Visualisation Techniques

The design of effective interaction techniques is obviously important. Shneiderman (1996) argues that by using a well designed and responsive interface the user will be able to recover from errors, repeat actions and predict the effects of carrying out such actions in an easy and intuitive way. He advocates the use of controls such as sliders, checkboxes, lists, buttons, etc., that can be bound to the dataset to prevent erroneous input and linked to the view to provide rapid visual feedback. Interaction methods such as these are examples of *dynamic queries*. These principles are used effectively in the Dynamic Homefinder (Williamson and Shneiderman 1992 ; Shneiderman 1994), FilmFinder (Ahlberg and Shneiderman 1994), LifeLines (Plaisant et al. 1996), and many other systems.

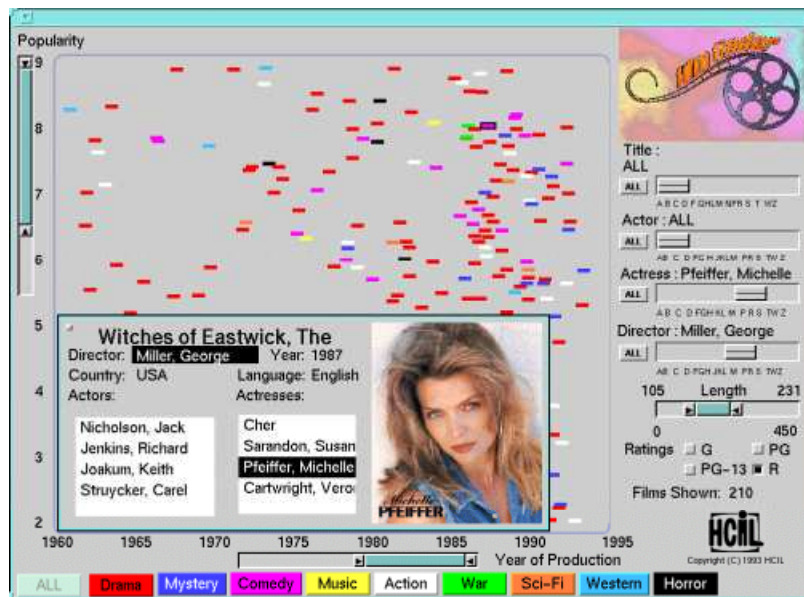


Figure 2.46: FilmFinder.

Actions in the view consist primarily of *object selection* and *navigation* and in many cases are achieved by interacting directly with objects in the view rather than via menus or other controls. This form of interaction is referred to as *direct manipulation*.

Object selection is a direct manipulation action that is often used to initiate some higher-level operation such as filtering, details on demand, brushing, or manipulation. Direct object selection can be achieved by clicking on individual objects or using a *bounding box* technique where large numbers of objects can be selected quickly by dragging a box around them. Indirect object selection is also possible using dynamic queries.

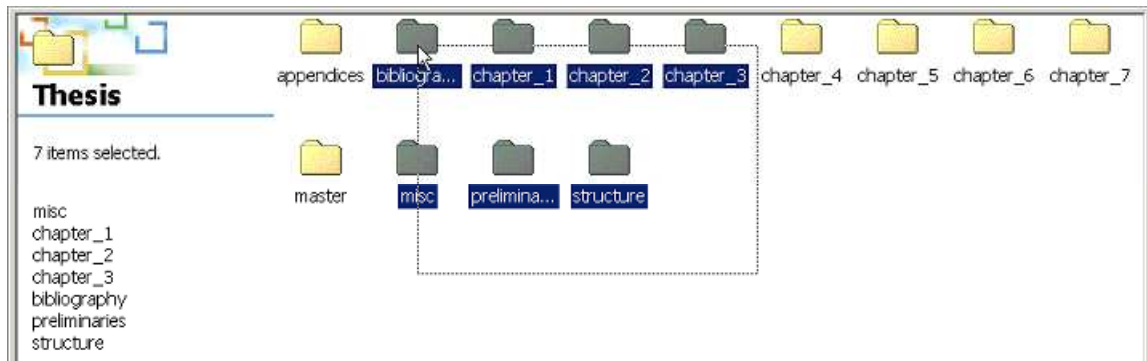
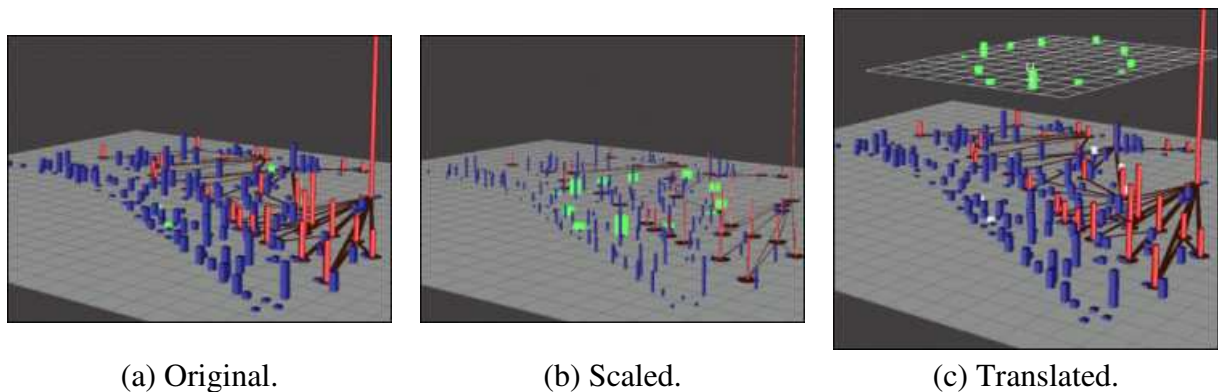


Figure 2.47: Bounding box technique used in Microsoft Windows Explorer™.

Navigation comes in two main forms *spatial* navigation and *detail* navigation. Spatial navigation allows the user to change their viewpoint. For example, in a 3D visualisation the user may be able to ‘fly’ from one location to another in 3D space. The navigation box used in the overview of an overview and detail visualisation may also be considered a form of spatial navigation. Specially designed GUI controls can also be used to achieve spatial navigation. Detail navigation provides the user with a means to change the level of detail presented in the view. This allows them to view entire data sets as well as drill down to specific data values. Again there are a number of techniques that can be used to achieve this, e.g. datatips.

Chuah et al. (1995) have developed a comprehensive set of interaction techniques called *selective dynamic manipulation* (SDM), which satisfy many of the requirements outlined by Shneiderman (1996, 1997). In the SDM system, sets of selected objects can be saved, scaled (figure 2.48.b), translated (figure 2.48.c) or manipulated in various other ways to help analyse the data, overcome occlusion and visual clutter, and in general improve the readability of the display.



(a) Original.

(b) Scaled.

(c) Translated.

Figure 2.48: Selective Dynamic Manipulation (SDM).

Regardless of interaction type the choice of interaction mechanism can affect user performance. For example, the bounding box technique is very good at selecting a large number of objects in fairly close proximity in a coarse way but is less appropriate for fine-grained individual object selection. If the user selects individual objects more often than

multiple objects and the required individual objects are distributed throughout the view, they may become frustrated using the bounding box technique and their performance will suffer.

Interaction requirements depend primarily on the task and the structure of the view. For example, navigation may be via an overview, based on some real-world 3D navigation model, etc., or may not be required at all if all of the relevant data can be displayed on the screen at one time.

The factors described in this section mean that the applicable interaction mechanisms, the choice of interaction technique, the effect of the technique on the user, the frequency of the task, and the coordination of interaction actions between the view and GUI, must all be considered during visualisation design. Doing so will help to increase user performance, reduce the number of errors, reduce learning time, and so on. In other words it will increase the usability of the visualisation.

2.8.8 Visualisation techniques summary

The purpose of this section has been to illustrate the wide variety of techniques that have been developed by visualisation designers to cope with the problems of large data sets, multidimensional data, a lack of screen space and a limited number of compatible visual features. In addition, this section should have further demonstrated the wide variety of tasks and domains that visualisation has been applied to.

However, after seeing the various visualisation designs presented in this section certain questions remain. What factors does the designer need to consider when deciding on the content and form of the visualisation? How does the designer know which visualisation techniques to use and how to apply them? How can the designer determine if one visualisation is better suited to supporting users in their tasks than another? These issues and methods that help solve them are part of the focus of this research.

2.9 Summary

This chapter has tried to establish what visualisation is, why we need it, and how it works with respect to the human visual system. In addition it has demonstrated the wide range of domains to which visualisation has been applied.

It has been shown that visual perception is fundamental to visualisation design and that any design should offload as much work as possible to the rapid processing systems embedded in the human visual system. This allows large quantities of visual information to be processed rapidly and with little conscious effort on the part of the observer. However, perception

researchers have established that the human visual system does have limitations. Restrictions on combinations of visual features, depth perception cues, conjunctive search, feature hierarchy, number of visual objects, etc., all impede the processing of visual information. In addition, our understanding of how the visual system works is still incomplete. Therefore, although perception is important it cannot drive visualisation design on its own.

Fortunately, the human visual system is not the only system involved in the analysis and interpretation of visual information. Memory and reasoning also play a part. Visualisation helps to expand memory and acts as a ‘drawing board’ upon which ideas can be worked out. However, there are still bounds on the amount of information that can be dealt with at any one time, and the drawing board, which in most cases is the computer screen, is limited in size.

In an attempt to overcome the restrictions imposed by perception, memory, and the complexity of the data involved in real-world domains, visualisation designers have developed a number of novel techniques. These include using three dimensions, overview and detail, focus and context, small multiples, multiple linked views, and perhaps most significantly of all interaction. Using these techniques users are given access to large quantities of data that they can effectively analyse and manipulate.

However, designing novel visualisations that are both effective and usable is a difficult task. Although visualisation designers have access to knowledge and techniques from disciplines such as perceptual psychology, computer graphics, and human computer interaction, they still need to be guided in the application of this knowledge. At present visualisation designers do not have this guide and so are forced to rely on limited information, past experience, and intuition. A methodology is required that can focus designers on the relevant factors identified in this chapter (e.g. perception, data characteristics, task characteristics, etc.), established visualisation techniques (e.g. overview and detail, filtering, etc.), and lead them to the design of usable visualisations. The foundations for such a methodology are discussed in chapter three.

Chapter 3: METHODOLOGIES, MODELS, HEURISTICS AND PATTERNS

Visualisation systems are software systems that incorporate a graphical user interface and a graphical view of the data. Therefore, before proposing a visualisation design methodology it is necessary to study existing methodologies from related disciplines. This chapter describes some of the more common methodologies used in software engineering and human-computer interaction (HCI), and attempts that have been made to combine them. Models for the visualisation process and the role of heuristics in visualisation design are also reviewed, as is the use of patterns in software and user interface design. The final section briefly discusses the benefits of the approaches listed above, the gaps that remain, and how this leads to the proposed methodology.

3.1 Overview

There are several reasons why using a formal methodology is more beneficial than using an *ad hoc* or intuitive approach to visualisation development. Some of the more obvious benefits include:

- The majority of the user requirements will have been determined and understood at the start of the development process. This will help to provide the users with what they want rather than what the developer thinks they want. It also reduces the chances of having to make costly alterations in the later stages of development.
- Effective ‘tried and tested’ techniques can be used rather than inappropriate or untested techniques. This will hopefully lead to a more effective and more usable design.
- Developers can focus on the content of the visualisation rather than trying to determine what needs to be done at each stage because the key tasks at each stage will have already been identified. Tasks and requirements vital to the system are less likely to be forgotten.
- The designer is provided with a complete picture of the development process, which can then be used to assess progress.

These benefits will help visualisation designers to overcome some of the problems described in chapter 2.

3.2 Existing Methodologies

The field of visualisation covers a wide range of disciplines. Some of these disciplines, such as perceptual psychology, provide low-level principles that can help to produce effective designs. Others, such as computer graphics, provide mechanisms for implementation. Software engineering and human-computer interaction are two disciplines that play a major

role in visualisation design. Software engineering can provide methodologies for the development of software, and human-computer interaction, in particular usability engineering, can provide techniques that help the designer to develop a usable, user-centred system. Methodologies and techniques from these disciplines need to be reviewed before a methodology specific to visualisation can be developed.

3.2.1 Software Engineering

Early software developers used an intuitive approach to the design, implementation, and testing of software. Clients would vaguely state requirements and software developers would provide what they thought was wanted. Software ‘evolved’ to fit the requirements in an *ad hoc* way rather than the requirements being clearly defined at the start of the development process and the software being designed to fulfil them. The final result was often less than satisfactory. The software failed to meet the needs of the users, developers had to continually make changes, software modules could not be easily integrated and overall system stability was low.

The Software Development Life Cycle was developed to overcome some of these problems. This more formal approach helps software developers to understand the user requirements, design robust maintainable code, and focus on what needs to be done at each stage of a project. One of the key features of this model is that it represents an iterative process. This is vital to software development since the requirements and specification may change, or inadequacies in the design may be identified during development.

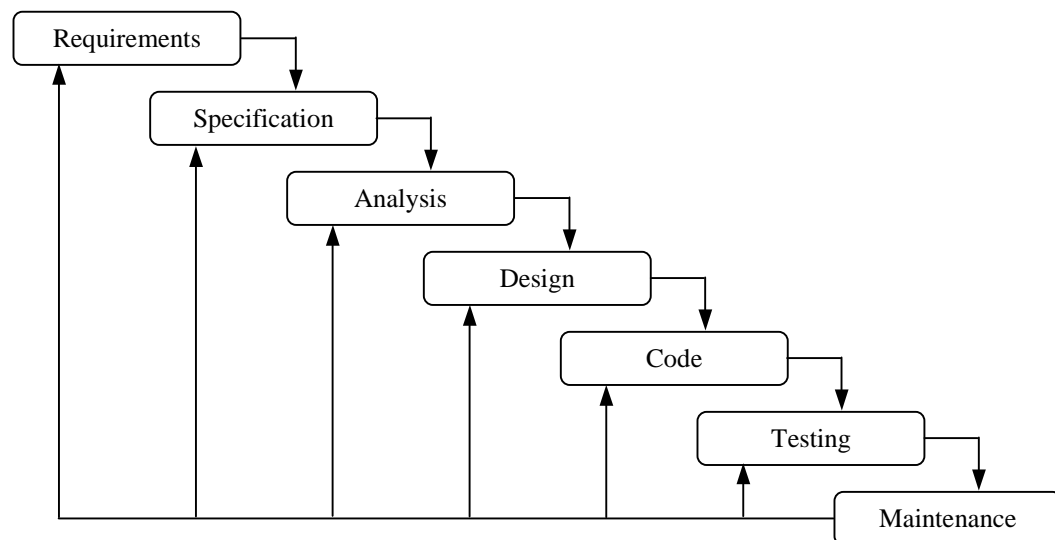


Figure 3.1: Software Development Life Cycle.

It should be noted that originally software development focused on the data required by the system, i.e. *data-centric*, or the functionality the system needed to provide, i.e. *process-*

3.2 Existing Methodologies

centric. More recently, these ideas have been combined to form the *object-oriented* approach, which encapsulates both data and functionality into objects. This approach emphasises code reuse, extensibility and maintainability.

A number of software engineering methodologies exist such as the Spiral Model, the Fusion Methodology, the Star Model, Reuse Model, etc.; too many to be reviewed in this thesis. Although many of these methodologies may be similar, each is tailored to meet specific requirements. Some emphasise testing, others are more business oriented, some focus on the user, and so on. If, for example, resources are limited and time is a critical factor then using the Spiral Model could be beneficial. This is because the Spiral Model focuses on developing high priority features first and uses the risk analysis phase to determine if another cycle is possible given the time and resource constraints. If a safety critical system is required then a methodology such as that proposed by Pfleeger (1987), which has three testing stages, may be suitable since the tolerance and stability of a safety-critical system is vital.

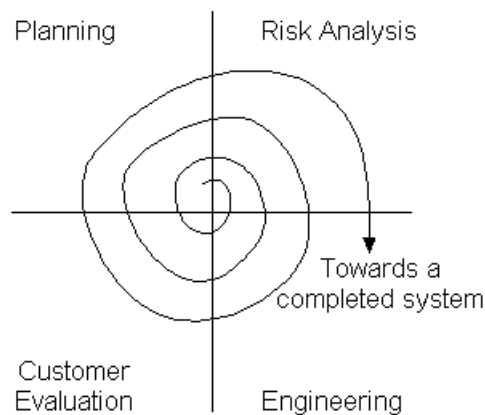
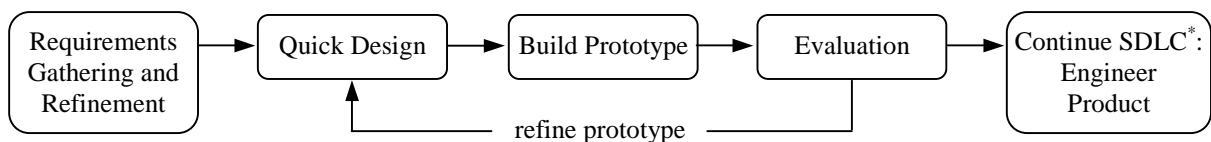


Figure 3.2: Spiral Model

Each of the methodologies mentioned so far could be considered *structured analysis and design methodologies* since they are each characterised by a number of stages that must be completed in sequence, although backtracking to previous stages is permitted. *Rapid prototyping* is a technique that can be embedded into these methodologies or used in isolation to gain a clearer understanding of the user requirements. The process of rapid prototyping, which involves quickly designing and building prototype systems that are then evaluated with users, can be seen in figure 3.3.



* Software Development Life Cycle

Figure 3.3: Rapid Prototyping

Software applications consist of two main parts, the code that the software engineer writes and the visual interface seen and interacted with by the user. Although these two parts are highly coupled, software engineers are rarely trained in interface design or evaluation. They are also expert computer users even though they may not be experts in the domain for which they are writing software. These factors can lead to interfaces which may seem intuitive to the software engineers but which are confusing and difficult to use for the user. Therefore, software engineers are not ideal user interface designers. A discipline that specialises in designing user interfaces and evaluating their usability is human-computer interaction.

3.2.2 *Human-Computer Interaction*

Human-computer interaction methods help to capture user requirements and user tasks, design and evaluate interfaces, and model the user. A number of specific techniques have been developed for each of these areas but only a generic description of each will be presented here.

3.2.2.1 *Capturing User Requirements*

The study of the tasks carried out by the user and how those tasks are achieved is known as *task analysis*. Performing a task analysis is an effective means of capturing user requirements. Task analysis techniques usually consist of identifying the objects that form part of the task and the actions that are applied to those objects. Observation, interviews, questionnaires, and analysis of existing systems can all be used to identify the objects and actions.

Different task analysis techniques emphasise different aspects of the task. For example, Dix et al. (1998) describe *hierarchical task analysis* as the decomposition of a task into subtasks, focusing in particular on the order and under what conditions the subtasks are performed. This is in contrast to *entity-relationship* based techniques, which focus on the relationships between objects used in the task.

Once the tasks have been identified, decomposed, and structured, they can be categorised by criteria such as task frequency, importance, complexity, etc. This means the software engineers can get a good understanding of the application they are trying to develop and allows the HCI specialists to design an effective user interface. Participants from both disciplines will be able to focus on the needs of the user.

3.2.2.2 *Cognitive and User Modelling*

It is obvious that the users play an important role in the development of any system. For this reason HCI specialists have attempted to model user behaviour, classify user types, and

identify different user needs.

Cognitive modelling attempts to understand the reasoning behind the actions taken as a user works through a task. The task analysis techniques mentioned previously can prove useful in achieving this aim.

Developing profiles of users such as their level of expertise (e.g. novice, intermediate, expert), preferences, physical disabilities (e.g. colour blindness), and so on, is known as *user modelling*. These models can be constructed by interviewing users, giving them questionnaires, observing them in the workplace, etc. Modelling the user in this way can help to determine the level of support the interface must provide. For example, expert users are more likely to prefer an interface that allows them to quickly issue a series of commands e.g. a command line interface, whereas a novice may prefer an interface that visually guides them through a process e.g. a wizard. Dix et al. (1998) define several different classes of user model including:

- Cognitive models that represent users of interactive systems.
- Hierarchical models that represent a user's task and goal structure.
- Linguistic models that represent the user-system grammar.
- Physical and device models that represent human motor skills.

User models such as these may assist the user interface designer in the development of usable systems, although they can be problematic and difficult to use.

3.2.2.3 User Interface Evaluation

HCI specialists measure the effectiveness of a user interface in terms of its *usability*. It is important to note the difference between usability and *usefulness*. Wesson (1999a) defines usefulness in terms of whether or not the system can support the user in achieving their goal. This is then split into *utility* and *usability*. Utility refers to the functionality of the system whereas usability is concerned with how well the user can access the functionality. So a poorly designed system may be usable but not useful, or alternatively, useful but difficult to use.

Usability is further split into several sub-categories, or *usability factors*, that can be measured when evaluating a software system. Shneiderman (1998) defines these as:

- *Time to learn*. How long does it take for typical members of the user community to learn how to use the commands relevant to a set of tasks?
- *Speed of performance*. How long does it take to carry out the benchmark tasks?
- *Rate of errors by users*. How many and what kinds of errors do people make in carrying out the benchmark tasks? Although time to make and correct errors might be incorporated

into the speed of performance, error handling is such a critical component of system usage that it deserves extensive study.

- *Retention over time.* How well do users maintain their knowledge after an hour, a day, or a week? Retention may be linked closely to time to learn, and frequency of use plays an important role.
- *Subjective satisfaction.* How much did users like using various aspects of the system? The answer can be ascertained by interviews or by written surveys that include satisfaction scales and space for free-form comments.

A number of techniques, such as *cognitive walkthroughs*, *user testing*, *focus groups*, etc., can be used to measure the usability factors. Analysing these measurements, HCI specialists have determined sets of guidelines that can help them to develop usable systems. Nielsen and Molich (1990) and Nielsen (1994) define ten usability heuristics that cover a variety of factors including system status, consistency, error prevention, error recovery, system flexibility, and help facilities. Shneiderman (1998) defines eight golden rules for interface design that are similar to the heuristics defined by Nielsen. In this case ‘heuristics’, ‘guidelines’, and ‘rules’ are synonymous.

Heuristics can be used to identify the usability strengths of different interface designs. Nielsen (1994) defines an *heuristic evaluation* as the examination of each dialogue element in an application to see if follows established usability principles (i.e. the heuristics). Nielsen (1994) also defines a variant of the heuristic evaluation, known as an *heuristic estimation*, in which HCI specialists estimate in quantitative terms the relative usability of different designs. The results of these evaluations can then be used to improve the usability of the final product.

3.2.3 Software Engineering meets Human-Computer Interaction

Clearly software engineers have the technical skills and methodologies to build efficient and reliable systems. Unfortunately interface design tends to be given a low priority. HCI specialists are interested in producing interfaces with a high degree of usability and have developed techniques that can help to achieve this goal. Recognising the need to address usability issues during software development, Nielsen (1992) proposed the *usability engineering life cycle*. This model describes a set of methods that can be incorporated into the development process to help ensure the usability of the final product. The elements that constitute this model are summarised in table 3.1.

It is important to note that many of the elements in the usability engineering life cycle are iterative and overlap and that due to resource constraints it may not be possible to apply all the methods listed. Therefore, Nielsen (1992) recommends ‘iterative and participatory design’ and ‘prototyping and empirical testing’ as part of any usability engineering process.

3.2 Existing Methodologies

1	Know the user (characteristics, current and desired tasks, functional analysis, user and job evolution).
2	Competitive analysis (examining existing products).
3	Setting usability goals (financial impact analysis, prioritising design goals).
4	Parallel design (several initial designs by independent teams).
5	Participatory design (actively involving users in the design process).
6	Coordinated design of the total interface (consistency within and across products).
7	Apply guidelines and heuristic analysis (style guides, standards, and guidelines).
8	Prototyping.
9	Empirical testing (user tests).
10	Iterative design (improve prototypes, capture design rationale).
11	Collect feedback from field use.

Table 3.1: Usability Engineering Life Cycle.

Wesson (1999a) believes that existing user-centred design approaches such as the Method for Usability Engineering (MUSE) (Lim and Long, 1995), the TRIDENT methodology (Vanderdonckt and Zucchinetti, 1995), and the Graphical User Interface Design and Evaluation (GUIDE) method (Redmond-Pyle and Moore, 1995), do not adequately integrate HCI techniques into the entire software development life cycle. She proposes the UNION methodology. UNION combines both software engineering and HCI methods to form an object-oriented, user-centred, approach to the software development life cycle. Briefly, the UNION methodology consists of three main phases:

- *Analysis*: is concerned with the collection and specification of requirements. A problem statement defines the tasks that need to be supported, the users of the system, and the level of support required. In addition, existing systems are analysed to determine the problems that need to be resolved. A task analysis is performed in order to construct a *task model*. Object oriented methods such as Class Responsibility Collaborator (CRC) modelling are then used to identify the main classes in the system. An initial analysis of the user interface and interaction between the interface and problem domain is also conducted.
- *Design*: refines and extends the requirements found in the first phase to derive a complete design specification embodied in three models:
 - *Problem Domain Object Model*: represents a model of the conceptual domain or business objects.
 - *UI Object Model*: represents a model of the interface input and output objects.
 - *Dynamic Model*: represents a model of the interaction and collaboration between the problem domain and the interface objects. It describes the changes in state of both the PD Object Model and the UI Object Model.

An iterative object-oriented user interface design approach is used together with macro and micro-level user interface design heuristics to prototype the user interface design. This object-oriented interface design approach increases the likelihood of producing interfaces that represent and support the kinds of objects and actions found in the users' real-world domain.

3.2 Existing Methodologies

- *Implementation*: maps the design specification into database schemas, objects, etc. This phase also includes a detailed system evaluation, which is used to refine the design.

It is evident that each phase of the UNION methodology can be associated with different elements of the usability engineering life cycle. For example, the analysis phase clearly relates to ‘know the user’ and ‘competitive analysis’, the design phase makes use of ‘heuristics’ and ‘prototyping’, and the implementation phase involves ‘empirical testing’ and ‘iterative design’. Thus, as well as using an object-oriented approach to interface design, the UNION methodology also includes some of Nielsen’s (1992) usability recommendations.

3.2.4 Generic Design Methodology

Looking at the methodologies presented in sections 3.2.1, 3.2.2, and 3.2.3, it is evident that each shares a number of common stages. However, each methodology also uses specific techniques to accomplish these stages. Identifying these common stages and removing specific details leads to a *generic design methodology* such as that shown in figure 3.4. An example of how this relates to the usability engineering life cycle is shown in table 3.2.

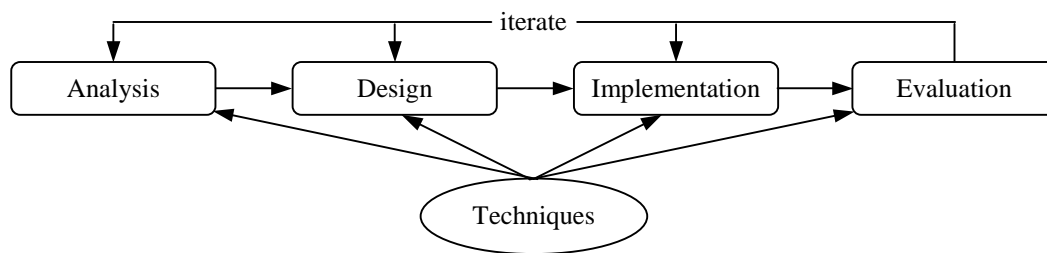


Figure 3.4: Generic design methodology.

		Description	Example Techniques
Analysis	1	Know the user (characteristics, current and desired tasks, functional analysis, user and job evolution).	Task analysis. User modelling.
	2	Competitive analysis.	Examine existing products.
	3	Setting usability goals.	Financial impact analysis. Prioritising design goals.
Design	4,5,6,7	User interface design.	Parallel design. Participatory design. Coordinated design of the total interface. Apply guidelines and heuristic analysis.
Implementation	8	Prototyping.	
Evaluation	9	Empirical testing.	Cognitive walkthroughs. Usability inspection. Questionnaires.
Analysis Design Implementation	10	Iterative design (improve prototypes, capture design rationale).	
Evaluation	11	Collect feedback from field use.	Questionnaires, Interviews, Observation.

Table 3.2: How the generic methodology relates to the usability engineering life cycle.

Analysing the methodologies in more detail it can be seen that each is tailored to the *value-system* used within the particular discipline or domain to which the methodology is applied. For example, software development values robust code that can easily be extended and maintained, but when applied to safety critical systems, the robustness of the software is of more worth than its extensibility. In HCI the value system is based on usability, which is then broken down into a number of specific factors.

Therefore, although each specific methodology includes the stages in the generic methodology the techniques that are used to accomplish each stage are different. In this way a methodology can emphasise the stages that are the most important to the value-system being used. For example, the usability value-system used in HCI leads to techniques such as interviews, questionnaires, cognitive walkthroughs, etc., during the evaluation stage, whereas for software engineering robust code is more desirable so techniques such as black-box testing, dry runs, code walkthroughs, unit-testing, etc., will be used.

The generic design methodology will most likely form the basis for a visualisation methodology since it embodies the elements common to most design methodologies. As noted at the beginning of the chapter, visualisation systems are software systems that incorporate a graphical user interface and a graphical view of the data, therefore the value-system used for visualisation design will include both the software and HCI value systems. In addition visualisation also values perceptual effectiveness. Also because visualisations are software systems and the usability of a visualisation, as with other interaction-based systems, is a high priority, both the software development life cycle and the usability engineering life cycle will have a significant influence.

3.3 Visualisation Reference Models

The basic components of any visualisation are the data, the representation of that data, i.e. the view, and the user interaction with the view to explore the data. At its simplest the visualisation process consists of transforming the data to some visual representation, which can then be manipulated via interaction. This very basic visualisation process is shown in figure 3.5.

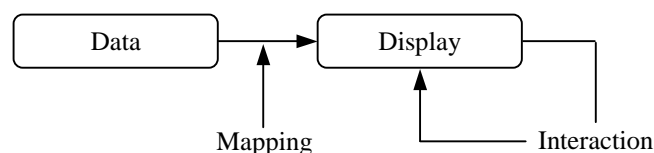


Figure 3.5: Basic visualisation process.

3.3 Visualisation Reference Models

The transformation from data to display is essentially a mapping from data attributes to visual features. The designer must decide what visual objects to use to represent data concepts and which features of those visual objects to represent data attributes. Together, this defines what Chuah (2000) calls a *graphical scene* and is equivalent to the *visual structures* defined by Card et al. (1999). However, at this point the graphical scene is still an abstract concept that requires additional transformations before it can be rendered in the display. This slightly more detailed visualisation process is shown in figure 3.6.

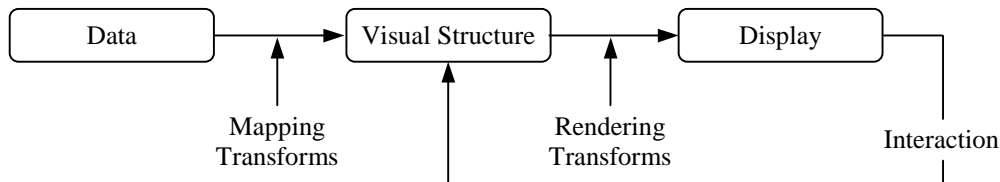


Figure 3.6: Visualisation process.

Several other researchers such as Haber and McNabb (1990), Csinger (1992), Robertson and De Ferrari (1994) have also developed visualisation reference models. Although none of these models are identical and none of them use exactly the same terminology, they all share the same underlying process. For example, Robertson and De Ferrari's (1994) model shown in figure 3.7 may look very different from the model shown in figure 3.6. However, on further analysis it can be seen that the data is transformed into a visualisation design and the design is rendered in a display. The display is then interacted with by the user, which then alters the visual structure or render process. In other words Robertson and De Ferrari's (1994) model is simply an extended version of that shown in figure 3.6.

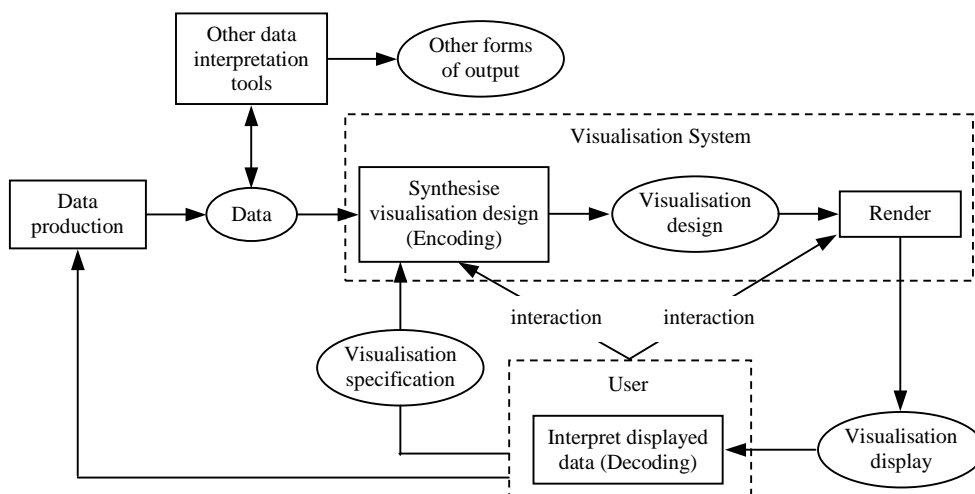


Figure 3.7: Integrated visualisation model.
(Modified from Robertson and De Ferrari (1994))

3.3 Visualisation Reference Models

Analysing and combining two of the more recent models presented by Card et al. (1999) and Chuah (2000) we propose the more complete model shown in figure 3.8. This can be described as follows:

- Raw data is converted to structured data via a set of *raw data transforms*. In essence, the raw data is organised into coherent units that represent data concepts that may then be stored, for example, in data tables, feature vectors, etc.
- This structured data then undergoes a set of *data transforms*, which are used to calculate derived results (e.g. add, subtract, etc.), or summarise attribute values (e.g. mean, etc.), or filter data items, or compute meta-data (e.g. count the number of times a data value appears, etc.), or to alter the order of the data (e.g. sort, etc.).
- This derived data is then transformed into a set of *visual structures* using *mapping transforms*. As stated previously the mapping transforms encode the data into some abstract visual form.
- The visual structures are then viewed using *rendering transforms*, i.e. the abstract visual form is transferred to some output media such as the screen or paper. In addition to mapping transforms, *graphical transforms* may also be applied to the visual structures. Graphical transforms are used to change the appearance of objects in the graphical scene e.g. colour, position, shape, size, etc.
- Finally user *interaction* can be used to modify the transforms at each stage. Using interaction the user can manipulate the data content, specify what the derived data should be, which subsets of the data should be viewed, which parts of the graphical scene should be rendered, and affect the appearance of objects in the scene.

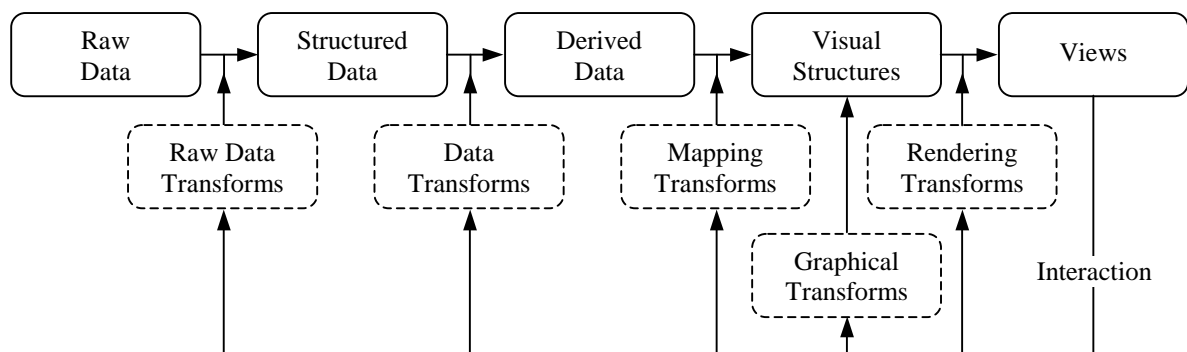


Figure 3.8: Extended visualisation process.

Understanding the visualisation process is important. It helps the designer to break-up the large and complex task of visualisation design into smaller more manageable components. The designer can then focus on the requirements of each component in turn. Although these models highlight the visualisation process they do not aid the designer in choosing the most appropriate visual structures, transforms that affect those structures, or interaction mechanisms.

3.4 Automatic Visualisation Generators

Having studied the principles of visual perception, gained experience developing visualisations for various domains, and following the basic visualisation process described in section 3.3, visualisation designers began research into systems that could automatically generate visualisation designs. The purpose of these automatic visualisation generators is to remove much of the mundane work associated with designing conventional chart-based presentations such as bar charts, scatter plots, and so on. This section discusses the history, evolution and limitations of automatic visualisation generators and how there is still a need for a methodology that allows human designers to develop effective novel visualisations.

3.4.1 History and Evolution

Mackinlay (1986) was one of the first researchers to develop a tool, called APT, which would automatically generate presentations. Mackinlay’s approach consisted of three parts. Firstly *expressiveness* criteria determine which presentation type e.g. bar chart, scatter plot, etc., can be used to represent the data. Having determined applicable presentation types a set of *effectiveness* criteria are used to select the most appropriate. Mackinlay mapped and ranked visual features to quantitative, ordinal and nominal data types based on the fact that certain perceptual tasks can be accomplished more accurately than others can. This ranking is shown in figure 3.9. This formed the basis for the effectiveness criteria. Finally *composition algebra* is used to generate composite designs.

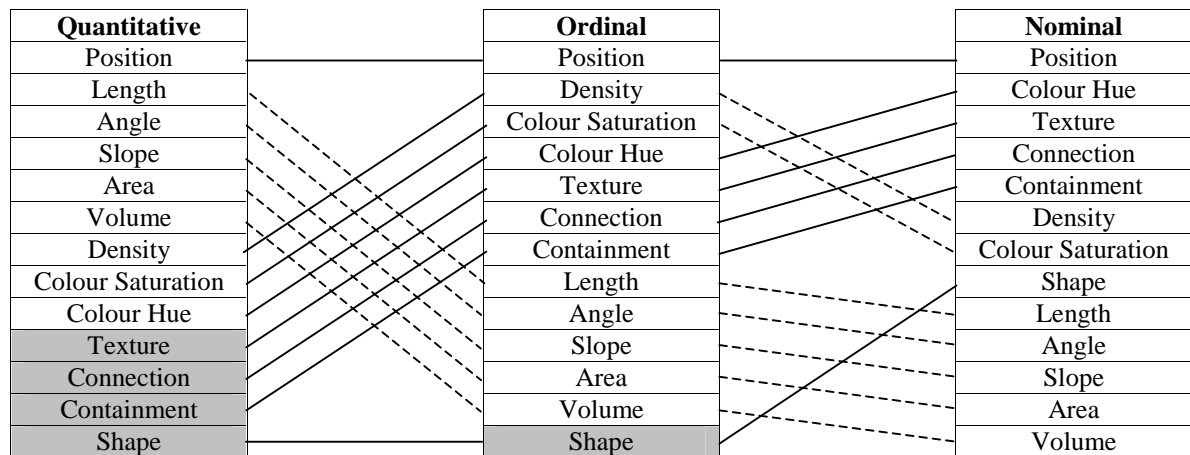


Figure 3.9: Mackinlay’s ranking of visual features to data type.

Note: The features in the grey boxes are not relevant to these types of data.

Casner (1991) noted that Mackinlay’s APT system designed presentations based on the data types only and did not consider the tasks the presentation would be used for. Casner developed the BOZ system, which extended Mackinlay’s work by analysing the task as well as the data. This task-analytic approach allows BOZ to generate different presentations to

support different tasks using the same data. Briefly, the technique used by BOZ is to map a set of *logical operators* defined by the task into a set of equivalent *perceptual operators* each of which is associated with a set of applicable visual features. The most appropriate visual features, based on Mackinlay's ranking, that can support the data and the task are then used to generate the presentation.

Senay and Ignatius (1994) have also extended Mackinlay's work by concentrating on the effective composition of three-dimensional charts.

The AVID system developed by Chuah (2000) is similar to Mackinlay's (1986) APT tool. However, as well as considering *mapping transforms* equivalent to Mackinlay's expressive and effectiveness criteria, Chuah also includes several other transforms. These are described in detail in sections 3.3 and 4.1.4. In terms of new automatic visualisation generator developments the most significant of these is the *data transform*, which offloads much of the cognitive load from the user to the computer. This is achieved by determining when it is more appropriate, given numerous constraints, to present the results of calculations or summaries rather than the data involved in the calculation. For example, imagine the task is to determine the combined weight of two people represented as bars in a bar chart. It requires less cognitive and perceptual load to determine the result from a single bar that represents the total weight than it does from two individual bars adjacent to each other. Therefore, it is appropriate to let the computer perform the calculation and only show the result. Several factors such as task specificity, task expressiveness, and so on, are considered in the AVID system.

Lange (1995), Zhou and Feiner (1996, 1998), Healey et al. (1998) and Salisbury (2001) have also successfully developed automatic visualisation generators.

3.4.2 Problems with Automatic Visualisation Generators

The main problem with the existing automatic visualisation generators is that the designs produced are all chart-based, e.g. bar charts, scatter plots, etc. This imposes a number of limitations on the structure, range, quantity, and dimensionality of the data that automatic visualisation generators can handle.

Considering the data structure it can be seen that chart-based presentations are only capable of supporting relational data of the type typically found in relational databases. Other data structures such as 1-dimensional text, tree structures, network structures, etc., cannot be represented using any of the standard charts. With the exception of Salisbury's (2001) system even map-based data is not supported. This lack of support for a wide range of data structures means that automatic visualisation generators fail to satisfy the needs of a large proportion of data intense domains.

Charts can be very good at presenting two or three dimensions of a small quantity of data. However visualisations are often required to support large quantities of multidimensional data. Unfortunately, many existing automatic visualisation generators do not generate solutions that use multiple linked views, interaction, and animation, and without employing at least some of these techniques they are unlikely to satisfy the requirements of many domains.

It should be noted that the limitation of automatic visualisation generators to chart-based presentations does not mean they are of no use. They establish basic frameworks and theories and help to discover new rules and indicate directions for future research. Chart-based presentations are well understood, well defined, commonly used, and have specific rules for construction and interpretation. The exactness of these rules is what makes automatic visualisation generators possible.

Automatic visualisation generators implicitly use methodologies to design and evaluate potential visualisations. The problem with the methodologies used in automatic visualisation generators is that they rely on the fact that a computer-based system can cover a large portion of the visualisation design space very rapidly. They can systematically follow each path in the design space and discard designs that fail to meet predefined criteria. Human designers cannot use this approach due to its complexity and repetitiveness. One exception to this is the automatic visualisation generator developed by Salisbury (2001). She developed a methodology that could be used by both the system and human designers. The methodology relies on a series of tables, based on empirical evidence, that combine and rate various factors such as user task, base visualisation type, visualisation expressiveness, encoding methods, etc. By following a very simple process and using the tables, the designer can determine the most appropriate visualisation design. The problem with Salisbury's methodology is that it does not cover all aspects of the development process (e.g. analysis, evaluation, etc.), fails to address certain design elements (e.g. interaction, composition, etc.), and can only be applied to chart and map-based visualisations, although it could potentially be extended to include other types of visualisation.

The main argument here is that current automatic visualisation generators are restricted to a set of pre-defined designs and therefore cannot generate truly novel visualisations. They are also limited to low quantity, low dimensionality, relational data based domains. These are serious limitations that make automatic visualisation generators inappropriate in a wide variety of high-density, complex domains, where effective novel visualisations are essential. It is doubtful that an algorithm will ever be developed that is capable of inventing the many different designs human designers are capable of. Therefore, human designers still need guidance to develop an effective and usable visualisation design.

3.5 *Heuristics*

Based on their analysis of user behaviour and empirical evidence gathered testing interfaces with users, HCI specialists have developed a number of heuristics (or guidelines) for interface design. An example of three usability heuristics as defined by Nielsen (1994) are shown in table 3.3.

Heuristic Title	Description
Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback, within reasonable time.
Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Error prevention	Even better than good error messages is a careful design that prevents a problem from occurring in the first place.

Table 3.3: Example usability heuristics.

In a similar fashion visualisation researchers have started to develop heuristics for visualisation design. These heuristics are based on their experience gained over several years designing, implementing, and evaluating visualisations. These heuristics differ slightly to those proposed by Nielsen (1994) as they include a more detailed rationale.

The heuristics developed cover a wide range of visualisation issues such as those described in chapter 2, including analysing the tasks, data structures, 2D vs. 3D displays, interaction, multiple view layout and so on. Incorporating these heuristics into a methodology provides the designer with knowledge of features that are desirable in a visualisation. Details about how this has been achieved can be found in section 4.2.

3.6 *Patterns*

Patterns are similar to heuristics in that they provide the designer with ‘good’ design knowledge. However, patterns include additional information that tells the designer when to apply the knowledge and why it works. This section describes what a pattern is, the benefit of patterns, and how they can be used throughout the design process.

3.6.1 *Brief history*

Alexander et al. (1977) proposed the idea of patterns as a method for capturing the knowledge underlying successful solutions to recurring architectural design problems. There have been various pattern definitions but each of these shares a common theme, which is that patterns describe a proven solution to a recurring problem in a given context. This definition will be expanded upon later.

The software community adopted the idea of patterns as a means of documenting and sharing solutions to recurring software design problems. More recently, the HCI community has also adopted the idea of patterns as a method of capturing user interface design knowledge.

Interestingly, Borchers (2000a) claims that the use of patterns in HCI is closer to the way in which they are used in architecture than the way they are used in software engineering because HCI and architecture share many of the same goals. Borchers argues that both architects and HCI specialists design artefacts that the user is expected to interact with. In contrast the underlying code designed by software engineers should never be seen by the user. The similarities between HCI and architecture may mean that HCI can benefit from using patterns in a way similar to architecture, possibly more so than software engineering.

3.6.2 HCI Patterns

A number of researchers have developed HCI patterns. However none of these researchers have used exactly the same pattern format. Despite the differences most of the pattern formats proposed include a pattern *name* by which the pattern can be referred, a *problem* statement, a *context* in which the problem occurs, *forces* that may constrain a solution, a proven *solution* and *examples* of the solution in action, and finally links to other related patterns that may be used to refine the solution. It should be noted that since there is no standard HCI pattern format some of these attributes might be considered optional. More formally Borchers (2000b) defines these attributes as follows:

- *Name* or *Title*: Helps to refer to the pattern's central idea quickly, and builds a vocabulary for communication within a team or design community.
- *Context*: The design situations in which the pattern can be used.
- *Problem*: States what the major issue is that the pattern addresses.
- *Forces*: Elaborate the problem and context statements.
- *Solution*: Generalises from a set of proven examples to balance the forces for the given design context. It is not simply prescriptive, but generic so that it can generate a solution when it is applied to concrete problem situations of the form specified by the context.
- *Examples*: Show existing situations in which the problem at hand can be (or has been) encountered, and how it has been solved in those situations.
- *References* or *Related Patterns*: Show what higher-level patterns this pattern contributes to and what lower-level patterns can be applied after this pattern has been used.

What is interesting here is that patterns provide a reusable solution. This means that a designer can identify a pattern based on the similarity between its problem statement and context to the current situation. They can then adapt the solution to fit the specifics of the current situation. This relates to Tidwell's (1999) definition of patterns as describing the

“invariant qualities” of a set of good solutions to a common design problem within a certain context.

The examples used in patterns help to clarify to the designer exactly how the pattern has been applied in the past. Fincher (1999) refers to this as sensitising the reader to the application of the pattern. To strengthen the validity of a pattern it should preferably refer to several examples. This helps to show that it encapsulates a recurring solution to a given problem. Only being able to find one example may imply that it is not a pattern at all since this contradicts the definition of a pattern.

Unfortunately, as noted by Fincher (1999) and Borchers (2000a), HCI often involves some temporal aspect. The fact that most patterns are currently only presented in document form sometimes makes it difficult to include a comprehensive example. One solution is to use storyboards, other media, such as HTML pages, may use animation or video clips.

The original architectural patterns were organised by size ranging from patterns dealing with entire neighbourhoods to patterns dealing with individual rooms. Patterns at each level point to more specific patterns that may be used after their application. Patterns may also point to more general patterns to which they might contribute. This forms a hierarchical collection of patterns known as a *pattern language*. This concept has led HCI pattern writers to include an attribute in the pattern format that refers to related patterns. Borchers (2000b) notes that the use of a pattern language allows designers to find relevant patterns quickly and refine the suggested solution.

Borchers (2000a), Fincher (1999), Welie and Veer (2000), and Tidwell (1999) have all identified the fact that different disciplines have different value systems and that this should be embodied in the pattern format. For example, in architecture a comfortable or enjoyable living environment is said to have a high value. In software engineering value relates to code reuse, maintainability, and extensibility. Borchers (2000a) describes HCI value in terms of user interfaces being “intuitive” or “transparent” to the user. It is generally agreed by researchers such as Wesson (1999a), Welie and Veer (2000), Granlund and Lafrenière (1999), and Granlund et al. (2001) that value in HCI relates to *usability*. This has led Welie and Veer (2000) and Welie et al. (2000) to develop a pattern format specifically for user interface design that focuses on solutions that improve system usability. This *usability-based* pattern format includes additional attributes such as *usability impact*, *rationale*, and *usability principle*.

The HCI community, having adopted the idea of patterns, have developed their own definition tailored to user interface design. Griffiths et al. (1999) have proposed a user-centred definition

3.6 Patterns

of HCI design pattern languages in terms of what they can be used for. This definition is as follows:

“The goals of an HCI design pattern language are to share successful HCI design solutions among HCI professionals, and to provide a common language for HCI design to anyone involved in the design, development, evaluation, or use of interactive systems.”

Although no standard format for HCI design patterns has been agreed upon, the definition above encompasses the general goals that most HCI pattern writers try to achieve.

To highlight the differences and similarities that different researchers use when describing patterns, two versions of the same pattern are presented in table 3.4.

	Tidwell (1999)	Griffiths et al. (1999)
<i>Title</i>	Short Description	Description at your fingertips
<i>Problem</i>	How should the artefact present additional content, in the form of clarifying data or explanations of possible actions, to the users that need it?	You are putting interactive objects on a dynamic medium such as a screen and you want to provide various levels of context sensitive help supporting uninterrupted tasks. Extensive explanations tend to clutter the interface but users may need such help. They do not want to leave the context of their current task, and experts may not want to see the help at all.
<i>Context</i>	The artefact contains a visual pointer, or “virtual fingertip” (mouse or pen point, for instance) that is the focal point for the user’s interaction with the artefact.	
<i>Forces</i>	<ul style="list-style-type: none"> • A short explanation may be all the user needs or wants; something long will be overkill. • Users generally don’t want to leave the artefact and go somewhere else for help, such as a manual; this usually breaks one’s concentration and costs too much time. • There isn’t room to put static descriptive text into the artefact, or visual elegance precludes doing so. • The descriptive text might be useless most of the time, and may become irritating if it is static or hard to turn off. 	
<i>Solution</i>	Show a short (one sentence or shorter) description of a thing, in close spatial and/or temporal proximity to the thing itself. Allow the user to turn it on and off, especially if the description obscures other things or is otherwise irritating; alternatively, don’t show it without some deliberate user action on an item-by-item basis, such as pressing a key or hovering over the item for a certain length of time.	Provide a short description of the object either close to it or in a fixed position. Let users turn it on and off or only provide it on some explicit user action (e.g., hovering).

Continued on next page...

	Tidwell (1999)	Griffiths et al. (1999)
<i>Examples</i>	<ul style="list-style-type: none"> • Windows tool tips. • Mac bubble-help. • Status bar help. 	In the Mac OS, a small balloon of textual help appears when the user turns on this feature and moves the mouse over an object. In Windows Tool Tips the same thing happens if the mouse hovers over an object. In Netscape, the URL of a link is displayed in a fixed position at the bottom of the screen if the cursor is moved over it. In a voice mailbox, options are explained if the user waits for a while.

Table 3.4: Pattern comparison.

It can be seen in table 3.4 that although the two patterns describe the same thing, their presentation is quite different. The context and forces in the pattern by Griffiths et al. (1999) have all been included in the problem statement whereas Tidwell (1999) separates them. With no clear pattern standard, differences such as this could cause confusion, especially for novice designers.

3.6.3 Benefits of Patterns

At first glance it may seem as though patterns are just *structured guidelines*, i.e. guidelines presented in the pattern format. However, patterns have a number of advantages over guidelines. Welie et al. (2000) list the following problems with guidelines:

- Usually guidelines are numerous and it is difficult to select the guidelines that apply to a particular design problem.
- Guidelines are usually very compact but their validity or appropriateness always depends on a *context*.
- They do not tell the designer *when, how, and why* the solution should be applied.
- Guidelines do not include a *rationale*.
- Guidelines are often too simplistic or too abstract.
- Guidelines can be difficult to select.
- Guidelines can be difficult to interpret.
- Guidelines can be conflicting.
- Guidelines often have authority issues concerning their validity.

Welie et al. (2000) and Griffiths and Pemberton (2001) emphasise three important differences between patterns and guidelines. The first difference is that patterns record all the information that would normally be required to use a guideline. The context, problem, and solution are all made explicit and a rationale is provided explaining why the solution works. A pattern might also incorporate several guidelines. The second difference is that patterns must provide a *proven* solution and examples of the solution in practice. Finally, patterns can be organised

into pattern languages allowing the designer to quickly navigate and refine solutions. This hierarchical organisation is not possible with guidelines.

Borchers (2000b) summarises one of the advantages of patterns quite succinctly stating, “design guidelines are *descriptive*, and merely state desirable features of a ‘good’ interactive system, patterns are *constructive*, they suggest how a problem can be solved”.

It may be that the differences between patterns and guidelines do not matter, what is more important is the way the design knowledge is captured and delivered to the designer, be it a pattern, or a guideline in pattern form.

Perhaps one of the most significant benefits of patterns is their ability to allow both HCI experts and users to communicate using a common language. This is possible because patterns can be referred to by their title and the use of examples allows non-experts to understand what could potentially be a complex idea. The examples also serve as a means of showing the user a potential design. Allowing HCI experts and users to communicate more effectively means users can be more involved in the design process, which then leads to a more user-centred design. This idea of cross-discipline communication can be extended to include domain experts, project managers, software engineers, HCI specialists, users, and any other members of a project.

3.6.4 Pattern Approach to Design

Wesson (1999b) proposes adapting her UNION methodology to take advantage of patterns. Part of the methodology includes an iterative user interface prototyping stage. Wesson suggests that the macro and micro-level user interface design heuristics could be replaced by a pattern language. This would then help the designer select the most appropriate interaction objects based on the underlying conceptual model of the problem domain.

Granlund and Lafrenière (1999) and Granlund et al. (2001) propose using patterns in all stages of the user interface design process. The stages in this process and the patterns that support each stage are shown in figure 3.10.

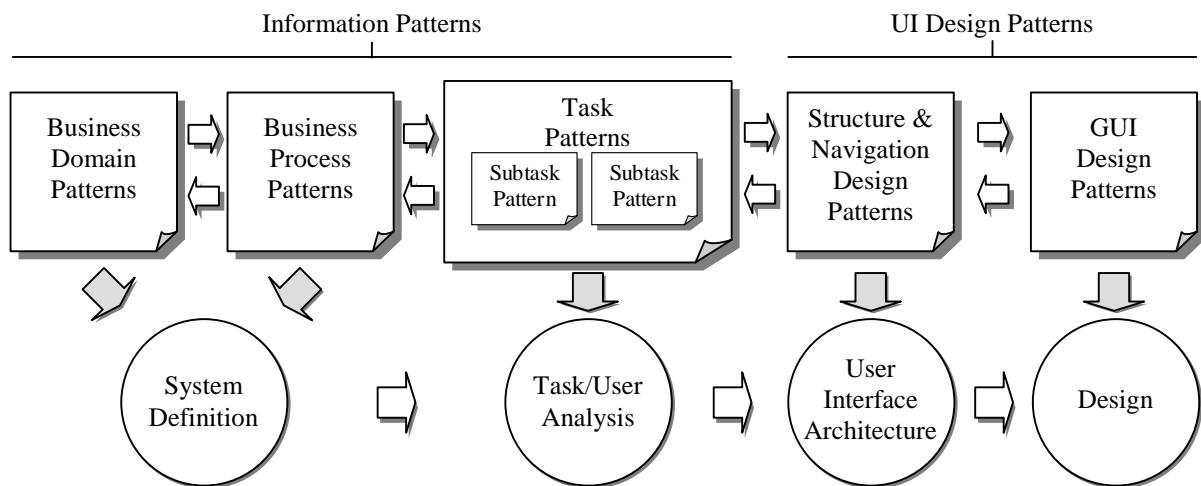


Figure 3.10: The Pattern Supported Approach Framework.

(Modified from Granlund et al. (2001))

Briefly, Granlund et al. (2001) describe each of the pattern types as follows:

- *Business Domain Patterns*: describe the type of business, its goals, plus the typical actors and business processes involved. They provide a starting point for initially defining the system design by pointing to relevant *Business Process* patterns and thereby to *Task* patterns. They help communicate the *System Vision*.
- *Business Process Patterns*: describe typical processes and actors involved in the delivery of services/goods in compliance with the business goals. They narrow the system definition and point to specific *Task* patterns to be considered.
- *Task Patterns*: are used for capturing and passing on knowledge about the task, typical users, and their work context from previous similar projects, and for suggesting an appropriate interaction design solution. They point to *Structure & Navigation Design* patterns that describe solutions that have proven suitable for the task type in previous designs.
- *Structure & Navigation Design Patterns*: describe ways to structure information and implement navigation in order to support the user's task. This design is based on information described in the *Task* patterns.
- *GUI Design Patterns*: document GUI design issues based upon information described in the *Task* patterns and *Structure & Navigation Design* patterns. They are based on the work of Tidwell (1999).

It is interesting to note that as part of this framework Granlund et al. (2001) have classified patterns into two broad categories. The domain-dependent *information patterns* capture background information, whereas the domain-independent *UI design patterns* are closer to existing HCI patterns. However, making this distinction does mean that pattern attributes may

3.6 Patterns

be interpreted in different ways or may not be included. For example, information patterns tend not to have a problem statement or solution. This is a novel use of the pattern concept but it is unclear if what has been defined can still be thought of as a pattern.

Borchers (2000a, 2000b) also advocates the use of patterns to model application domains. In addition he shows how a pattern language can be mapped to each stage in Nielsen's (1992) usability engineering life cycle. This can be seen in table 3.5.

	Description	Pattern Use
1	Know the user (characteristics, current and desired tasks, functional analysis, user and job evolution).	Extract application domain experience and concepts as pattern language.
2	Competitive analysis (examining existing products).	Generalise good UI solutions into HCI patterns.
3	Setting usability goals (financial impact analysis, prioritising design goals).	Use competing usability goals as forces in abstract HCI patterns.
4	Parallel design (several initial designs by independent teams).	Use general HCI design patterns (maybe from book) as common design guidelines for the teams.
5	Participatory design (actively involving users in the design process).	Application domain expert (user) and HCI designer exchange their pattern languages for better mutual understanding.
6	Coordinated design of the total interface (consistency within and across products).	Lower-level HCI design patterns, including project-relevant, concrete examples, communicate the common look and feel efficiently.
7	Apply guidelines and heuristic analysis (style guides, standards, and guidelines).	Patterns improve upon those formats because of their standard format, hierarchical networking, inclusion of examples, and discussion of problem context as well as solution.
8	Prototyping.	Software design patterns express the standards, components, and specific project ideas of the development team in a way better understandable by the HCI experts.
9	Empirical testing (user tests).	Problems discovered can be related to applicable patterns to solve the problem (vocabulary function).
10	Iterative design (improve prototypes, capture design rationale).	HCI and software patterns (constructive, unlike guidelines) inform designers about design options at each point, and help capturing the space of design options explored (structural design rationale) – possibly with anti-patterns for bad solutions.
11	Collect feedback from field use.	Use application domain pattern language again as common vocabulary between UI experts and users. Use feedback to strengthen successful HCI and software patterns, and to re-evaluate sub-optimal ones.

Table 3.5: Borchers' application of patterns to the usability engineering life cycle.

What can be seen in each of these approaches is that patterns can be an extremely useful mechanism for supporting combined software engineering and HCI methodologies. They can cover all aspects of the development life cycle, from user requirements and system definition, through to UI design and evaluation. Visualisation development also includes these and other stages. Consequently a visualisation methodology that was supported by patterns at each stage would seem appropriate.

3.7 Summary

Visualisation is a form of human-computer interaction. Visualisation consists of a view, or views, of the data and an interface that surrounds each view. This means that the lessons learnt in HCI and the HCI patterns developed for interface design can be applied to visualisation.

A visualisation system is also a software system. This implies that the software methodologies described in section 3.2.1 can also be applied. However, pure software approaches to design tend to be focused on the data and functionality of the system rather than on the user. HCI on the other hand focuses on the user. For this reason combined software engineering and HCI methodologies have been developed such as the UNION methodology described in section 3.2.3.

Combined methodologies emphasise the roles experts from different domains can play in the development of a software product. HCI specialists can work closely with users, defining user models, analysing tasks and developing prototype interfaces. Users can also provide feedback helping to refine initial designs. Together HCI specialists and users can work with software engineers to develop usable software.

Analysis of methodologies from different disciplines and different domains shows that each share the same basic form and that it is the value-systems and techniques applied to each stage that differ. Similarly, a visualisation methodology will include the same and additional stages with each stage employing software, HCI, and visualisation specific techniques.

Reference models describe the visualisation process but do not aid the designer in choosing the most appropriate transformations or visual structures to use. They are however useful in breaking up the complex task of visualisation design into more manageable chunks. They also help form the basis for a visualisation methodology.

Patterns are a method of capturing and sharing design knowledge that may be based on many years of practical experience. They promote the reuse of proven solutions. Both novice and experienced designers can use pattern languages to refine solutions and, by reading the rationale, actually understand why the pattern works rather than just blindly following a set of guidelines. They also provide a common language that all members of a project team, including the users, can use to discuss ideas and evaluate potential designs.

The visualisation community has developed a number of techniques and heuristics that can solve visualisation problems that are independent of domain. In effect these techniques are being reused to solve recurring problems. This is the definition of a pattern. Therefore it

3.7 Summary

should be possible to formalise these techniques into patterns and organise them to form a pattern language. The inclusion of patterns in user-centred methodologies would suggest that any visualisation methodology could be supported by patterns in a similar fashion.

Heuristics provide useful guides for designers but there are a number of problems with the current visualisation heuristics. The visualisation heuristics have been developed separately by a number of different researchers. Many of these heuristics are identical but have different titles and all of them use an unstructured format. Compiling a catalogue of these heuristics and using the pattern approach to capture visualisation design knowledge can overcome these problems.

Visualisation evaluation is also an issue. Empirical evaluation with users can often be difficult to organise, time consuming, and expensive. The heuristic evaluation and estimation techniques employed by usability engineers are one way to overcome these problems. Similar evaluation techniques specific to visualisation would be useful, allowing the designer to compare and rank different visualisation designs.

As stated visualisation is a form of HCI and a visualisation system is a software system. But visualisation involves issues other than those found in standard interface design. Issues such as data type, structure, dimensionality and quantity, display issues such as visual interference, visual clutter, visual structures, layout, etc., and finally interaction issues such as selection, navigation and manipulation. Each of these issues needs to be considered by visualisation designers and should therefore be incorporated into a formal visualisation design methodology.

It is argued that by combining the ideas presented in this chapter and tailoring them to visualisation design a new methodology can be developed. This new methodology can combine software engineering, HCI techniques, and visualisation heuristics, and at the same time take advantage of the benefits of patterns.

Chapter 4: A PATTERN SUPPORTED VISUALISATION DESIGN METHODOLOGY

This chapter describes the proposed methodology and how it represents a synthesis and extension of the research described in chapters 2 and 3. The first section lists the stages that constitute the methodology. The design stage is then described in detail in terms of the heuristics collected, the development of visualisation patterns, and the use of several different methods to corroborate good visualisation designs. Methods for selecting and applying patterns are also included.

4.1 Pattern Supported Methodology

4.1.1 Overview

The proposed methodology consists of five main stages, these being *information gathering*, *analysis*, *design*, *implementation*, and *evaluation*. Initially, information is collected and then structured (e.g. data tables, feature vectors, etc). There then follows a detailed analysis of the data, the tasks, the users and the usability factors that are important to the system design. This defines the structure and content of the visualisation, the tasks that the visualisation needs to support, the target users, and the usability goals. To assist the designer the analysis stage is supported by patterns that help to capture and define the system requirements. If necessary the designer can also examine existing systems as a way of generating ideas and identifying potential problems. Understanding the user, analysing existing systems, and setting usability goals are consistent with the first three elements of the usability engineering life cycle.

In the design stage the designer must determine an appropriate set of transforms, interaction mechanisms, and composition elements. This defines the graphical scene and how the user can interact with the visualisation either to modify its contents or create a new visualisation. For details regarding the transforms refer to section 3.3. The design stage is supported by visualisation design patterns and GUI design patterns, which provide the designer with design knowledge and help guide them to effective designs. In addition, visualisation heuristics are also used, both as design guidelines, and as a way of ranking alternative designs.

Depending on the circumstances, prototyping, or a more formal software engineering methodology, can be used to implement the visualisation. As part of the implementation, software design patterns can be used to support the development of a robust and extendable system.

Once an initial visualisation has been designed and implemented it must then be evaluated. Since the development of the visualisation is an iterative process this evaluation may lead to

4.1 Pattern Supported Methodology

the information gathering, analysis or design stages being revisited. This process may be repeated several times.

An overview of the complete methodology is shown in figure 4.1.

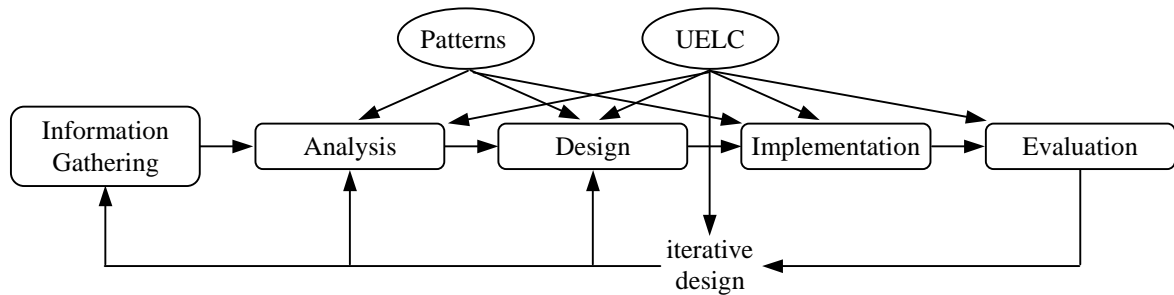


Figure 4.1: Overview of methodology.

The remainder of this section will describe the methodology in more detail.

4.1.2 Information Gathering

Data can come from a variety of sources. In scientific domains, for example, it is common to collect large quantities of sensor data. This type of raw data often has to be structured before it can be interpreted. Similarly, the raw text in a document collection is often represented as a set of document feature vectors before it is viewed. Even data that is entered directly (e.g. on screen forms) such as medical records is in effect being transformed from a set of raw information bits to a predefined unit of related information.

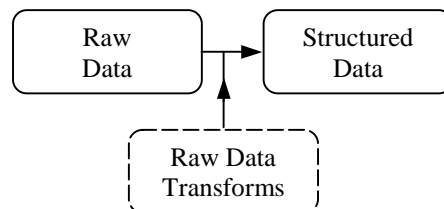


Figure 4.2: Information gathering.

It can be seen that the goal of the raw data transforms is to convert the raw data into some predefined structure. Typically the conversion process will force the data into data tables that can be stored in a database. The benefits of database systems such as fast searches, merging of tables, etc., can then be taken advantage of.

4.1.3 Analysis

To develop a visualisation a designer must determine what transforms need to be applied to convert the structured data into visual structures that can then be rendered and interacted with.

4.1 Pattern Supported Methodology

They must also decide what interaction techniques to use and how to lay out each individual visualisation or view. To do this they must first of all analyse four main components, the *data*, the *task*, *usability goals* and *user characteristics*. The analysis phase simply highlights the relevant factors that need to be considered for each component so that the designer can then try to balance the forces from each one. The analysis phase is shown in figure 4.3.

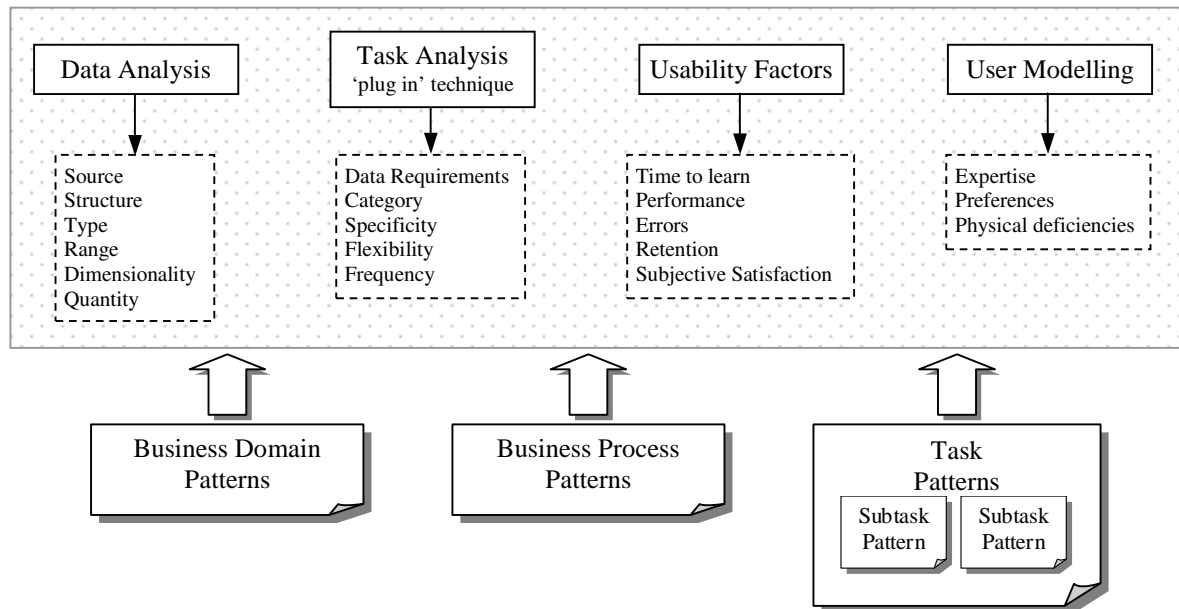


Figure 4.3: Analysis stage.

4.1.3.1 Patterns

In this phase, domain-dependent information patterns similar to those described by Granlund et al. (2001) can be used to define the system, describe the business goals, and capture knowledge about the tasks and users. By using patterns the designer can be led to solutions that have been proven to work with similar tasks elsewhere. Other benefits of patterns such as communication between project team members, capture of knowledge for future projects, project documentation, etc., will also be available.

4.1.3.2 Data

Data analysis helps to determine the content and form of the visualisation. Specifically, data analysis can be used to decide when data needs to be manipulated (i.e. data transforms), what graphical structures and objects can be used, and which data features should be mapped to which visual features (i.e. mapping transforms). To determine these mappings the following data factors should be considered.

4.1.3.2.1 Source

The source of the data can provide a useful starting point for the primary visual structures to be used in the visualisation. For example, sun spot activity could be presented on a sphere, the temperatures of different engine parts could be viewed on a model of an engine, text in a document can be shown using the layout of the document, traffic flow data can be encoded onto a map of the city from which it was taken, and so on. In other words the physical object from which the data is collected can be used as a template for the visual structure used in the visualisation.

A similar approach is based on the idea of *metaphor*. This is the technique of using objects and concepts that are familiar to the user as part of the visualisation design. Typical examples of a metaphors used in many software environments include the ‘desktop’, ‘folders’ and the ‘recycle bin’. The importance of metaphor in interface design has led Alty et al. (2000) to develop a framework that software designers can use to apply metaphor in the design of interactive systems.

Analysing the data source and employing metaphor helps the designer to develop a visualisation that may more closely match the users mental model and therefore be more easily learnt and comprehended. However, when using the data source, and in particular metaphor, the designer must take care to ensure that any behaviour within the visualisation (e.g. interaction) is consistent with the expectations of the user.

4.1.3.2.2 Structure

The data structure and the data source are obviously closely related. The data structure is a domain-independent classification of the source. So, for example, document text is classed as one-dimensional, map-based data is spatial or two-dimensional, data collected over time may be classed as temporal, etc. However, data from a single source may consist of multiple structural classifications. Consider sun spot activity data. This is inherently spatial because each spot location is stored as a pair of co-ordinates (latitude/longitude). The activity on the other hand is recorded over long periods of time so the data also has a temporal aspect. Both of these structural components may need to be incorporated into a visualisation depending on the results of a task analysis.

Some data structures lend themselves more readily to visual structures than others. Spatial data is often viewed using a two or three dimensional co-ordinate system, and hierarchical data and network data map well to node-link diagrams. Multidimensional data and temporal data however have no obvious visual form. This does not mean that these data structures

cannot be visualised it simply means more abstract structures or specialised techniques may have to be used e.g. temporal data is often represented using three dimensions or animation.

Analysing the data structure can lead the designer to visual structures and techniques that have been used successfully in the past.

4.1.3.2.3 Type

Recall that data items (or records) represent data objects (or concepts) and collections of data items form datasets. Whereas the data source and the data structure refer to datasets, the data type refers to individual attributes of each item i.e. properties of the object a data item represents.

There are three basic data types, *nominal*, *ordinal*, and *quantitative*. The accuracy with which data can be interpreted using different visual features has been studied in depth by a number of researchers including Mackinlay (1986), Casner (1991) and Salisbury (2001). The results of these studies have been applied to numerous visualisation designs and encoded in automatic visualisation generators.

The data type has a significant influence on the choice of chart-based presentation. This is because chart-based displays are only capable of supporting certain data types e.g. scatter plots support only quantitative data and bar charts require one quantitative data type and one ordinal. Salisbury (2001) has looked at the capabilities and requirements of chart-based displays in detail and used the results to implement an automatic visualisation generator. Since initially novel visualisation designs have few restrictions, the most appropriate visual feature can often be used to encode the data type as required. Obviously, as visual features are 'used up' the number of encoding options is reduced.

4.1.3.2.4 Range

The data range refers to the set of values that a data attribute has had assigned to it within a dataset. Considering the range is important for several reasons. Casner (1991) suggests that there are practical limitations on the number of unique values that different visual features can encode. This means that, for example, there is little point using line thickness to encode 100 different values since, according to Casner, only 3 line thicknesses can be practically encoded into a visualisation.

While the validity of the exact values that Casner places on each visual feature is arguable, it is clear that limits do exist. Techniques for reducing the range of values such as filters or converting quantitative data types to ordinal types may need to be considered.

4.1.3.2.5 Dimensionality

The more data dimensions that a visualisation is required to display the more difficult it is to design a visual structure that can be rapidly and accurately interpreted by the user. Analysing the dimensions and comparing the current requirements to those supported by existing visualisations may lead the designer to appropriate techniques for coping with high-dimensional systems.

4.1.3.2.6 Quantity

The quantity of data affects both the choice of visual structure and the interaction techniques used in the visualisation. As with data dimensionality, visualisation techniques that have been developed to cope with large quantities of data have been discussed in detail in section 2.8. Since quantity has such a significant effect on the design it is obvious that it must be carefully considered during the analysis phase.

4.1.3.3 *Task*

Task analysis determines the goals the user is trying to achieve and the methods they use to achieve them. Analysis of existing systems can help to highlight problems that should be resolved in the new system and to identify patterns. The visualisation designer can use the identified tasks and task processes to determine both the data and the interaction mechanisms needed to support the tasks. In terms of visualisation design the following task factors should be considered.

4.1.3.3.1 Data Requirements

The results of a task analysis will include the data items and specifically the data attributes relevant to the tasks. Depending on the results of the task analysis the data source and structure may or may not be relevant to solving the task. For example, if the task is to compare the populations of various cities there is no point in using a map-based visualisation since the relative locations of the cities is irrelevant. In other words, in this instance, the data source has no effect on the task requirements. This is not to say that using a map would be incorrect, doing so may or may not impede task performance, again, depending on the task itself. On the other hand the data types are always relevant since they relate directly to the most effective visual features to use.

In general, the data requirements will affect the design of the data and mapping transforms. Data that summarises information or is the result of calculations such as additions, divisions,

counts, means, etc., will likely be determined using data transforms. Other task requirements may mean that data attributes are mapped to visual features.

4.1.3.3.2 Category

Tasks can be classified into several different task categories. Different researchers have suggested a number of different task classifications. The task categories presented here are based on those proposed by Salisbury (2001) since they cover a wide range of tasks. However, some of Salisbury's definitions are unclear and it was found that some categories could be further divided to make task classification more accurate. The definitions of the categories we use are as follows:

- *Calculations*: Performing arithmetic operations, such as addition, subtraction, multiplication, etc., on data values.
- *Comparison*: Looking at the similarities and differences between values of attributes *within* a data set or *between* data sets. For example comparing the engine size of car A to the engine size of car B, car C, and so on.
- *Trends*: Looking at the change in the value of an object attribute over time.
- *Relationships*: Looking at the dependencies between the values of object attributes *within* a data set or *between* data sets.
- *Aggregation*: Grouping data according to some unifying principle and analysing group attribute values.
- *Spatial*: Requires knowledge of locations and often asks *where* something occurred.
- *Distinguishing*:
 - *Find*: Find all data items matching a set of criteria.
 - *Lookup*: Find the value of an attribute for an object or objects in the data set.
- *Optima*: Finding the best, worst, most, least, etc., values within a data set.

It has been shown by researchers such as Casner (1991) and Salisbury (2001) that the time it takes humans to perform different tasks varies depending on the task category. The order of difficulty of the task categories is as presented above i.e. humans find calculations the most difficult and optima tasks the least difficult.

Knowing the task categories that a visualisation needs to support can help the designer determine the data and mapping transforms. For example, imagine the task is to calculate the mean of a set of values. The designer could use a mapping transform and encode each individual value as a visual feature of some visual object in the display e.g. the length of a bar. Alternatively the designer could use a data transform, which uses an algorithm to calculate the mean and then simply displays the result. Using the mapping transform the user must judge the length of each individual bar and then estimate the average length. This places a

significant cognitive load on the user. The data transform on the other hand allows the user to look at one visual object and know the answer. This places far less cognitive load on the user and is significantly more accurate, so in this case is more appropriate. Also, it can be seen from the order of difficulty of the task categories, calculations are the most difficult for humans therefore getting the computer to perform this difficult task is sensible. However as will be discussed, other task factors need to be considered which may result in the mapping transform being more appropriate.

4.1.3.3.3 Specificity

The specificity of a task refers to how well the values of arguments to the task are known when the task is defined i.e. the task constraints. The higher the level of specificity the more likely it is that data transforms should be used. Consider the following task examples, based on the task specificity examples proposed by Chuah (2002), which relate to the general task of finding a car or cars in a database:

- a) Find a car with registration number 'BXW 2002'.
- b) Find a "good" car.

In example (a) the task has a high specificity. In this case it would be relatively straightforward to write an algorithm that searches the database for a car with the specified registration number and present details about it to the user. In other words a data transform would be most appropriate.

In contrast, example (b) has low task specificity. In this case it is difficult to know exactly what criteria different users may consider when determining if a car is "good". The designer could assume that a "good" car relates to its price, top speed, and fuel consumption. The designer could then map each of these attributes to an interface control and link each control to a data transform. The user could then use these controls to specify the values of the task arguments and matching cars could be displayed as a list.

The problem with this pure data transform design is that the argument values are too precise. Cars that have similar argument values but do not match exactly will be eliminated despite the fact that the user may consider them acceptable. For example, a user may define a "good" car as one that has a low price, high top speed, and low fuel consumption. However, they may be willing to accept a lower top speed if the price and/or fuel consumption of all high speed cars is unacceptable. In general the user is trying to balance these three factors, therefore the task input argument values cannot be specified fully at the outset. An alternative would be to use a pure mapping transform approach with each car represented as a visual object on screen and attributes of the car encoded as visual features of the objects e.g. as marks in a scatter plot.

The user could easily adjust their constraints by looking at different areas of the scatter plot. However, if the number of cars in the database is large a pure mapping solution may introduce the problems of visual clutter, occlusion, etc. In this case the designer may opt to combine both data and mapping transforms.

What is important to note here is that if the task has a high specificity then using an algorithm to find the result is probably the best solution.

4.1.3.3.4 Flexibility

Chuah (2000), who refers to task flexibility as task variation, states that the main disadvantage of using data transforms is that they are only designed to support one specific task. In addition, only the results of data transform calculations are returned, the initial values and any intermediate results are lost. Consequently, any visualisation that makes use of these results is potentially less flexible than one that is based on the original data.

A visualisation that relies on mapping transforms may be far more flexible in terms of the tasks it is capable of supporting. For example a simple scatter plot allows a user to see clusters, outliers, min and max values, and so on. It would require multiple data transforms to solve the same set of tasks. The ability of a human user to interpret a visual display in multiple ways must be balanced against the benefits of data transforms when considering flexibility.

4.1.3.3.5 Frequency

The task frequency can be used to determine which tasks, and the data attributes required to support those tasks, should be given priority when designing the visualisation. When designing the mapping transforms the visualisation designer should try to map the data attributes that are most relevant to the most frequent tasks to the most appropriate visual features first. This will allow the user to more easily focus on the data relevant to the common tasks and so improve their performance. Less common tasks may be more difficult to perform using the visualisation but the gains made with the common tasks may be worth the additional effort.

As with standard GUI's, task frequency should be considered so that frequent tasks can be easily learned, easily performed, and easily recalled from one session to the next.

4.1.3.4 Usability Factors

Recall that HCI specialists can determine the effectiveness of a GUI by measuring a set of usability factors which include the time to learn the system, the speed with which tasks can be completed, the number and rate of errors, how well users remember how to use the system, and how satisfied users are with the system. When designing a visualisation, these same usability factors need to be taken into account, and the designer must have clearly defined usability goals.

The application domain often has a significant effect on the usability goals of the system. In domains where users are given adequate training, the time it takes to learn the system is less important than in a domain where users are expected to start using the system straight away, with little or no training. Similarly, when the users are given sufficient training, and the system is in use every day, retention is less of an issue. In safety critical systems in particular, and most domains in general, maximum performance with minimum errors are usually the most important factors. However, both of these may be affected by the users' subjective satisfaction. If users feel they have to 'fight' the system then their performance may suffer and error rates increase. It is therefore important that the user feels content using the system.

Just as the GUI can affect the usability factors so can the visualisation. If the visual structures and the interaction techniques used in the visualisation do not match the user's mental model or expectations then the visualisation will be more difficult to learn, performance will be impeded, the number of errors will increase and the user's satisfaction levels will be low.

4.1.3.5 User

Various user attributes must also be considered when designing a visualisation. Recall that HCI specialists have developed techniques for modelling users, recording details such as their level of expertise (i.e. novice, intermediate, expert), preferences, and physical disabilities.

Some of these user attributes relate to usability factors. For example, novice users may take longer to learn the system and initially be slower and more error prone. Other attributes are especially important when designing mapping transforms. For example, using red and green in a display may lead to misinterpretation by red-green colour-blind users, which in turn may lead to high error rates. In this case an alternative mapping such as size, or a combination of both size and colour may be more appropriate.

4.1.3.6 Discussion

It is clear that there are conflicts between each of the analysis factors. For example, a task may have a low specificity so the designer would like to show the entire dataset using mapping transforms, but the quantity of data is large, which may result in occlusion. Therefore, the designer may have to include some data transforms in order to reduce the quantity of data displayed, which means potentially reducing the task flexibility of the visualisation. Typically, there is no optimal solution but, by analysing the factors identified in this stage of the methodology, it is hoped that the visualisation designer can reach a satisfactory compromise.

4.1.4 Design

A visualisation can be thought of as a visual representation of a dataset that can be interacted with directly or via standard GUI controls. The visualisation process that converts the data into a view was established in section 3.3. Essentially this process consists of a number of transforms. Once the designer has analysed the various factors that affect the visualisation, they can proceed with the design of each of the remaining transforms and the interaction techniques that will be used to initiate or modify them. If multiple views or multiple visualisations are required the designer may also have to design both the layout of these views and any inter-visualisation communication.

The transforms that the designer must design at this stage include *data transforms*, *mapping transforms*, *graphical transforms* and *rendering transforms*. The data and mapping transforms define the contents of the visualisation. The graphical and rendering transforms alter visual aspects of the visualisation in order to provide user feedback and enhance readability.

It is important to understand the relationship between interaction and the graphical and rendering transforms. Interaction techniques determine how the user can take some action whereas graphical and rendering transforms are the result of that action. The designer must design both the interaction mechanisms and the resulting transforms. The user simply activates a pre-designed transform using a pre-designed interaction technique.

The components that the designer must create for each visualisation, the transforms and the interaction mechanisms, and the composition elements required for multiple linked visualisations, are shown in figure 4.4 together with the patterns and heuristics that can support them in this process.

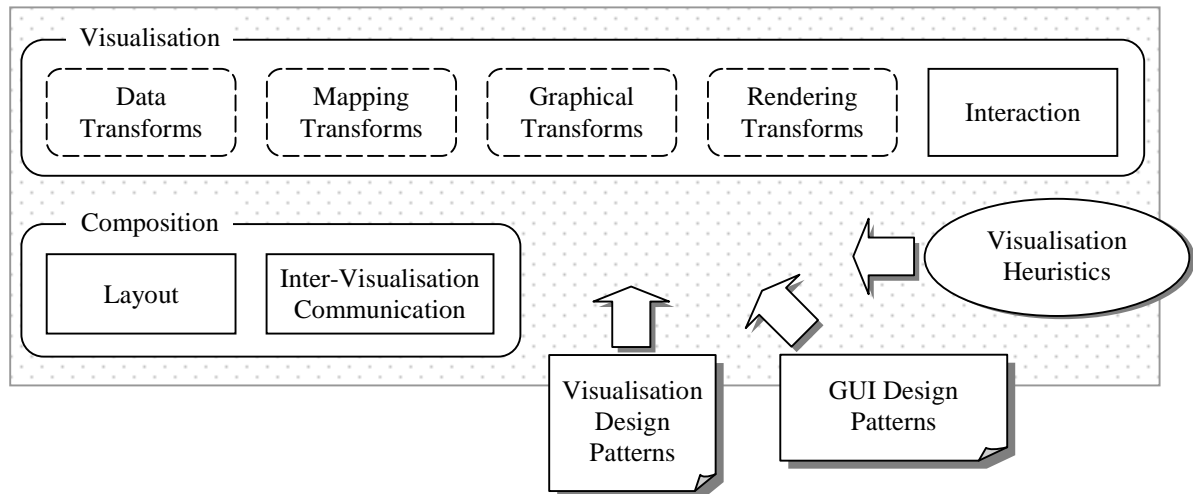


Figure 4.4: Design stage.

The analysis phase highlighted the factors that need to be considered during design. The design phase has established what visualisation components need to be designed. However we still need methods that lead the designer towards effective designs.

One of the goals of this research is to provide a human designer with a methodology that allows them to create designs that combine novelty, perceptual effectiveness, and usability. Visualisation heuristics help to guide the designer but at the same time give them the freedom to be creative. Heuristics are not fixed rules that must be obeyed; they simply describe desirable characteristics. It is up to the designer to determine how these characteristics are achieved. As Welie et al. (2000) point out, and as was discussed in section 3.6.3, there are also a number of problems with heuristics. Therefore, this freedom of creativity comes at a price. As well as producing good designs it is also possible to produce bad ones.

The benefits of patterns were discussed in section 3.6.3. Patterns are a convenient method of capturing and sharing design knowledge that has worked effectively in previous designs. However patterns tend to be less flexible than heuristics. The solution a pattern describes can only be adapted in a limited number of ways, and once a designer has seen an example of how a pattern has been applied, they may be less inclined to develop a novel solution of their own. Therefore, although patterns will lead to effective designs they inhibit the creativity of the designer.

The approach we propose is to use a combination of visualisation heuristics, visualisation design patterns, and GUI design patterns. Mixing the visualisation heuristics and the visualisation design patterns allows the designer to be both creative and at the same time use techniques that have been proven to be effective. The GUI design patterns are those found in HCI literature such as Tidwell (1999) and can be used to design the GUI surrounding the view part of the visualisation.

4.1 Pattern Supported Methodology

During the design process a designer will inevitably need to consider several alternative designs. A novel use of visualisation heuristics is proposed to rank the usability of each design. This design evaluation technique, the development of visualisation heuristics and visualisation patterns and methods for selecting and applying them are discussed in detail in sections 4.2, 4.3, and 4.4.

With reference to the usability engineering life cycle, parallel design, participatory design, coordinated design of the interface, and the application of heuristics, can all be used to help ensure the usability of the visualisation. These elements and their relationship to the patterns that support this stage of the methodology are summarised in table 4.1.

	Usability Engineering Life Cycle	Pattern Use
4	Parallel design (several initial designs by independent teams).	*Use general HCI design patterns (maybe from book) as common design guidelines for the teams. **General visualisation design patterns can be used in the same way.
5	Participatory design (actively involving users in the design process).	*Application domain expert (user) and HCI designer exchange their pattern languages for better mutual understanding. **Domain expert, HCI designer, and visualisation designer can all exchange pattern languages for better mutual understanding.
6	Coordinated design of the total interface (consistency within and across products).	*Lower-level HCI design patterns, including project-relevant, concrete examples, communicate the common look and feel efficiently. **Visualisation design patterns can be used in the same way.
7	Apply guidelines and heuristic analysis (style guides, standards, and guidelines).	*HCI patterns improve upon those formats because of their standard format, hierarchical networking, inclusion of examples, and discussion of problem context as well as solution. **Visualisation design patterns provide the same benefits as HCI patterns.

*Borchers' (2000a, 2000b) application of HCI patterns to usability engineering life cycle.

** Application of visualisation design patterns to usability engineering life cycle.

Table 4.1: The relationship between patterns and elements of the usability engineering life cycle in the design stage.

4.1.5 Implementation

Once the designer has designed all the components listed in section 4.1.4 they can then start to implement the visualisation system. At this point, any of the common software engineering methodologies described in section 3.2.1 can be used. The use of Rapid Prototyping, in implementation and at the design stage, can help the software engineers to understand the user requirements, and the visualisation designers and HCI specialists to identify usability problems. The design team can simply 'plug in' the approach they are most familiar with or feel is most appropriate to their business domain. Software engineers may even use software design patterns as part of the software design and implementation process. This is shown in figure 4.5.

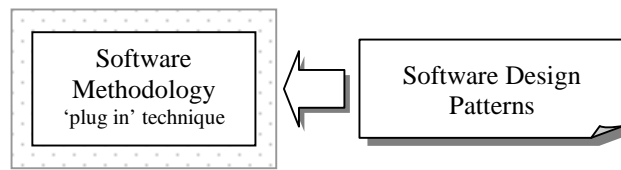


Figure 4.5: Implementation stage.

4.1.6 Evaluation

Once the visualisation has been designed and built it must be evaluated to see if it is capable of supporting the user in their tasks and meets all of the desired usability criteria. Researchers in HCI have developed a number of usability evaluation techniques including cognitive walkthroughs, focus groups, usability inspection, heuristic evaluation, etc. The most appropriate technique to the given situation can simply be 'plugged in' to the methodology. However, the decision regarding which technique to use is a difficult one that depends on many factors including time and cost constraints, type of user, and the usability factors you are most interested in achieving.

One method of evaluating visualisation designs is to test each visualisation with users under controlled conditions. The data gathered then undergoes statistical analysis in order to answer usability questions such as which visualisation took the least time to learn, which had the highest percentage of correct answers in the fastest time, which did the users prefer and so on. Studies that have used this approach include Morse et al. (2000), Trafton et al. (2000) and Sutcliffe et al. (2000).

A more formal evaluation methodology developed by Graham et al. (2000) is shown in figure 4.6. It is interesting to note that their approach evaluates both the visualisation and the user interface that surrounds it and focuses on usability testing. These ideas fit well with those we propose in this research.

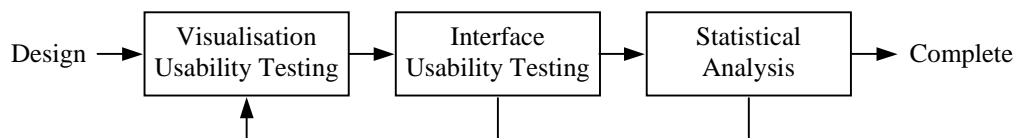


Figure 4.6: Evaluation process.
(Modified from Graham et al. (2000))

It should be noted that the visualisation evaluation is different to the design evaluation technique mentioned in section 4.1.4 and discussed in more detail in sections 4.2 to 4.4. The evaluation discussed in this section is typically conducted with users after implementation whereas the design evaluation is conducted by the design team as part of the design phase.

4.1.7 *Iterative Design*

It is important to note the iterative nature of the visualisation design methodology. In the early stages of the design process, after information gathering and initial analysis, the design team should use prototyping and testing with users as a way of establishing requirements and identifying usability issues. These prototypes do not have to be fully functional, fast, or particularly stable. After generating an initial prototype design and gathering feedback during evaluation the design team may return to the information gathering, analysis, or design phase to determine the causes of any problems and try to resolve them. This prototyping process can continue until the desired usability or understanding has been reached. If necessary, at this point an alternative software methodology can be chosen to develop a more final implementation.

The complete methodology can be seen in figure 4.7.

4.1 Pattern Supported Methodology

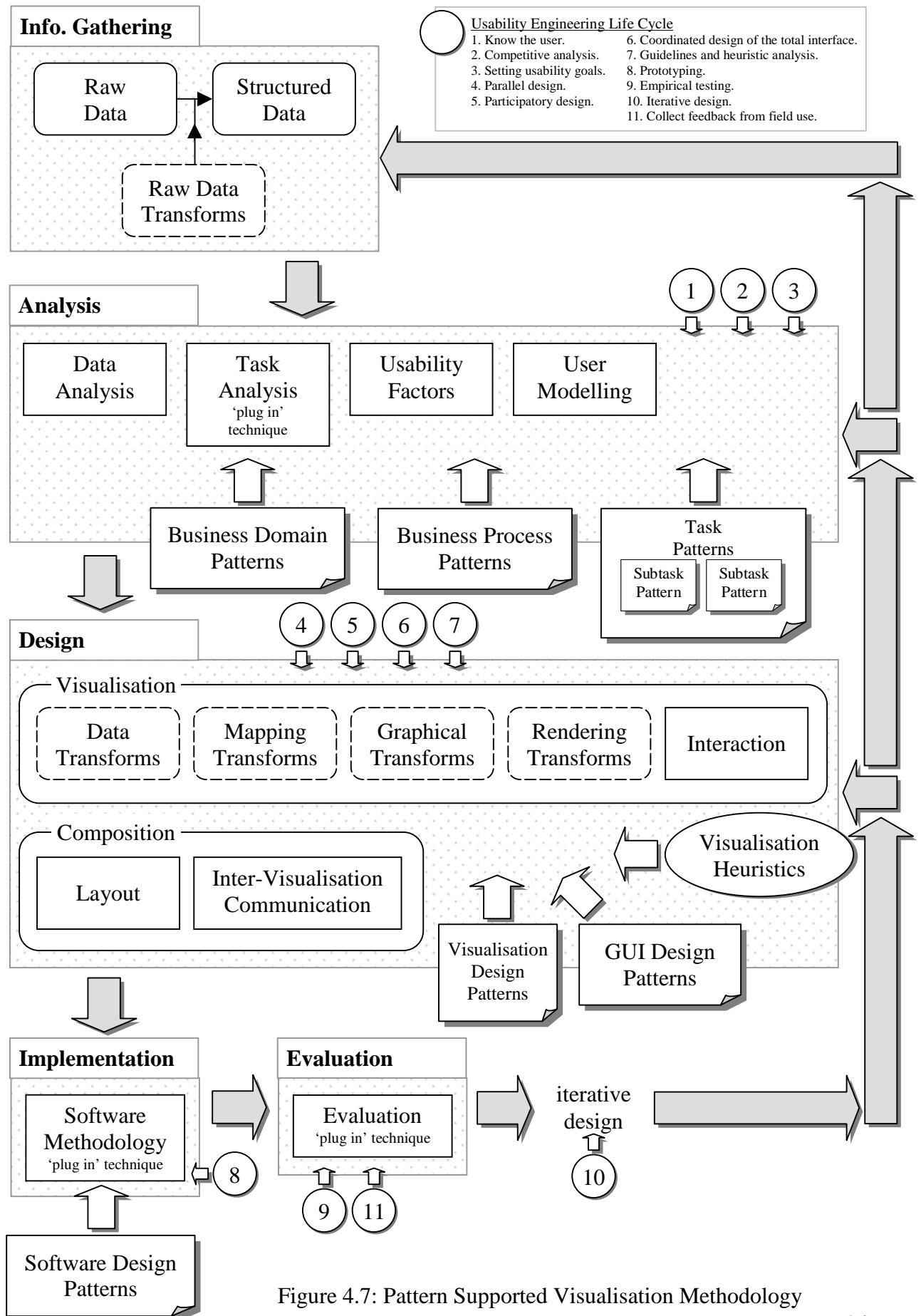


Figure 4.7: Pattern Supported Visualisation Methodology

4.2 Heuristics

Recently visualisation researchers have started to record their experiences and the lessons they have learned during visualisation design using heuristics. It should be noted that, unlike the lists of heuristics presented by Nielsen (1994), or Shneiderman (1998), the visualisation heuristics reviewed came from research papers describing successful visualisation techniques, and as such included detailed rationales. The hope is that these heuristics will help other visualisation designers to produce effective designs and avoid common pitfalls. However, the visualisation heuristics recorded so far tend to be unstructured and, because they are from various sources, are often repeated but under different titles.

One objective of this research is to compile a comprehensive catalogue of these experience-based heuristics. Providing a single source makes information easier to find, reduces the chances that information will be missed, may help to generate new ideas, helps to identify gaps in knowledge and provides a useful place where other heuristics can be added in the future. It is the starting point for the development of a single comprehensive library of heuristics that can be added to, updated, and maintained by other researchers.

Compiling a list of heuristics is also a useful starting point for the development of a visualisation pattern language. This idea is discussed in section 4.3.

Table 4.2 shows the list of domain independent heuristics collected together with the visualisation researchers that have proposed them. It should be noted that different researchers have often used different titles for the same heuristic. Therefore the title used may not match the heuristic title used in the original source.

Some researchers, particularly those who have focused on only one or two heuristics, have been omitted from the table to aid clarity. For example, the SDM techniques developed by Chuah et al. (1995) are an example of heuristic 32, direct manipulation, and the layout principles presented by Baldonado et al. (2000) relate to heuristic 31, multiple linked views.

#	Title	Brath, R. (1999)	Carr, D.A. (1999)	Eick, S.G. (1995)	Foley, J. Van Dam, A. (1995)	Rheingans, P. Landreth, C. (1995)	Shneiderman, B. (1996)
1	Use a real world physics model					✓	
2	Visually refer all graphical objects to a reference context	✓ (3)					
3	Use connotative mappings	✓ (5.2)					
4	Use an organisational device the user already knows	✓ (9)					
5	Use redundancy to aid discrimination and comprehension	✓ (6)				✓	
6	Use different visual dimensions differently	✓ (5.1)		✓	✓		
7	Minimise illusions	✓ (7)				✓	
8	Use colour carefully	✓ (11)					
9	Use smooth animation and motion			✓	✓		
10	Visualisation is not always the best solution		✓				
11	Don't use 3D if the number of data points is low	✓ (10)	✓				
12	Map data to an appropriate visual object		✓				
13	Test your designs with users		✓				
14	Use datatips for identification, education and validation	✓ (2)					
15	Provide a simple 3D navigational model	✓ (1)					

Continued on next page...

4.2 Heuristics

#	Title	Brath, R. (1999)	Carr, D.A. (1999)	Eick, S.G. (1995)	Foley, J. Van Dam, A. (1995)	Rheingans, P. Landreth, C. (1995)	Shneiderman, B. (1996)
16	Use small multiples to encode multiple data attributes	✓ (4)			✓		
17	Use legends, scale and annotation	✓ (8)					
18	Do not rely on interaction	✓ (12)					
19	Occlusion is undesirable	✓ (13)					
20	Use interaction to explore large data sets	✓ (14)					
21	Let users control visual bindings				✓		
22	Emphasise the interesting					✓	
23	Task specific		✓	✓			
24	Overview		-	✓	✓		✓
25	Zoom		-		✓		✓
26	Filter		-	✓	✓		✓
27	Details on Demand		-	✓	✓		✓
28	Relate		-				✓
29	History		-				✓
30	Extract		-				✓
31	Multiple Linked (Co-ordinated) Views		✓	✓	✓		
32	Direct Manipulation			✓			

Key

✓ Indicates a researcher who agrees with the stated heuristic

(n) The number in brackets in the Brath (1999) column is the ranking or level of confidence that Brath has assigned to the heuristic. Lower numbers indicate more confidence in the heuristic.

The rows in grey are defined in Shneiderman's (1996) task by data type taxonomy and represent a generic set of requirements all visualisations should meet.

- The work by Carr (1999) is based on that of Shneiderman (1996) therefore he agrees with each of Shneiderman's guidelines, this is indicated by the dash (-).

Note: Foley and Van Dam's (1995) work is used as cited by Brath (1999).

Table 4.2: Heuristics and sources.

In the original sources the heuristics were presented using a short title and a detailed description of the rationale behind the heuristic. Whilst compiling the catalogue we have kept this basic format and extended it to include a heuristic number, multiple supporting authors and links to related heuristics. This is only possible since heuristics from multiple sources have been collected.

This format is useful since it allows heuristics to be more easily converted to the pattern format where appropriate. The heuristic format attributes are defined as follows:

- *Number*: Can be used to refer to the heuristic and provides a useful mechanism for referring to other heuristics. The number format is designed to allow reference to specific sub-heuristics if appropriate.
- *Name* or *Title*: Provides a short description of the guide the heuristic describes.
- *Description* or *Rationale*: Describes the reasoning behind the heuristic. In some cases this may also include written or visual examples. The descriptions used in the catalogue tend to represent a more concise summary of those found in the original sources.
- *Supporting author(s)*: Lists the authors that have explicitly stated this heuristic as part of their design guidelines. The number of supporting authors may be a good indication of the validity of the heuristic.
- *Related heuristics*: Lists any heuristics that may also need to be considered.

4.2 Heuristics

An example of the format used in this catalogue is shown in table 4.3. The complete set can be found in appendix A.

Heuristic 9.0	
<i>Title</i>	Use smooth animation and motion especially for temporal data
<i>Description / Rationale</i>	Animation or motion is an effective means of representing data that has some temporal aspect. The most important consideration when using this technique is to make the transition from one state to another as smooth as possible. Animation or motion that produces a large change of state distracts the user from the information and may cause important features to be missed. Large jumps from one location in a 3D environment to another may cause the user to become disoriented; a smooth transition helps them to maintain their context in the environment. If possible the user should be able to control the rate of change and direction of the animation, this facilitates the users exploration the data.
<i>Supporting author(s)</i>	Eick (1995), Foley and Van Dam (1995)
<i>Related heuristics</i>	15.0, 25.0, 32.0

Table 4.3: Heuristic format.

Assigning numbers to the heuristics and providing references between them allows the entire catalogue to be viewed as a matrix as shown in table 4.4. A colour-coded bar chart has also been attached to the top of table 4.4 indicating the number of supporting authors. The matrix representation can help a designer to see the relative strengths of each heuristic and which related heuristics should be considered. For example, looking at heuristic 9 in the matrix it can be seen that two authors support the heuristic and there are three related heuristics, 15, 25 and 32.

Having a catalogue of heuristics provides a useful source of information. However the visualisation designer still needs to know what effect the heuristic has on the design. Chapter three emphasised the fact that visualisation is a form of HCI and that HCI focuses on developing usable systems. To measure the usability of a system HCI specialists use a set of usability factors. Therefore, the heuristics should be classified by their effect on these usability factors.

4.2 Heuristics

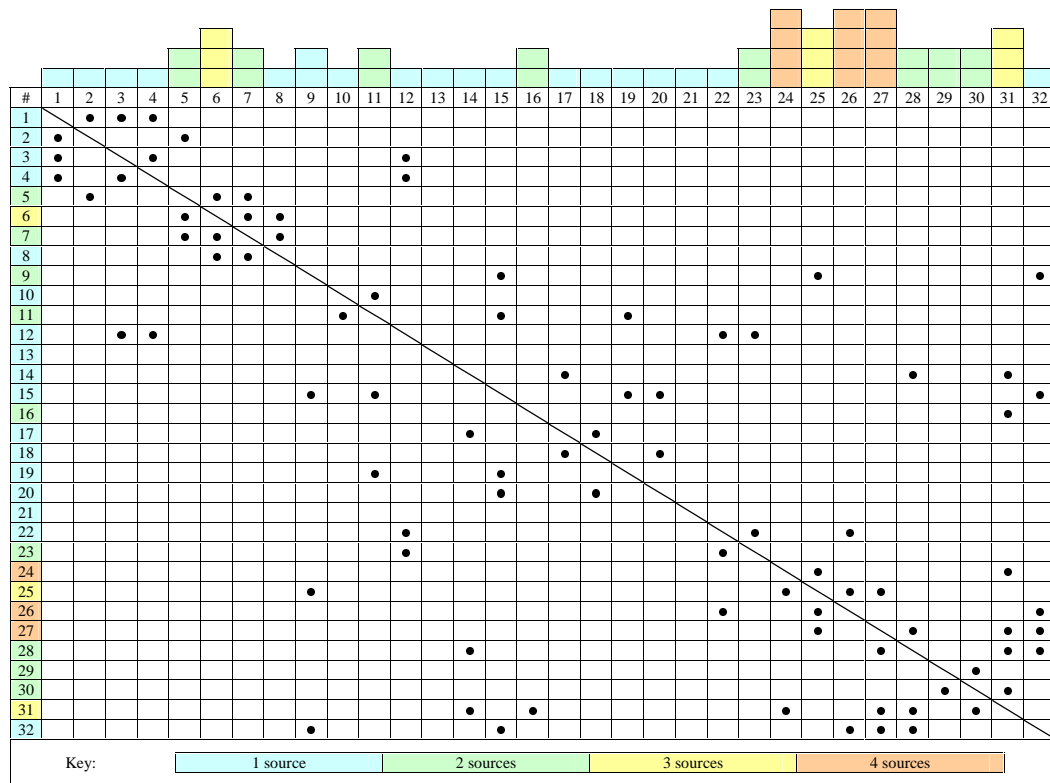


Table 4.4: Heuristics matrix.

Recall that usability factors include *time to learn*, *speed of performance*, *rate of errors*, *retention over time*, and *subjective satisfaction*. It is important to note that to classify the heuristics these definitions refer to the visualisation, **not** the data contained within the visualisation. For example, the time to learn refers to the time it takes the user to learn how to use the visualisation, not how long it takes them to interpret what the data means.

In order to determine the effect of each heuristic on each usability factor it is necessary to simplify the factor as much as possible. For example, learnability consists of a number of dimensions including the time to learn how to use the system, the ease with which tasks can be carried out, the experience level of the user, the levels of functionality of the system, the frequency of use, and so on. To analyse each factor in detail and to determine the effect of each heuristic on each dimension of each factor is beyond the scope of this research. It is also unnecessary; a simplification of each factor should allow the relative merits of each heuristic to be determined. In addition, it is often not possible to perform a completely accurate analysis simply because we do not have enough knowledge about how people learn and use systems. For example it is not known exactly what factors make something easy or difficult to learn. Therefore simplifying the usability factors is a valid solution.

Instead of trying to quantitatively determine the effect of each heuristic, a qualitative approach has been used. This fits well with the idea of using a simplified model. It should be noted that these assignments are not definitive. As stated each factor is extremely complex

4.2 Heuristics

which, together with the fact that heuristics may influence each other, means that no assignment can be said to always be correct.

The effect of each heuristic on each usability factor has been assigned as shown in table 4.5. These assignments were developed from the literature reviewed and from discussions with psychologists at QinetiQ, and visualisation and HCI researchers.

		Learn	Perf.	Errors	Retnt.	Sb.St
1	Use a real world physics model	+	+	+	+	?
2	Visually refer all graphical objects to a reference context	0	+	+	0	+
3	Use connotative mappings	+	+	+	+	+
4	Use an organisational device the user already knows	+	+	+	+	+
5	Use redundancy to aid discrimination and comprehension	0	+	+	+	?
6	Use different visual dimensions differently	-	+ & -	+ & -	0	?
7	Minimise Illusions	0	+	+	0	?
8	Use colour carefully	+	+ & -	+	0	+
9	Use smooth animation and motion especially for temporal data	0	0	+	0	+ & -
10	Visualisation is not always the best solution	+	+	+	+	?
11	Don't use 3D if the number of data points is low	+	+	+	+	?
12	Map the data to an appropriate visual object	+	+	+	+	?
13	Test your designs with users	0	+	+	+	+
14	Use datatips for identification, education and validation	+	+	+	0	?
15	Provide a simple 3D navigational model	+	+	+	+	+
16	Use small multiples to encode multiple data attributes	+ & -	+ & -	+ & -	?	?
17	Use legends, scale and annotation	+	-	+	+	?
18	Do not rely on interaction	N/A	N/A	N/A	N/A	N/A
19	Occlusion is undesirable	0	+	+	0	+
20	Use interaction to explore large data sets	+ & -	+	+	0	?
21	Let users control visual bindings	+ & -	+	+ & -	+	+
22	Emphasise the interesting	0	+	+	0	?
23	Task specific	+	+	+	+	+
24	Overview	-	+	+	0	?
25	Zoom	-	+	+	0	?
26	Filter	-	+	+ & -	0	?
27	Details on demand	0	0	+	0	+ & -
28	Relate	0	+	+	0	?
29	History	+	+	+	0	+
30	Extract	N/A	N/A	N/A	N/A	+
31	Multiple linked (co-ordinated) views	-	-	+ & -	0	?
32	Direct manipulation	+	+	+ & -	+	?

Key:

0	<i>No effect.</i>	The heuristic has no known effect on the factor.
+	<i>Positive effect.</i>	The heuristic has a positive effect on the factor e.g. increases performance, reduces the number of errors, etc.
-	<i>Negative effect.</i>	The heuristic has a negative effect on the factor e.g. decreases performance, increased the number of errors, etc.
?	<i>Undefined effect.</i>	The heuristic has an undefined effect on the factor. This is typically for subjective satisfaction where it is not possible to tell if a heuristic will have a positive or negative effect. However, occasionally it is difficult to determine the effect on other factors.
N/A	<i>Not applicable.</i>	The heuristic does not apply to the factor.

Table 4.5: Heuristic effect on usability factors.

4.3 Visualisation Patterns

A full list of the reasons for the effects of each heuristic on each usability factor is given in appendix B. As an example, table 4.6 illustrates the reasoning behind the values assigned to heuristic 9.

Heuristic 9: Use smooth animation and motion especially for temporal data		
<i>Usability Factor</i>	<i>Effect</i>	<i>Rationale</i>
Time to learn	0	Smooth animation and motion has no known affect on the time it takes to learn how to use the visualisation.
Performance	0	Smooth animation and motion has no known effect on the time it takes users to perform benchmark tasks.
Errors	+	There is less chance of important information being missed if changes from one state to another are taken in small steps therefore errors will be reduced.
Retention	0	Smooth animation and motion has no known effect on how well the user maintains their knowledge of how to use the visualisation.
Subjective Satisfaction	+ & -	When we view objects in motion in the real world we are used to seeing a continuous path from one location to another. Replicating this effect in a visualisation should lead to positive subjective satisfaction. However, if the animation is too slow the user may become frustrated which may cause less satisfaction.

Table 4.6: Rationale for heuristic 9.

Having classified the heuristics in terms of their effects on the usability of a visualisation the designer can now select those heuristics that most closely match their usability goals. In addition, as will be shown in section 4.4, the designer can also use the heuristics together with their effects on the usability factors as a method of ranking alternative visualisation designs.

4.3 Visualisation Patterns

Referring to user interface design patterns Griffiths and Pemberton (2001) state, “A good pattern will have evolved out of the experience (both successes and failures) and observations of a number of designers.” Since the heuristics collected as part of this research are based on the experiences of several visualisation designers the pattern format is an appropriate mechanism for capturing visualisation design knowledge.

Visualisation patterns fall into three main categories (although there is some overlap):

- *Structure Patterns*: Focus on defining the form and content of the visualisation i.e. the graphical scene, and as such relate primarily to data and mapping transforms.
- *Interaction Patterns*: Focus on the interaction mechanisms that can be used to achieve tasks and the visual effects they have on the scene. As such they relate primarily to graphical and rendering transforms.
- *Composition Patterns*: Provide solutions for the effective layout of multiple visualisations and the communication between them. Composition patterns are essentially an extension of GUI design patterns and should conform to similar GUI design principles such as

consistency, feedback, use of shortcuts, etc. However, composition patterns do include solutions that are specific to visualisation and therefore the distinction is justified.

The patterns presented in appendix C have been derived from two main sources: recurring visualisation techniques such as those described in section 2.8, and the heuristics catalogue compiled as part of this research.

As an example consider the *overview and detail* technique used in many visualisations and also described in heuristic 24. Briefly, this technique allows the user to see an overview of the entire dataset in one window and a more detailed subset of the dataset in another window. For a more detailed explanation see section 2.8.2. This technique represents a recurring solution to the problems of lack of screen space and maintaining user context when displaying large datasets. This can be described using the pattern shown in table 4.7.

Recall from section 3.6.2 that collections of patterns can be organised hierarchically to form pattern languages. More formally, Borchers (2000b) defines a pattern language as a directed acyclic graph with the patterns representing nodes and the context and references representing edges in the graph. A designer can use a pattern language to find and refine suggested solutions. Initially the term pattern language may be somewhat misleading. A language implies a syntax, grammar, etc., which are not present in pattern languages. A more obvious term may be pattern hierarchy. However, the term pattern language is used throughout the pattern literature and refers more to the use of patterns as means of communicating ideas between various groups (HCI experts, software engineers, users, etc.). Consequently, the term pattern language is used throughout this thesis.

The pattern language shown in figure 4.8 consists primarily of structure patterns and will be referred to as the *structure pattern language* (SPL). This pattern language is not exhaustive and there are undoubtedly more patterns at and between each level. As with other pattern collections, new techniques and new solutions are constantly being developed and this refinement of, and extension to, existing pattern languages is simply part of the natural process of pattern writing. It can be seen that several of the visualisation patterns exist at the same level and can be applied to most visualisation designs. Note also that some of Tidwell's (1999) GUI design patterns fit within this pattern language. Despite the obvious gaps in the SPL it still supplies the visualisation designer with useful design knowledge.

4.3 Visualisation Patterns

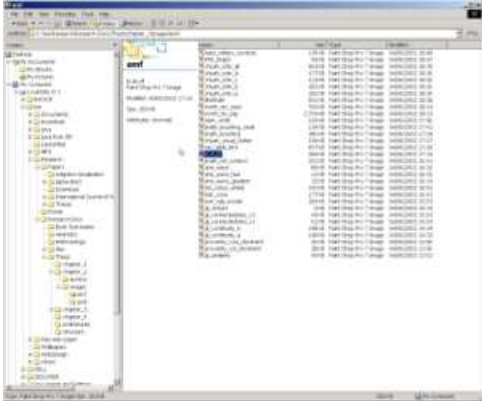
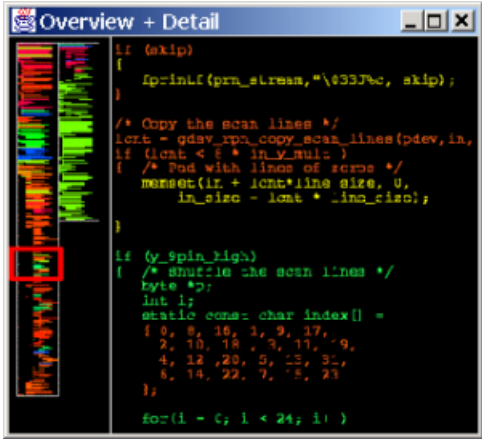
Title	Overview and Detail
Context	The dataset is large, too large for all the details to fit in a single view, and there is a need to view details about subsets of data items. The data can be viewed at one or more levels of abstraction e.g. directories and files within a directory, aggregated document content and detailed document content, etc. Alternatively the dataset may be large and continuous but only a subset can be viewed at any one time e.g. map data.
Problem	How to display the entire contents of a large dataset at once, allow users to explore the dataset, and at the same time show details about subsets of items.
Forces	<ul style="list-style-type: none"> • The entire contents of the dataset need to be displayed. • The user needs to know which part of the dataset is being viewed, i.e. where they are in the dataset, at all times. • Details about subsets of data items within the dataset are required. • Lack of screen space. • Large amount of data. • Easy navigation between different parts of the dataset.
Solution	<p>Show an overview of the entire dataset together with some visual indication as to which part of the dataset is currently being viewed. Show details about subsets of items in a separate view.</p> <p>The overview can be a scaled version of the main view, i.e. a spatial zoom, or some other representation, i.e. a semantic zoom. Since the overview tends to display a higher number of data items than any more detailed view it is necessary to use simple glyphs that minimise clutter, maximise use of screen space and portray the data attributes most relevant to the task.</p> <p>The overview is usually spatially adjacent to the main view in order to reduce eye-shift.</p> <p>An overview provides several functions, it reduces the visual space the user has to search, it shows the user their current location and therefore context within the data set, and it acts as a navigation aid by helping them to decide which area of the data space to explore next.</p> <p>Typically the visual location indicator can also be used as a selection and navigation mechanism (Navigation Box).</p>
Examples	<p>Windows Explorer™</p>  <p>SeeSoft (Eick et al. 1992)</p>  <p>FilmFinder (Ahlberg and Shneiderman 1994) LifeLines (Plaisant, et al. 1996)</p>
Related Patterns	Navigation Box, Teleportation, Smooth Transitions

Table 4.7: Overview and Detail structure pattern.

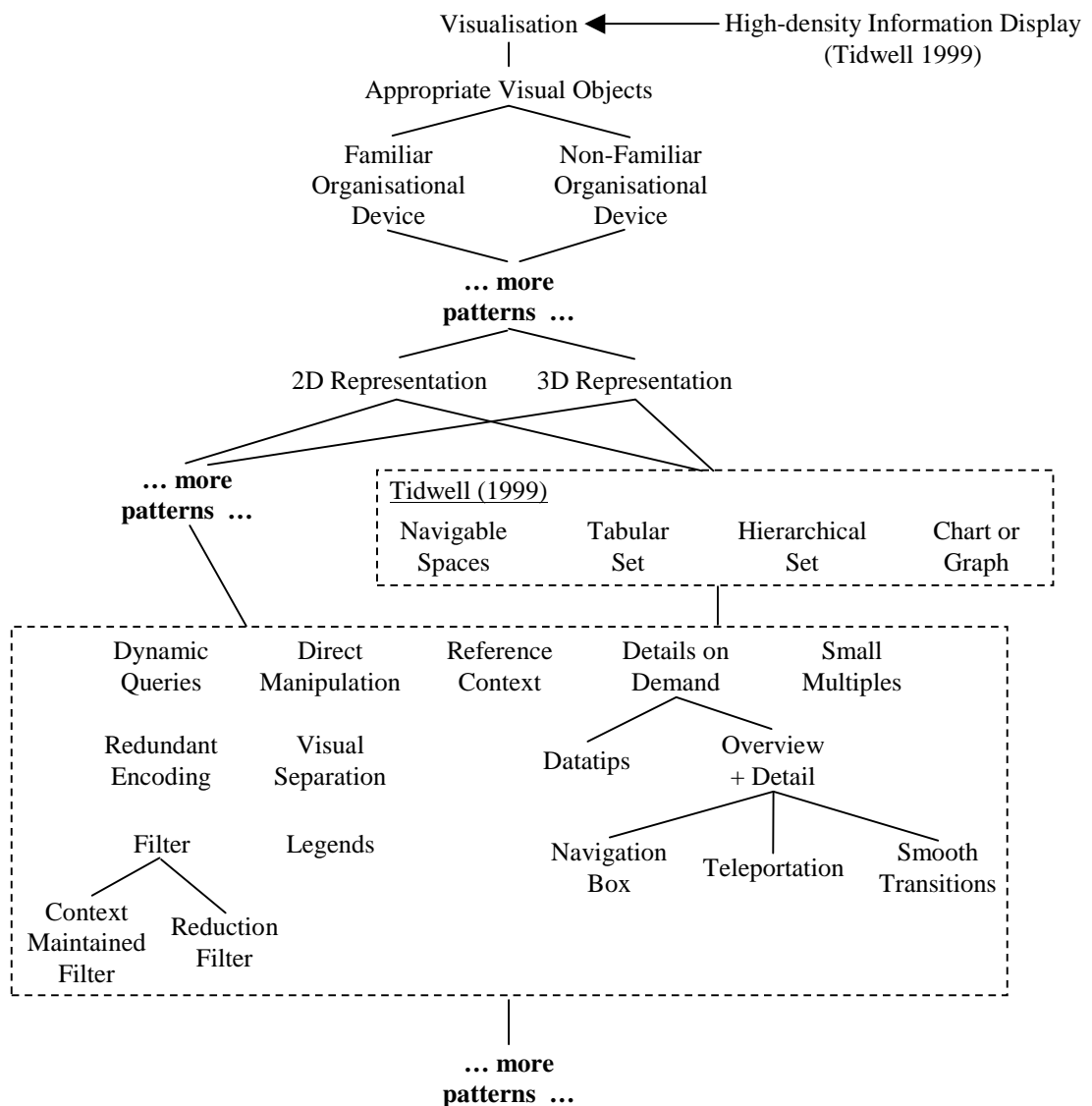


Figure 4.8: Structure pattern language.

The pattern language shown in figure 4.9 consists primarily of interaction patterns although there are links to the SPL. This will be referred to as the *interaction pattern language* (IPL). It is likely that additional patterns exist within the IPL; however, to maintain the clarity of the figure these additional patterns have been omitted. It should be noted that the grey nodes in the IPL are not actually patterns but instead help to define the context for lower level patterns and assist the designer in navigating the pattern language. Note also that some of Tidwell's (1999) GUI design patterns fit within this pattern language e.g. controls can be used to set filter parameter values, scrollbars can be used to view different areas of the data space, etc.

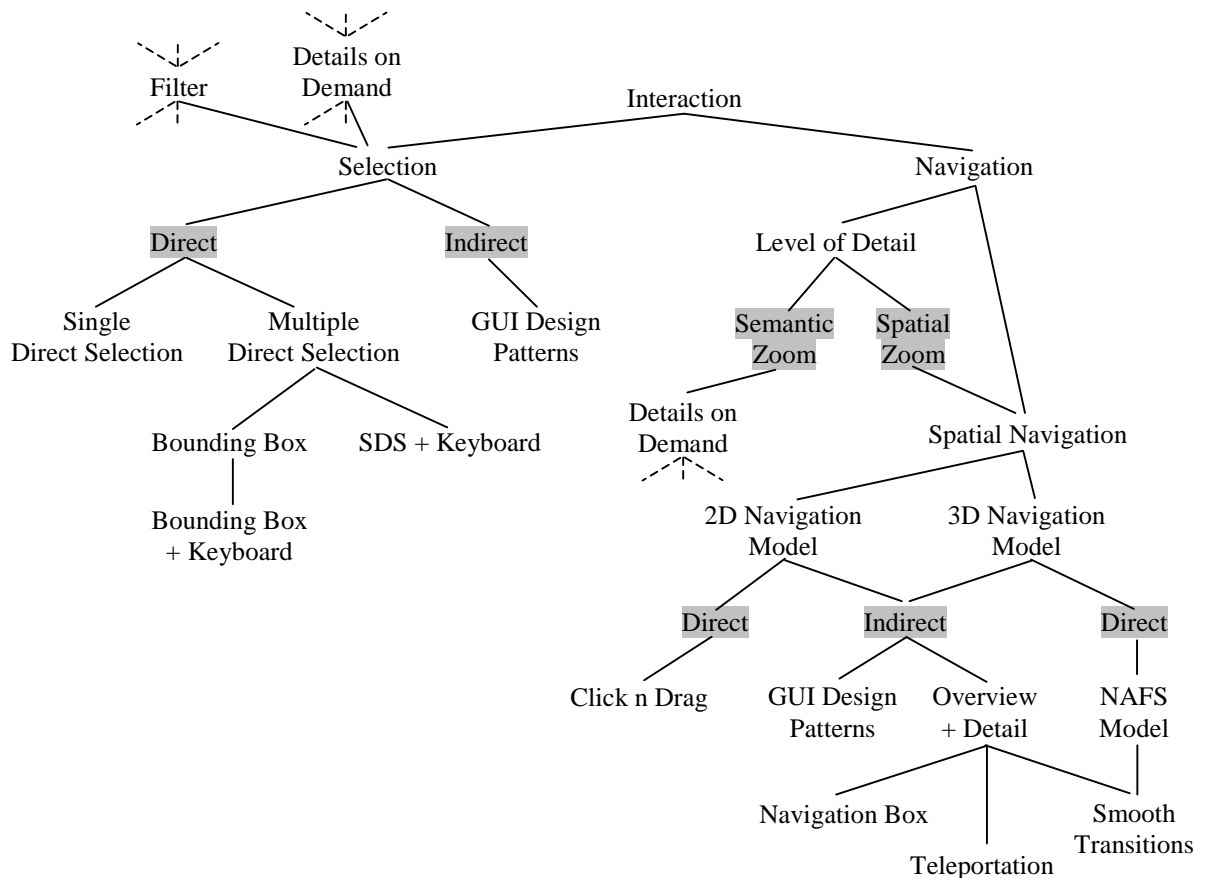


Figure 4.9: Interaction pattern language.

The patterns presented in this research are a first attempt at a set of visualisation specific patterns and are not meant to be exhaustive or definitive. Some of the patterns, such as *Appropriate Visual Objects*, *2D Representation*, *3D Representation*, *Dynamic Queries*, *Direct Manipulation*, *Interaction*, *Selection*, *Level of Detail*, and *Spatial Navigation*, are less concrete than others. Often these more abstract patterns are identifiable by their lack of examples. However, these patterns do help to form a more complete pattern language, often setting the context and forces that apply to lower level patterns.

4.4 Design Methods

The design phase of the methodology highlights the components that need to be designed in order to produce a visualisation. The design of these components is supported by visualisation design patterns, visualisation heuristics, and GUI design patterns. Therefore, the designer is focused on what to design and is supplied, in a convenient format, with effective design knowledge. However, it would be useful if the designer could be guided in their selection of this knowledge.

The methodology we propose is intended to be used by designers who wish to develop novel visualisations. However, it may be difficult for the designer to apply the patterns and heuristics precisely. In addition, applying the patterns and heuristics in isolation is usually not possible. More probably, the designer will be faced with several possibilities and could combine many of the heuristics in a number of different ways. The designer may also wish to explore a number of alternative designs. Therefore, it would be useful if the designer had some means of comparing each design.

We propose methods that will help guide the designer in *design construction* and to evaluate alternative designs in *design evaluation*.

4.4.1 Design Construction

One of the functions of pattern languages such as the SPL and IPL is to help guide the designer to effective design solutions. For example, the designer may have developed a 2D visualisation that has an overview and a detailed view and is unsure of the mechanisms available that allow the user to navigate the data space. By using the IPL the designer can see that the *Navigation Box* and *Teleportation* patterns are the most likely options but in addition *Click n Drag* and *GUI design patterns* are also possibilities. Looking at the context and forces of these patterns the designer can make an informed decision as to which they feel is the most appropriate. The designer may also note that, depending on their choice, the *Smooth Transitions* pattern may also be applied. Thus by using a pattern language the designer can be guided to appropriate design solutions.

Using the components identified in the analysis phase as a guide we propose a small set of questions the designer can ask which will help steer them towards appropriate patterns and heuristics. These questions act as a decision making tool that can be used in addition to the pattern languages. The questions are not intended to guide the designer to fine grain design details but rather to gross structural decisions. Therefore, this design method could be interpreted as a simple model the designer can use to reach a pattern quickly. The solution may then be used as a starting point from which the designer can get a feel for the overall structure of the visualisation and fill in the details using other patterns as indicated by the pattern languages.

Questions relating to the data analysis factors listed in section 4.1.3.2 include:

- Is the data based on a physical model?
- Is the data currently represented in an existing organisational device?
- Does the data have any obvious structure, or can it be related to some obvious structure?
- How many data dimensions need to be displayed at once?
- What is the quantity of data?

4.4 Design Methods

Each of these questions, their solutions, and any additional information such as conditions of use, rationale, advantages, disadvantages, etc., is listed in full in appendix D. An example question and solution is shown in table 4.8.

<i>Data Factor</i>	Quantity
<i>Question</i>	What is the data quantity?
<i>Answer: Large</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Use three dimensions</i> Rationale: Increase in the amount of space available. Disadvantages: Occlusion, navigation issues, depth perception. • <i>Use filters</i> Rationale: Filters can be used to allow the user to reduce the amount of data that needs to be displayed at any one time. Advantage: Simple method for reducing data quantity. Disadvantage: If visual objects are removed from the display then the context of the remaining items may be lost. • <i>Use Overview and Detail technique</i> Rationale: A reduced representation can show the entire dataset while a separate view shows the details of a subset of the data. Advantages: Context preserved, navigation aid, visual search aid.
<i>Answer: Medium</i> <i>Answer: Small</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Use a chart-based representation</i> Rationale: Low data quantities should be compatible with chart-based representations such as scatter plots, bar charts, etc. Advantage: Users are familiar with these types of display. • <i>Use filters</i> See filter solution in large data quantity answer.

Table 4.8: Example data factor question and solution.

Ideally, questions relating to other analysis factors such as task, usability factors, and user modelling should also be developed. For example, a task category question could be “Do humans find the task difficult?” If the answer is ‘yes’ then data transforms may be applicable, if ‘no’ then mapping transforms may be more appropriate. However, although a number of these questions have been addressed as part of this research they have not been made explicit due to time limitations. In addition questions regarding other analysis factors such as the usability factors have been covered extensively in existing HCI literature.

4.4.2 Design Evaluation

We propose three different methods for evaluating designs. The first of these methods is based on Salisbury’s (2001) ranking of data type to visual feature, the second is based on the classification of the visualisation heuristics in terms of their effect on usability, and the third uses Brath’s (1997a, 1997b, 1999) metrics. Together these three methods help to corroborate effective designs by looking at low-level perceptual features, usability issues and cognitive issues.

4.4.2.1 Mapping Evaluation

Recall that Mackinlay (1986) proposed a ranking of data type to visual feature. This ranking has been used by several automatic visualisation generators. However, recently Salisbury (2001) has proposed a slightly different ranking, based on Mackinlay's, but modified to take account of data collected during Salisbury's own user studies. A comparison between the two rankings is shown in table 4.9.

	Quantitative		Ordinal		Nominal	
	<i>Mackinlay</i>	<i>Salisbury</i>	<i>Mackinlay</i>	<i>Salisbury</i>	<i>Mackinlay</i>	<i>Salisbury</i>
0	Position	Position	Position	Position	Position	Position
1	Length	Length	Cl. Brightness	Cl. Intensity	Colour Hue	Colour Hue
2	Angle	Angle	Cl. Saturation	Colour Hue	Texture	Texture
3	Slope	Slope	Colour Hue	Texture	Connection	Connection
4	Area	Area/Volume	Texture	Text	Containment	Containment
5	Volume	Cl. Intensity	Connection	Connection	Cl. Brightness	Shape
6	Cl. Brightness	Text	Containment	Containment	Cl. Saturation	Text
7	Cl. Saturation		Length	Length	Shape	Length
8	Colour Hue		Angle	Angle	Length	Angle
9			Slope	Slope	Angle	Slope
10			Area	Area/Volume	Slope	
11			Volume		Area	
12					Volume	

Table 4.9: Comparison between Mackinlay (1986) and Salisbury (2001).

Ranking of data type to visual feature. Key: Cl. = Colour.

In the same way that automatic visualisation generators assign the most appropriate visual feature to data type mapping using the ranking in table 4.9, we propose to use the rankings as a measure of the relative perceptual effectiveness of different visualisations. Each visual feature is assigned a value based on its position in the table. The designer can then simply add up the values based on the assignment of data type to visual feature used in the visualisation. The lower the score the more perceptually effective the visualisation should be.

As a simple example consider, a visualisation in which a quantitative data type is mapped to position and a nominal data type to colour hue, the total score is 1 (i.e. 0+1). However, if the mapping is changed so the quantitative data type is mapped to length, the score becomes 2 (i.e. 1+1). Therefore, the first mapping should be perceptually more effective.

4.4.2.2 Visualisation Heuristic Evaluation

The design phase of the methodology proposed relies partly on using visualisation heuristics to guide the designer to effective designs. However, designers may often use combinations of heuristics in different ways to produce different designs that support the same tasks. For example, a designer may choose a 3D display with smooth animation and direct navigation, a

second design in which an overview is used to provide context and navigation, a third alternative which uses filters and specially designed navigation controls, and so on. We propose that the visualisation heuristics can be used to evaluate alternative designs based on their usability. We have referred to this as an heuristic evaluation although it should be noted that it does not meet Nielsen's (1994) definition.

Firstly the designer should assign a score to each heuristic based on how well they feel it has been implemented. The score assigned will be referred to as the *heuristic score* (H_s) and ranges between -1 and 2 inclusive. The reason a score of -2 is not included in scoring system is that a heuristic is either is or is not violated, it cannot be violated by a small amount. This is different to not applying the heuristic at all, which is given a zero score. In contrast, a heuristic can be implemented to a greater or lesser extent, or may only apply to certain aspects of the design. Therefore, the +1 and +2 scores are justified. In addition, the scoring system has deliberately been kept simple due to the complexity of the factors involved and in accordance with the simple model used to classify the heuristics. For each heuristic the scores should be assigned as shown in table 4.10.

Score	Description
-1	the visualisation violates the heuristic
0	the heuristic has not been applied or is not applicable
1	poor implementation, problems remain, or the heuristic only applies to certain aspects of the visualisation e.g. certain data dimensions.
2	good implementation

Table 4.10: Heuristic score values and descriptions.

It should be noted that some of the scores are not applicable to certain heuristics. For example, the overview and detail heuristic can never really be violated, it is either implemented or it isn't. However, since this is true across all visualisations it should not unduly affect any ratings.

At this point the designer should have a matrix of heuristics, visualisations, and scores, similar to that shown in table 4.11. The designer can use this heuristic score matrix to see the deficiencies of each visualisation and consequently areas that need to be improved. It can also be used as a very rough guide in deciding which visualisation designs, if any, should be abandoned. If necessary, cells in the matrix could be colour coded to highlight positive and negative aspects of each visualisation or the matrix could be converted to some simple chart-based representation.

#	Heuristic	Vis_1	Vis_2	Vis_3	Vis_4
1	Use a real world physics model	0	1	1	0
2	Visually refer all graphical objects to a reference context	2	0	-1	2
3	Use connotative mappings	1	-1	0	1
4	Use an organisational device the user already knows	0	2	0	2
...
31	Multiple linked (co-ordinated) views	2	-1	0	0
32	Direct manipulation	2	2	0	0

Table 4.11: Heuristic score matrix.

It was established in chapter 3 that usability is of key significance to visualisation design. This led to the assignment of a set of values to each heuristic based on the heuristics effect on each usability factor. These will be referred to as *heuristic usability factor effect* (H_{ufe}) values. To allow designs to be rated the qualitative values used in table 4.5 have been assigned numeric values as follows (+ \rightarrow 2), (+&- \rightarrow 1), (0, ? \rightarrow 0) and (- \rightarrow -1). The reason that (+&-) has been assigned a value 1 is that it is felt the negative effects of the heuristic can be overcome by training. However it may be equally valid to assign 0 or -1.

Before rating each design the designer must first assign normalised weights to each of the usability factors. In effect the designer is saying how important each of the usability factors are with respect to the types of users, the business domain, and so on. This will be referred to as the *usability factor weight* (UF_w). For example, the designer may assign weights as in table 4.12.

Usability Factor	Weight
Time to learn	1.0
Speed of performance	1.0
Rate of errors	0.5
Retention over time	0
Subjective satisfaction	0.5

Table 4.12: Example usability factor weight assignments.

Each visualisation design is then assigned an overall rating using the following equation:

$$V_{rate} = \sum_{\text{for each u.f.}} (UF_w * \sum_{\text{for each h.}} (H_s * H_{ufe}))$$

Key:
 V_{rate} : visualisation rating.
 UF_w : usability factor weight.
 H_s : heuristic score.
 H_{ufe} : heuristic usability factor effect.

The rating for each visualisation can then be compared and the visualisations ranked according to their scores. The visualisation with the highest score can be considered the most usable with respect to the prioritised usability factors. However, it should be made clear that the ratings are not on a scale and can only be used qualitatively. For example, if there are three visualisations A, B, C, with ratings 5, 10, 50, it cannot be said that visualisation C is ten times more usable than visualisation A, but it may be possible to say that visualisations A and

B have roughly the same level of usability, and that visualisation C should be much more usable than A and B.

4.4.2.3 Metric Evaluation

A heuristic tells the designer what features should or should not be included as part of a design. Metrics are similar but allow the quantitative measurement of various features of a visualisation. In this way they can be used to rank alternative designs.

Brath (1997a, 1997b, 1999) presents a number of metrics that can be used to measure different features of a visualisation. We propose that these metrics can be used as a way of corroborating the predictions made by the mapping and heuristic evaluation techniques.

The first of Brath's metrics uses a simple model of the cognitive effort required to recall the mapping from data dimension to visual feature. Each mapping falls into one of four categories:

- *n-to-1 mapping*: Several data dimensions are mapped to a single visual feature. So for example in table 4.13 the different colour hues are used to represent profit and loss due to stock, profit and loss due to the exchange rate, and highlighting.






	Positive profit due to stock
	Negative profit (loss) due to stock
	Positive profit due to exchange rate
	Negative profit (loss) due to exchange rate
	Highlighting

Table 4.13: Example of n-to-1 mapping.

(Modified from Brath (1999))

- *1-to-1 general mapping*: This is the most common mapping found in visualisations, where each visual features represents a single data dimension.
- *1-to-1 intuitive mapping*: Some mappings are more connotative than others. For example a data dimension that represents the 'size' of something maps well to the visual feature size. In essence, the visual feature implies the data dimension thus the effort required to recall the mapping rule is reduced.
- *Pre-existing representation*: Some representations such as bar charts, scatter plots, maps, etc., are commonplace and can therefore be easily decoded.

Brath assigns each of the four categories a score, as shown in table 4.14, with lower scores indicating the mapping requires less cognitive effort to recall. This *dimensional score* can then be used to compare different visualisation designs.

Mapping	Score	Reason
n-to-1	3 x n	<ul style="list-style-type: none"> • 1 point to recall the data dimension • 1 point to recall the visual feature • 1 point to recall the mapping rule for that dimension multiplied by the number of data dimensions, since the recall is required for each data dimension that uses this feature
1-to-1 general	2	<ul style="list-style-type: none"> • 1 point to recall the data dimension • 1 point to recall the visual feature
1-to-1 intuitive	1	<ul style="list-style-type: none"> • data dimension implied by visual feature
Pre-existing representation	0	<ul style="list-style-type: none"> • no cognitive effort required to recall mapping rule

Table 4.14: Dimensional score mapping assignments.

Brath defines several other metrics including:

- *Number of data points*: The number of discrete data values presented on screen at an instant. For example a 2-axis scatter plot that shows 20 points has 20*2 coordinate pairs therefore 40 data points.
- *Data density*: number of data points / number of pixels in the display (not including window borders, menus, etc.).
- *Number of simultaneous dimensions*: The number of different data dimensions displayed simultaneously.
- *Maximum number of dimensions from each separable task representation*: The number of dimensions is not a useful measure when comparing multiple view visualisations. This measure calculates the number of dimensions for each separate representation based on the task. For example, a six dimensional dataset can be represented using three separate scatter plots. If only two dimensions within a scatter plot need to be compared at any one time then the maximum score is 2. However, if a task requires correlating between all three scatter plots simultaneously then the maximum score is 6.
- *Percentage of occlusion*: the number of data points completely obscured / the number of data points. It should be noted that this does not address partial occlusion.
- *Percentage of identifiable points*: number of visible data points identifiable in relation to every other visible data point / (number of visible data points)²

Unfortunately Brath's metrics do not take into account interaction, which may have a significant effect on some of these measurements. Since interaction is present in nearly all visualisations we believe that some of these metrics need to be altered so that they measure ranges of values. For example, interaction can often reduce occlusion or even make it worse, thus measuring the minimum and maximum percentage of occlusion may be more valuable. It is also not clear from Brath's research how some of the metrics can be applied, even if interaction is ignored. For example, it is not clear how 'percentage of occlusion' and 'percentage of identifiable points' can be measured if the contents of the dataset are dynamic. In addition, it appears that these metrics can only be accurately measured if the visualisation

has already been implemented. This is of little use to designers who are trying to decide which design to proceed with before implementation. Therefore, the metrics may not be strictly applied during the evaluation of any visualisations developed during this research.

4.5 Summary

The basic methodology, shown in figure 4.10, consists of five main stages, *information gathering, analysis, design, implementation, and evaluation*. Four of these stages, together with *iterative design*, are consistent with the generic design methodology presented in section 3.2.4. The first stage, information gathering, is specific to visualisation and is necessary since by definition visualisation is the visual representation of data and therefore data must at least be collected and structured before the design of the visualisation can proceed.

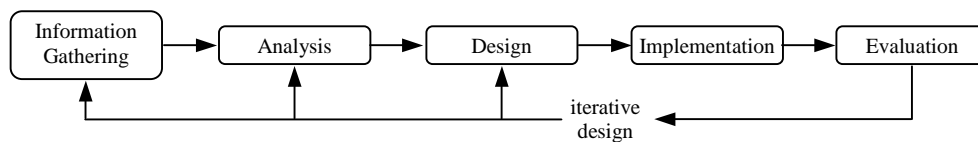


Figure 4.10: Stages of the methodology.

Using the factors that influence visualisation design identified in chapter two, the HCI techniques reviewed in section 3.2.2, and the visualisation reference model developed in section 3.3 (figure 3.8), the information gathering, analysis, and design stages have been further decomposed into a number of sub-components. These sub-components serve to breakdown the design process and allow the designer to focus on specific tasks within each stage.

In order to define the structure and content of the visualisation, the tasks that the visualisation must support, the target users, and to set the usability goals, the designer must analyse the data, the tasks, the usability factors, and the user. Using the information gathered during analysis the designer must then define the appropriate transforms and interaction mechanisms. Essentially this will define the graphical scene and how the user interacts with it. If necessary, the designer may also need to consider how multiple visualisations will be organised and communicate with each other.

The analysis, design, and implementation stages are each supported by patterns as shown in figure 4.11. The designer can use the business domain patterns and business process patterns to describe the business, its goals, and typical processes. In addition the task patterns are used to capture knowledge about the tasks. Together this knowledge helps to define the business goals, the system requirements, and potential interaction mechanisms.

4.5 Summary

The design stage is supported by visualisation design patterns, visualisation heuristics, and GUI design patterns, each of which help guide the designer to effective design solutions. The visualisation design patterns have been developed as part of this research and are derived from the recurring solutions seen in the visualisations described in chapter two and the visualisation heuristics, also collected as part of this research. The visualisation design patterns capture specific visualisation design knowledge and are complemented by the GUI design patterns, which relate to the surrounding GUI components.

The implementation stage makes use of software design patterns as a way of developing robust systems that can be easily maintained and extended.

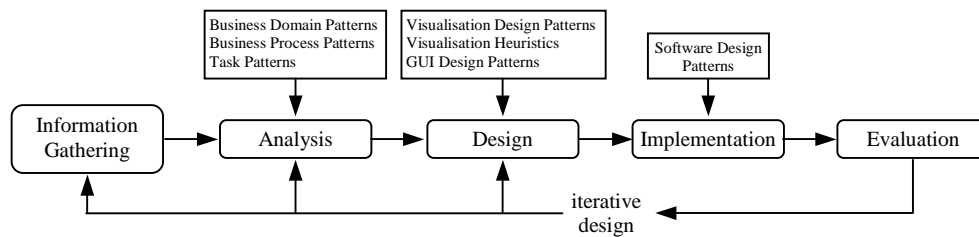


Figure 4.11: Stages of the methodology including supporting patterns.

Elements of the usability engineering life cycle can be applied throughout the methodology as shown in figure 4.12. Knowing the user, competitive analysis, and setting usability goals are achieved in the analysis of the tasks, users, and usability factors. Actively involving the users in the design and evaluation stages (i.e. participatory design), prototyping, and iteratively refining the design, help to ensure the usability of the product. The design stage can also take advantage of the visualisation patterns and visualisation heuristics both as guidelines and for heuristic analysis. Having developed a series of prototypes a more formal software engineering methodology may then be used to implement the final product. The evaluation of both the prototypes and the final product can use the most appropriate technique depending on the situation e.g. cognitive walkthroughs, heuristic evaluation, etc. Once the final product is in use, additional feedback can be collected via software logs, monitoring help desk calls, sending out surveys, etc., this data can then be used to improve the usability of future systems.

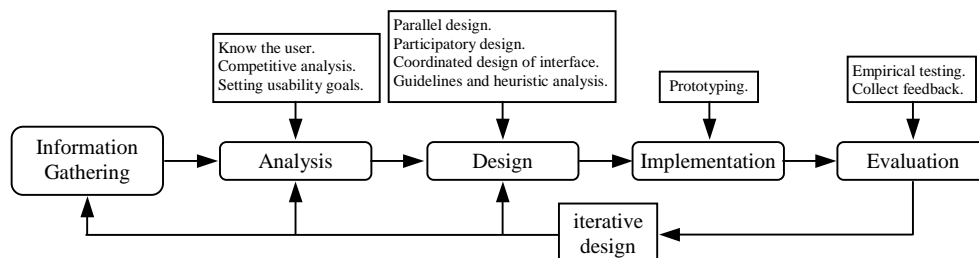


Figure 4.12: Methodology including usability engineering life cycle elements.

4.5 Summary

To assist the visualisation designer in the evaluation of alternative designs three different design evaluation techniques are used. These are a mapping evaluation, heuristic evaluation, and metric evaluation. Together these techniques help to corroborate effective designs.

Figure 4.13 shows how the stages of the methodology, the patterns that support each stage, the usability engineering life cycle, and the design evaluation techniques, fit together to form the complete methodology.

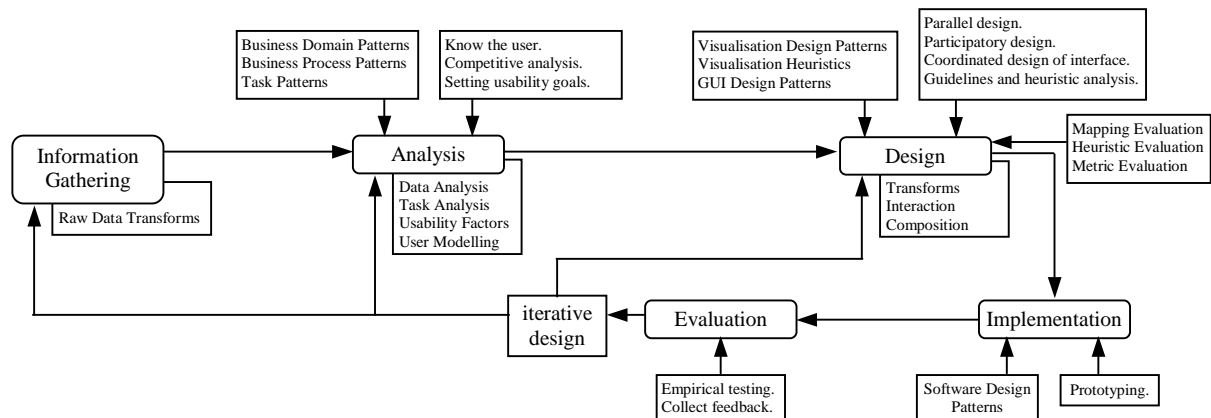


Figure 4.13: The complete pattern supported visualisation design methodology.

To sum-up, in this chapter we have presented an iterative methodology that covers all aspects of the design process. Using the knowledge gained from chapters two and three we have identified the factors relevant to visualisation design and attempted to support the designer at each stage by providing them with experience-based design knowledge presented in pattern form. To complement these patterns we have also collected a set of visualisation heuristics that the designer can also refer to during the design stage. Recognising the importance of designing user-centred and usable systems we have classified these heuristics by their effect on various usability factors and developed a heuristic evaluation technique that the designer can use to rank alternative designs.

Chapter 5: APPLICATION OF THE METHODOLOGY

This chapter explains how the methodology and the design evaluation techniques described in chapter four have been applied in order to develop a set of visualisations for military command and control. It begins with a brief description of the motivations behind the development of these visualisations, which is expanded upon throughout the chapter. There then follows a description of the command and control domain and the analysis of that domain in order to determine the relevant data and typical tasks. This is followed by a description of the visualisations designed to support these tasks and the predictions made by each of the design evaluation techniques.

5.1 Motivation

Ideally we would like to empirically evaluate the methodology proposed in chapter four. One approach would be to develop two visualisations, one using an existing methodology and another using the proposed methodology. During the development of each visualisation various measures would need to be taken relating both to the methodology (e.g. cost, time, developer satisfaction, etc.) and to the final product. To do this correctly would require a complete development team including managers, visualisation experts, software engineers, HCI specialists, and users. Even if this were possible, we could only compare the relative merits of each methodology based on the measures taken. This may be useful but it is not the purpose of this research to determine whether or not one methodology is better than another. Instead, we are proposing a methodology that is based on a generic framework (described in section 3.2.4) and that uses and extends well-established ideas and techniques (described in chapters two and three) and should therefore lead to the development of visualisations with high usability. Hence we have concentrated on key aspects of the proposed methodology.

We are interested in finding evidence that supports the overall process involved, the use of heuristics and patterns to provide design knowledge at each stage of the methodology, and the use of several design evaluation techniques as a way of corroborating effective designs. Of specific interest is the classification and use of visualisation heuristics as a way of ranking alternative designs. In order to gather this evidence it is necessary to develop and evaluate a set of visualisations that make use of the design knowledge and techniques indicated by the methodology. The remainder of this chapter describes these visualisations and the predictions made by the design evaluation techniques presented in chapter four. Chapter six then describes how the visualisations were empirically evaluated and compares the results of these evaluations to the predictions made in this chapter.

5.2 *Command and Control*

Command and Control (C^2) can be broadly defined as the co-ordination and direction of resources to achieve some goal. Typical examples of C^2 domains include military campaigns, air traffic control, financial management, pharmaceutical processes and disaster management. From these examples, the resources could include aircraft, weapons, personnel, money, medical supplies, and the goals may be to destroy a target, direct aircraft to a free runway, purchase stock, etc.

C^2 domains are typically complex environments involving large quantities of dynamic data. C^2 users are often required to monitor these environments and take appropriate actions based on the information available. This makes C^2 a domain in which visualisation tools may be useful in assisting C^2 users in their monitoring and decision-making activities.

The reason for choosing the command and control domain is primarily due to our connections with QinetiQ, previously known as the Defence Evaluation Research Agency (DERA). QinetiQ have long been associated with the development of military hardware and software and have identified the need for decision-making tools that can help personnel in military campaign analysis. For these reasons the development of novel visualisations for the military command and control domain was thought to be appropriate.

5.2.1 *Military Context*

Military operations are based on a 72hr cycle, which is split into three 24hr segments. In the first 24hrs Intelligence nominate all required targets. Each target may be broken down into sub-targets, for example the primary target may be an airbase which is then decomposed into hanger, bunker, control tower, airstrip, etc. This process produces a target nomination list for each day of the operation.

The next 24hrs of the cycle are used for planning. Planners take the target nomination list and determine what resources/assets are available at that time, how many to use, how to deploy them, etc., in order to achieve some of the targets nominated by Intelligence. Each of the plans developed may then be divided into sub-plans that may then be further divided until basic mission units are formed. The plan stage of the cycle brings together intelligence and resource management.

The final 24hrs are known as the *combat operations* stage. In this stage the missions developed in the plan stage are executed. However, the plan stage missions are ideal plans, which, because they are formulated 24hrs before execution, may use resources that are no longer available. In this case combat operations personnel are required to develop equivalent

plans ‘on the fly’, i.e. dynamically at the time the plan should be executed, using the resources currently available.

It should be noted that several C^2 visualisations have already been developed. However, the majority of these have concentrated on providing C^2 personnel with information relating to the position and movement of friendly and enemy resources such as aircraft, troops, etc., as well as detailed 3D terrain images. This is known as *situation awareness*, an example of which can be seen in figure 5.1. Although very sophisticated and capable of providing detailed spatial information, these systems do not assist the combat operations personnel in basic resource management and plan development. Instead these personnel rely on simple representations such as tables, charts, and paper-based maps. However, due to the increasing complexity of the military environment and the need to make rapid and accurate decisions, these representations are no longer suitable. Therefore this is an area in which novel visualisations that support combat operations personnel in their decision making process can be of benefit.

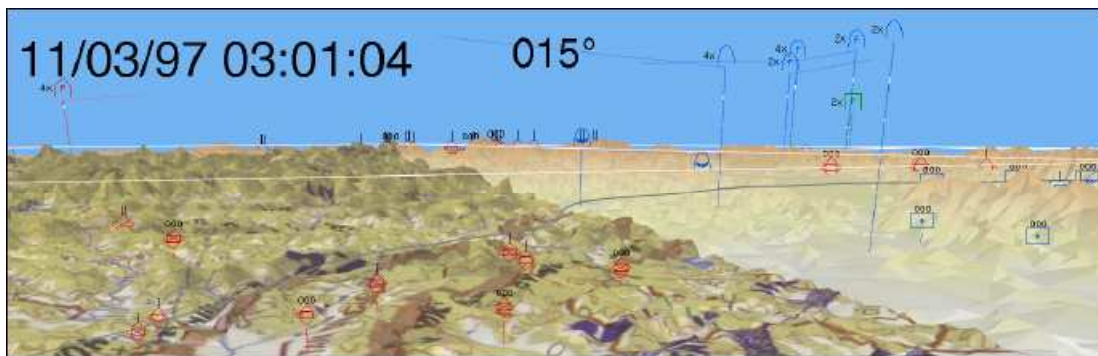


Figure 5.1: Situation awareness visualisation.
(Feibush et al. (1999)).

5.2.2 Military Task Requirements

In order to develop visualisations that can help support command and control teams it is necessary to understand and define the types of tasks that such teams carry out. In particular, the types of questions that are asked by planners and more importantly the combat operations personnel who are trying to execute the plans.

The basic premise is that combat operations are given a set of mission plans that must be executed within some time frame. However, problems can occur with plans in one of two ways. A plan may make incorrect assumptions about various aspects of a mission or a plan may start to fail due to some unforeseen event occurring during the execution of a mission. In either case it will be necessary for combat operations to develop new plans ‘on the fly’ and to try to judge the impact of these modifications to the overall campaign. It is hoped that by

5.2 Command and Control

using visualisations, plan problems can be identified more easily, new plans can be developed more rapidly, and their consequences can be more easily assessed.

Once the tasks that combat operations perform are understood and the relevant data used during the tasks extracted, visualisations can be developed that represent the information in the most appropriate way. This may then lead to an increase in combat operations performance.

Initially a set of military scenarios were developed and confirmed with military personnel. These scenarios were then analysed in order to identify relevant task objects. These scenarios consist of a number individual mission plans typical of those executed by combat operations. An example scenario is shown in table 5.1.

Bombing Run Scenario

Bomber (BM1) leaves base (B1) at time 0. Two fighters (F1&F2) leave base (B2) at time 1. The two fighters rendezvous with the bomber at time 2, location ER. All three aircraft head for target (TGT1), the fighters defending the bomber along the route as required. At time 3 the three aircraft arrive at TGT1. Bomber BM1 deploys 75% of its bombs while the fighters protect it. Also at time 3 a tanker (T1) leaves its base (B3) and heads towards rendezvous point RR. At time 4 the two fighters head back towards their home base (B2) and the bomber heads towards rendezvous point RR. At time 5 the bomber (BM1) and tanker (T1) rendezvous at RR and refuelling takes place. After refuelling the bomber heads towards target (TGT2) and the tanker returns to its home base (B3). At time 6 the bomber BM1 deploys 25% of its bombs on TGT2 and then heads for its home base (B1). At time 8 all aircraft have returned to their home bases. At time 9 a reconnaissance aircraft (RA1) leaves base B1. At time 10 RA1 arrives at TGT1 to photograph damage inflicted on target. At time 11 RA1 returns to base B1.

Resource	Time	Location	Action
BM1	0	B1	Leave base.
F1	0	B2	At base.
F2	0	B2	At base.
T1	0	B3	At base.
RA1	0	B1	At base.
F1	1	B2	Leave base.
F2	1	B2	Leave base.
BM1	2	ER	Escort rendezvous point.
F1	2	ER	Escort rendezvous point.
F2	2	ER	Escort rendezvous point.
BM1	3	TGT1	Bomb TGT1 (75% bombs).
F1	3	TGT1	Protect BM1.
F2	3	TGT1	Protect BM1.
T1	3	B3	Leave base.
F1	4	B2	Returned to base.
F2	4	B2	Returned to base.
BM1	5	RR	Refuel using tanker T1.
T1	5	RR	Refuel bomber BM1.
BM1	6	TGT2	Bomb TGT2 (25% bombs).
T1	7	B3	Returned to base.
BM1	8	B1	Returned to base.
RA1	9	B1	Leave base.
RA1	10	TGT1	Photograph TGT1.
RA1	11	B1	Returned to base.

Bombing run event table.

Table 5.1: Typical military scenario used for task analysis.

Analysing these scenarios led to the following task object definitions:

- *Mission*: A mission consists of a set of resources, a route those resources take, an objective and a set of timing constraints to which the mission must adhere.
For example, a fighter using a set of air-to-ground missiles must takeoff at 12.00, reach a SAM site at 13.00, destroy the SAM site and then return to base by 14.20. In this case the resources are the fighter, its fuel, its pilot, its ammunition, etc., and the air-to-ground missiles. The route is the path that the fighter must take to reach the objective, which is the SAM site. This must all be accomplished in the specified time frame. It should be noted that the objective is the action to take against a specified target, so for this example the objective would be destroy and the target would be the SAM site.
- *Resource*: A resource or asset is any object that can be used as part of a mission to help achieve the mission objective. A resource consists of a set of attributes such as its type, start location, current location, direction, destination, estimated time of arrival, status, financial cost, etc. Metadata such as the quantity of a resource in stock, the minimum acceptable quantity, etc., can also be derived about resources. In addition, a resource may include a set of sub-resources. For example, the primary resource may be a fighter, with the set of sub-resources being the armaments the fighter uses, its fuel, the pilot, etc.
- *Route*: A route consists of a set of waypoints and a set of regions that the route crosses. Each region within a route has the following attributes:
 - *Type*: The type of a region can be friendly, enemy, neutral, etc.
 - *Location*: A set of co-ordinates defines the location of the region.
 - *Area*: Each region covers a particular area.
 - *Strength*: The strength of a region, or more precisely the forces within a region is defined by the size of the force, the type of force, the threat that force represents and so on.
 - *Sensitivity*: Objects within a region may be considered sensitive. For example a school or hospital is a sensitive object.
 - *Weather*: The weather within a region can be defined by its type e.g. blue sky, stormy, etc., whether it is crossed in the day or at night and may possibly include the forecast for that region.
- *Objective*: An objective (or target) represents the goal of a mission. An objective has a status (e.g. active, destroyed), a priority, a type (e.g. airfield, communications, radar, industrial factory, SAM site, etc.), and a location. An objective may also have coverage, for example a SAM site has an associated effective range of fire.
When looking at a target it may be necessary to consider surrounding objects, for example if a weapons factory is next to a hospital it may be classified as a sensitive target. In this case it is important to make sure that the correct resources e.g. bomb type, are used against the target.
- *Timing Constraints*: The timing constraints define when each part of a mission should occur and at what time each resource should be at a specified location. Rather than stating an exact time at which a resource should be at a location, instead a time window is

defined. This allows a resource to be at a specified location within a certain period of time, for example 12.00 plus or minus 10 minutes. Time windows allow for *slippage* within the mission plan.

- *Mission Package*: A mission package consists of a set of individual missions. These missions may be independent or they may rely on each other in some way.
- *Mission Interdependence*: Several missions within a mission package may be interdependent. For example, a bomber may need to be refuelled by a tanker at some point in its mission. Therefore the bomber mission depends on the tanker mission to achieve its objective. The criticality of a mission can be thought of in terms of how many missions are dependent upon it within the package and the campaign as a whole. The relative priority and importance of dependent missions also contributes to the criticality of a mission. In addition, resources can also be reused between missions and mission packages as long as timing constraints permit it.

Further analysis of the mission scenarios led to a generic set of common problems experienced by combat operations teams. These are as follows:

- *Pre-mission*
 - A particular resource is unavailable at the start of the mission. Either there are no resources of this type left, or there are none that will be in a ready state by the start of the mission.
 - A particular resource is not capable of fulfilling the requirements of the mission in some way. For example it cannot travel the distance indicated without being refuelled.
 - Certain timing constraints cannot be met, for example a resource must be in two locations at the same time.
 - The regions along the route planned have changed ownership, which may make them too risky for the current mission.
 - The target/objective is no longer at the location specified.
- *During-mission*
 - Timing constraints cannot be met. For example, there is a strong headwind that may slow an aircraft down, or, an aircraft is damaged by flak and therefore cannot operate at peak efficiency.
 - The target/objective is no longer at the location specified.
 - A particular resource fails on route.

Using the military scenarios, the task objects, and the common problems experienced by command and control teams, a series of task scenarios were developed. Each task scenario describes a typical problem that may arise during the execution of one of the military scenarios described earlier. The task scenarios were used to confirm the types of tasks combat operations personnel perform and in particular the types of questions they ask themselves

5.3 Visualisation Designs

when trying to solve mission plan problems. For completeness the reasoning behind the questions was also discussed. A typical task scenario is shown in table 5.2.

Situation	One of the problems with a mission plan is that a SAM site exists along the route planned.
Task	Assess the reliability of this information using the intelligence reports. How reliable is the information that indicates there is a SAM site at the specified location? Rank the reliability as one of the following: <i>very reliable</i> , <i>fairly reliable</i> , <i>poor reliability</i> , <i>unreliable</i> , <i>don't know</i> .
Sub-tasks / Questions	<ul style="list-style-type: none">• Find the report(s) that relate to SAM sites in the specified location.• Find reports of other activities in the area that may provide supporting evidence for a SAM site being located in the area. For example, vehicles heading in towards the site, aircraft destroyed in the region, etc.• Find reports that may provide contradictory evidence. For example reports stating that there is no unusual activity in the area.• Find other reports by the same author(s).• How has the rate of submitted reports changed over time?
Reasoning	To determine the reliability of a report you must either find evidence to support the report or evidence of contradictory information. To do this you can look for other reports that explicitly state that a SAM site is in the area or you can look for reports that describe enemy activity in the area, which may indicate a SAM site, or other installation. For example if reconnaissance shows a major infrastructure in the region, or if supply vehicles are often seen heading in the appropriate direction, this may indicate the presence of some installation. Similarly if friendly aircraft are regularly lost in the region this may also indicate the presence of something like a SAM site. Contradictory reports may state no activity in the region. The reliability of these contradictory reports may also need to be investigated.

Table 5.2: Example task scenario.

The complete set of military and task scenarios can be found in appendix E.

The task objects and the task scenarios form part of the analysis stage and are vital to the development of any visualisation.

5.3 Visualisation Designs

5.3.1 Motivation

Before describing the visualisations produced it is important to discuss the motivation behind them. Unlike generic presentations such as bar charts and scatter plots, which have a fixed structure, the domain often significantly influences the design of a visualisation. In this case any designs considered must be capable of supporting the tasks described in section 5.2.2. It is rare for a visualisation to support a single well-defined task, in which case an algorithm may be more effective. Therefore, to add realism and increase design complexity we decided to make each visualisation capable of supporting multiple tasks.

The representations currently used by command and control teams consist of tables of data and simple chart and map-based displays. There is some debate as to the need for more complex visualisations in such an environment or the form such visualisations would take. In

In addition it was unclear whether or not command and control personnel would even accept visualisations as viable tools to help them in their decision-making. This led to the design of a diverse range of novel visualisations, some of which were based on the more traditional presentations already used, some based on spatial representations, and some completely abstract. A diverse range of visualisations also helps to cover a variety of heuristics and gain insights into different visualisation techniques.

A subjective evaluation was performed with command and control personnel to determine which visualisations were acceptable and also to confirm the appropriateness of the task scenarios we had developed. To achieve this a series of cognitive walkthroughs were conducted (described in section 6.2) using the prototype visualisations presented in this section. These visualisations would be improved upon later and used in more rigorous trials. The design of these trials is described in detail in section 6.3.

The remainder of section 5.3 presents six visualisations designed with these motivations in mind. Each visualisation is designed to assist command and control personnel in their decision-making process when faced with tasks similar to those outlined in the task scenarios. The visualisations are described in terms of their construction, interaction facilities, and the primary tasks that they support. The visualisations are Resource Checks, Parallel Coordinates, Route Checks, Organix, Mission Execution, and Mission Dependencies.

5.3.2 *Resource Checks*

The Resource Checks visualisation combines both map-based and table-based displays. The map part of the visualisation shows a small set of bases together with their associated resources. Each base is at a fixed distance away from a specified location represented by a cross at the centre of the map. Also displayed are a series of concentric circles that can be used to estimate relative distances. Resources are represented as small circles of varying sizes and colours clustered around the base centre. Bases have a low, medium, or high quantity of each resource, which is mapped to three different circle sizes. Each resource has an associated power level, ranging from 1 to 5, which is represented by the circle colour. A cross placed inside the circle is used to indicate that a resource is non-operational. Resources can be selected by dragging a bounding box around them and are highlighted by increasing the thickness of the resource circle. Details about individual resources are shown in the table part of the visualisation when they are selected. Looking at figure 5.2 all the resources at base 1 have been selected.

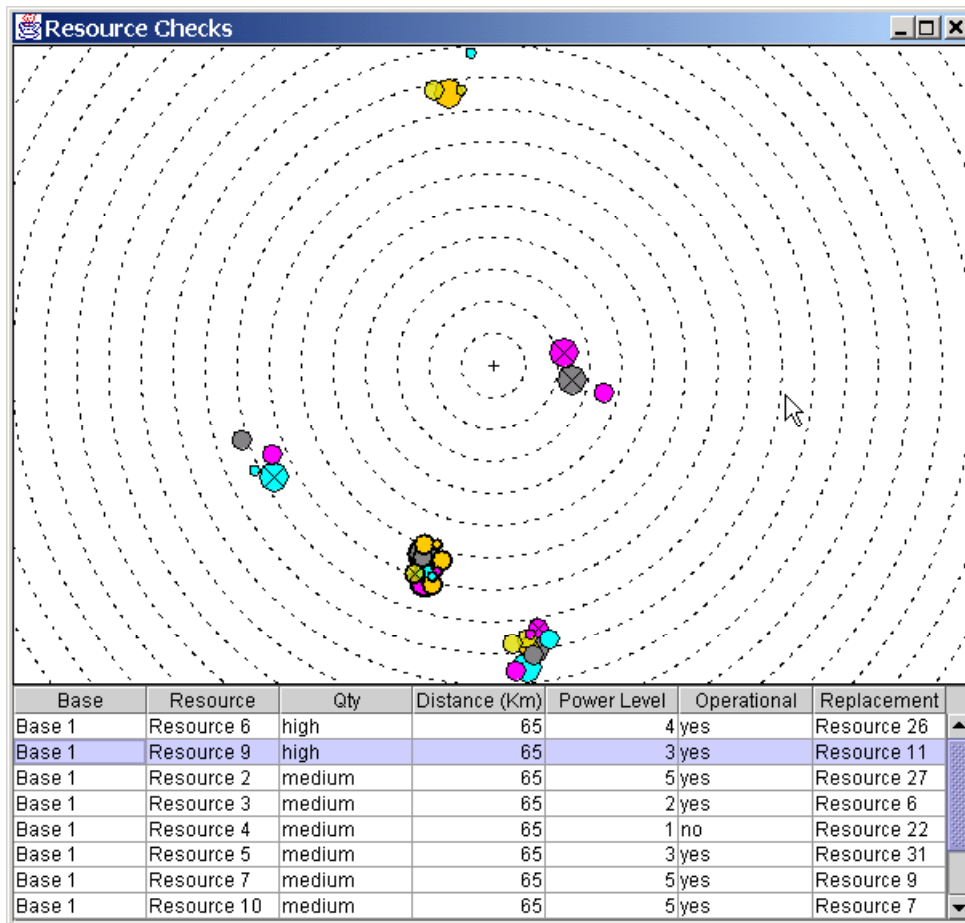


Figure 5.2: Resource Checks visualisation.

The Resource Checks visualisation provides the user with an overview of how the bases are distributed and the power and quantity of the resources at each base. This allows the user to see which bases have the most powerful resources, which are low on stock, and so on. In addition, the user can determine which bases to transfer stock to/from in the shortest time based on their proximity.

5.3.3 Parallel Coordinates

The Parallel Coordinates visualisation technique, developed by Inselberg and Dimsdale (1990), is capable of displaying a large number of data dimensions. To achieve this each data dimension is represented as a vertical axis, with the values on the axis relating to the unique set of values found across all the data items. The values for each data item are then connected across axes to form a path through the data dimensions. For the cognitive walkthroughs each axis represents a report attribute such as author name, report age, keywords, etc. Initially all paths are coloured green.

To allow the user to filter out unwanted items two 'tick' marks have been placed on each axis.

5.3 Visualisation Designs

Dragging the top tick mark on an axis filters out all reports with attribute values above the tick mark value. Similarly dragging the bottom tick mark filters out all resources with attribute values below the tick mark value. Paths through filtered items are coloured red.

Reports can be selected by clicking on their label at the right hand edge of the display. Selected reports are then highlighted in the main view by changing the colour of the associated path to yellow.

In the cognitive walkthroughs the parallel coordinates technique was used to represent report attributes. However, it should be noted that the attributes presented in figure 5.3 relate to resources. The axes, labelled at the bottom, are base, resource, quantity, distance, power level, operational, and replacement. In this example all the resources for three bases are shown and a number of resource filters have been applied. The top tick mark on the resource axis has been lowered to filter out all resources above resource 22. Resource quantities can be low, medium, or high, therefore raising the bottom tick mark on the quantity axis filters out all low quantity resources. Similarly, resources have power levels ranging from 1 to 5, thus, lowering the top tick mark on the power level axis causes all resources with power level greater than 2 to be filtered out. Finally, a small number of replacement resource values have been filtered out both at the top and bottom ends of the replacement axis. All filtered resources are shown in red. In addition, the yellow paths indicate that resource 3 and resource 6 have been selected.

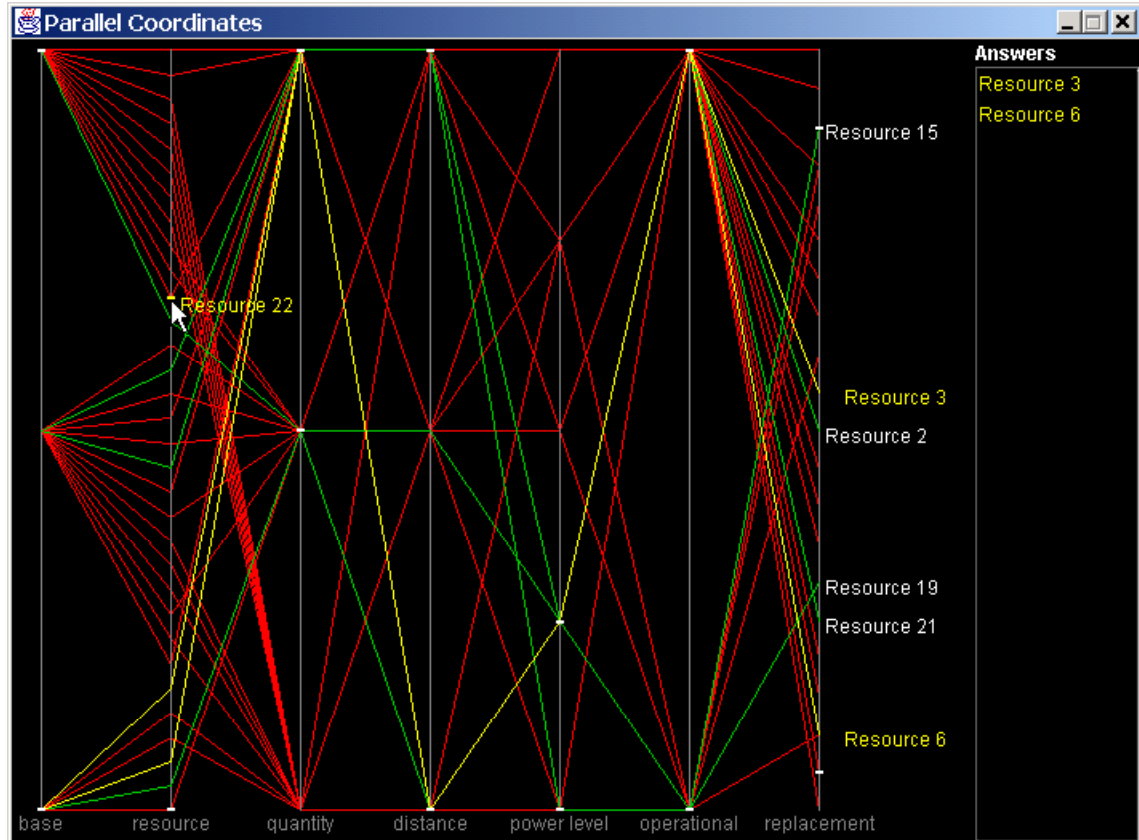


Figure 5.3: Parallel Coordinates visualisation.

Parallel coordinates are particularly useful at rapidly filtering out unwanted items. This allows the user to quickly find items that match specific criteria. To a lesser extent it also allows them to determine which values for a particular attribute are the most or least common.

5.3.4 *Route Checks*

The Route Checks visualisation is based on the more traditional map-based displays used by the military. The map is composed of several *layers*, each layer displaying different information. The list of layers includes a terrain layer, owner layer, danger level layer, path layer and installation layer. The terrain, owner, and danger level layers are mutually exclusive and each uses its own colour coding. For example, the terrain layer uses different shades of green to indicate land height above sea level and blue for water, whereas the danger level uses two shades of green to indicate no or low danger and orange and red to represent high or very high danger levels. All layers can be toggled on and off as required by the user. In addition, to reduce visual clutter, resource names and routes can also be filtered.

Routes are represented as tapered lines, the thick end indicating in the start location and the narrow end (or point) the destination. Each route has an associated resource that can be thought of as travelling along that route. Details about each resource are presented in an ‘info. window’ that can be toggled on and off by clicking on a route icon. Resources can also be selected by clicking on the associated route. This causes the route colour to turn white, which then allows the user to see which routes/resources are selected.

The Route Checks visualisation displays both friendly and enemy installations. Different installation types are represented using different shapes. Squares depict bases, triangles depict surface to air missile (SAM) sites, and circles depict power stations. The installation colour denotes whether it belongs to friendly forces (green) or enemy forces (red).

In figure 5.4 the danger level, path (or route) and installation layers are active, resource names are being displayed and all paths are shown. In addition, resources 8 and 9 have been selected and the details about resource 27 are displayed in an ‘info. window’.

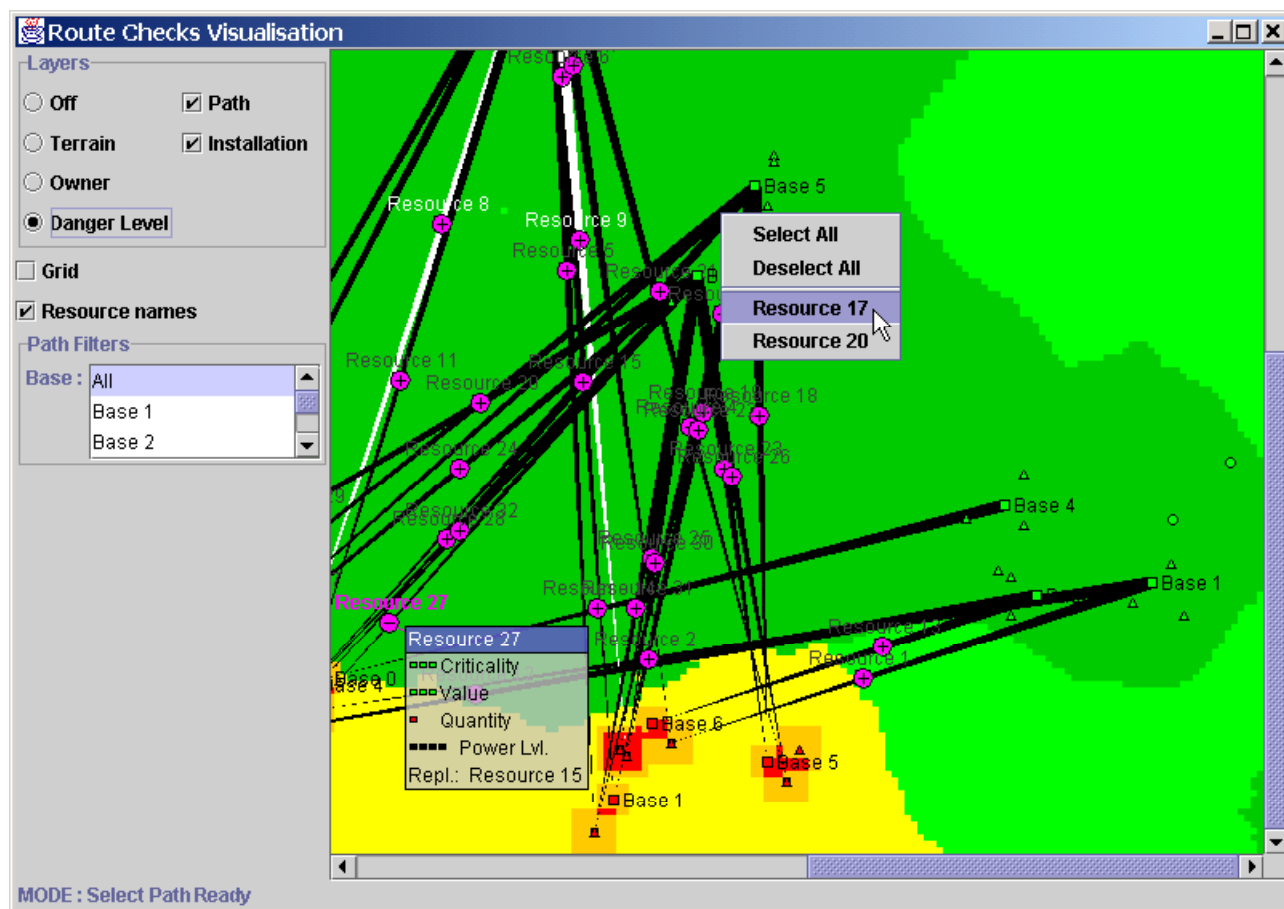


Figure 5.4: Route Checks visualisation.

The Route Checks visualisation provides the user with details about the type of terrain resources cross, the owner of the terrain, and the associated danger level. Using this information together with resources specific details such as resource criticality, value, etc., the user can judge the risks each resource is exposed to and if necessary can plan alternative routes. Additional information, such as the location and type of enemy bases, allows the user to see where the greatest threats are and plan future missions accordingly. In addition, by comparing the number outgoing/incoming resources, the user can estimate the activity levels of each base.

5.3.5 Organix

The Organix visualisation is a novel approach to visualising document collections. Common words such as *in*, *the*, *and*, etc., are ignored and all words are stemmed as required. This means that words such as *replace*, *replaced*, *replaceable*, etc., are equated to *replac* and treated as one term. The term frequency (tf) is calculated as the number of occurrences of that term within a document. However, the frequency of a term is affected by the document length so some form of normalisation is required. This is carried out in the following way. A document frequency (df) is calculated by summing the presence of a term within a document

for each document in the collection. For example, if the word 'class' occurs in 20 documents out of 100 in the collection, the document frequency is 20. The inverse document frequency (*idf*) is $1/df$. The weights for each term in each document are then calculated using the following formula:

$$weight = \frac{tf * idf}{\sqrt{(tf_1 * idf_1)^2 + (tf_2 * idf_2)^2 + \dots + (tf_n * idf_n)^2}}$$

The set of weights for each document is called the *document feature vector*.

An Organic is grown using a cell-based model. Each feature vector weight is used in turn to determine the properties of a single cell, for example its shape, colour, number of children, relative location of each child cell, etc. In addition, *pre* and *post survival rules* are used to determine if a child cell survives. The rules are based on the proximity of the other cells in the surrounding region, the size of the cell, and so on. For example, a post survival rule may state that if a cell is less than a certain radius then it dies. If the child cell survives, the process is repeated using the next weight in the vector. The result is an Organic whose form and composition is based on the weights in an individual document feature vector.

Growing an Organic for each document allows documents to be visually compared. This is useful for finding similar documents in a collection or identifying anomalous documents. It should be noted that although in this case the Organix method was applied to documents, it could as easily be applied to any dataset in which the data attribute values have been converted to feature vectors.

With respect to the military task scenarios it was decided that the Organix visualisation would be used to represent a set of situation reports. Due to time limitations, the visualisation developed was not interactive and although each Organic is three-dimensional it was decided that for the purposes of the cognitive walkthroughs each Organic should be considered as if viewed from the same location. Participants would be asked to identify the Organic most similar to the centre Organic shown in figure 5.5. Details describing which term each individual cell in each Organic represents have been omitted, as they were not relevant to the task.

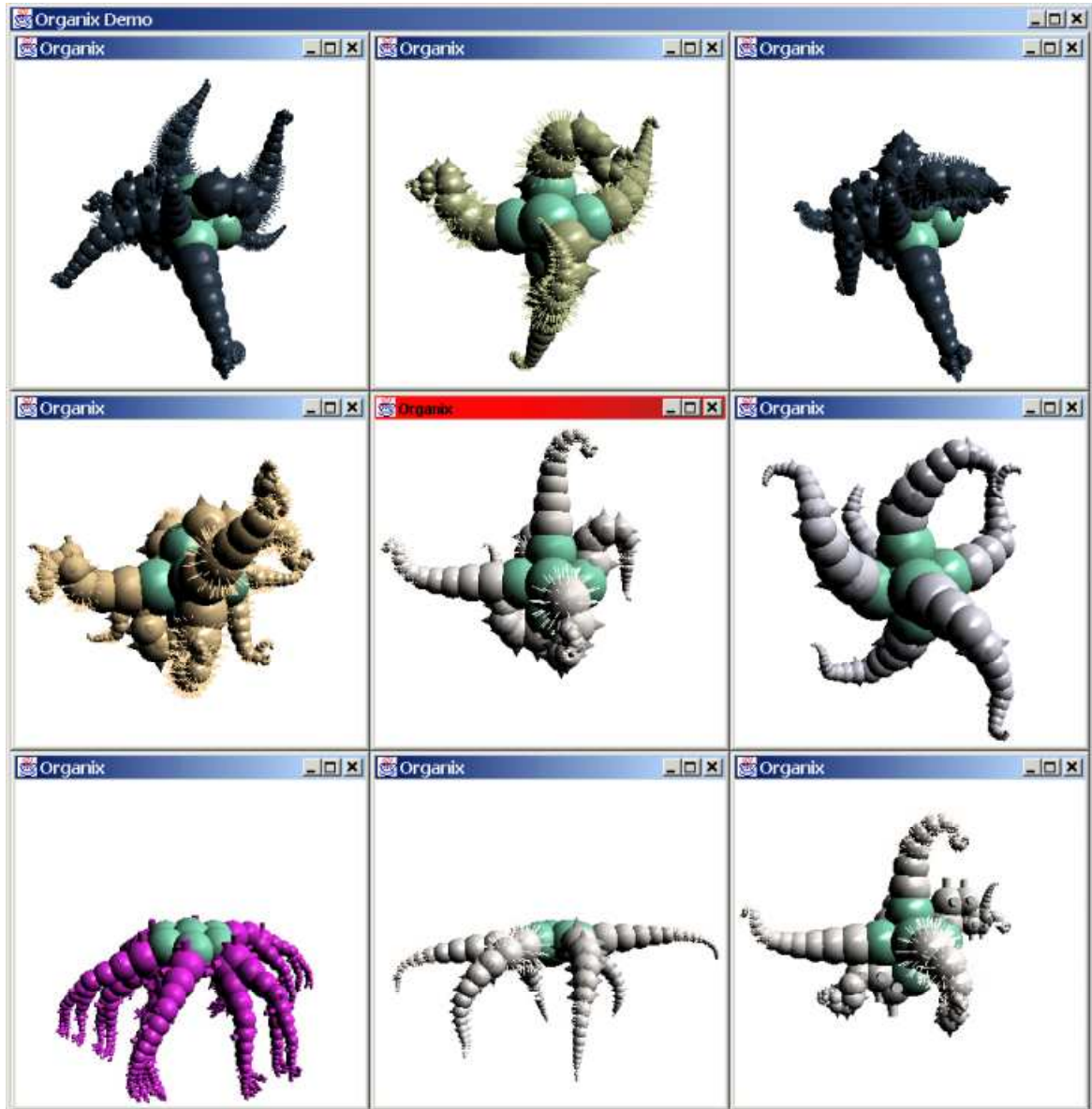


Figure 5.5: Organix.

5.3.6 Mission Execution

The Mission Execution visualisation was designed to help users monitor resource locations, directions of travel, and schedule information. Each resource is depicted as a coloured circle positioned relative to the desired position of the resource, which is at the centre of each resource display. The colour of the circle redundantly encodes how far away the resource is from the desired location. Attached to the circle are two coloured arrows indicating the desired heading and the actual heading. Again, the colour of the actual heading arrow indicates how close to the desired direction the resource is travelling. Schedule information is encoded using a 'T-bar' positioned at the centre of each resource display. Resources behind the T-bar are behind schedule and those in front are ahead of schedule. The background

5.3 Visualisation Designs

colour of each resource display is used to indicate the operational status of the resource and whether or not it is selected.

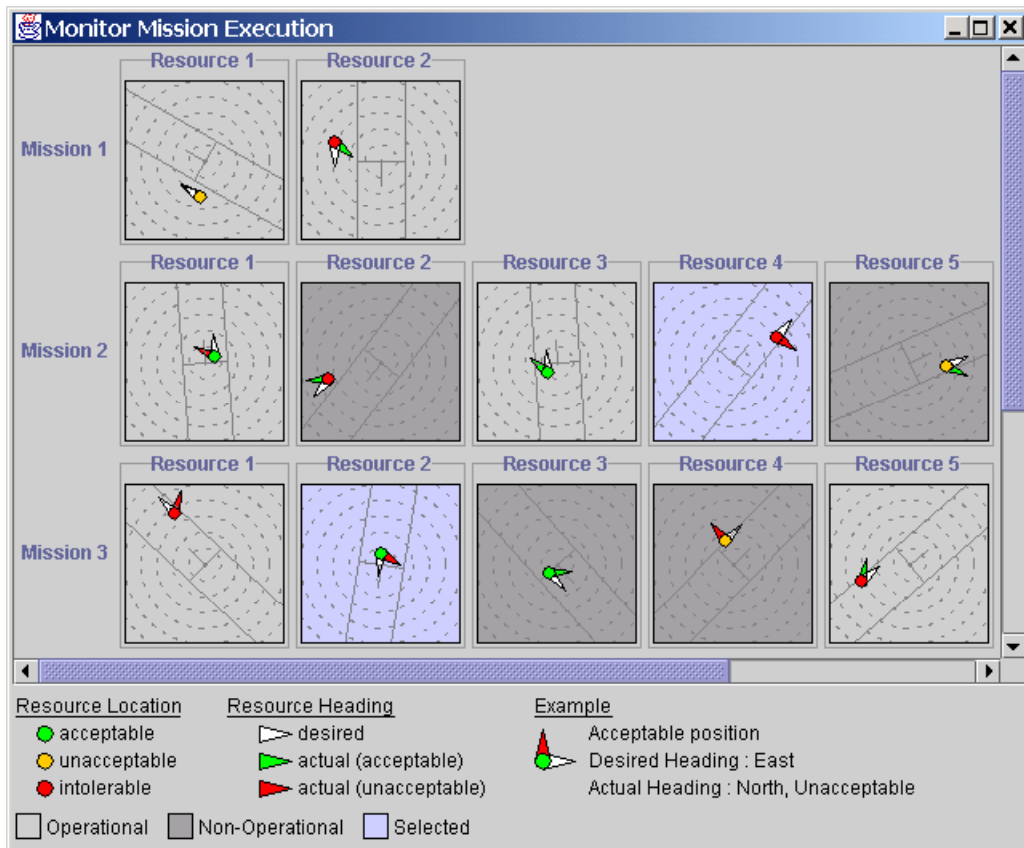


Figure 5.6: Mission Execution visualisation.

The primary aim of the Mission Execution visualisation is to allow users to determine the status of resources on each mission i.e. are they still operational, are they heading in the right direction, and are they on time? For example, in figure 5.6, mission 3, resource 2, is operational, is within an acceptable distance of its desired location, is slightly behind schedule, but is heading in the wrong direction.

5.3.7 Mission Dependencies

A mission often requires other missions to have been successfully executed before it can proceed. For example, it may be necessary to conduct a reconnaissance mission before a bombing mission attacks a target. In addition, resources are often used between planned missions, so a fighter that is planned to be used in missions on days one and three may also be required in another planned mission on day two. These complex inter-dependencies make it difficult for a user to determine a course of action when some unexpected event occurs. For example, if the fighter is destroyed on day one it cannot be used in the planned missions on days two and three, the user needs to determine if these planned missions should still go ahead based on the mission priority, risk of failure, whether an alternative resource can be

5.3 Visualisation Designs

used, and so on. The mission dependencies visualisation tries to help the user make this decision.

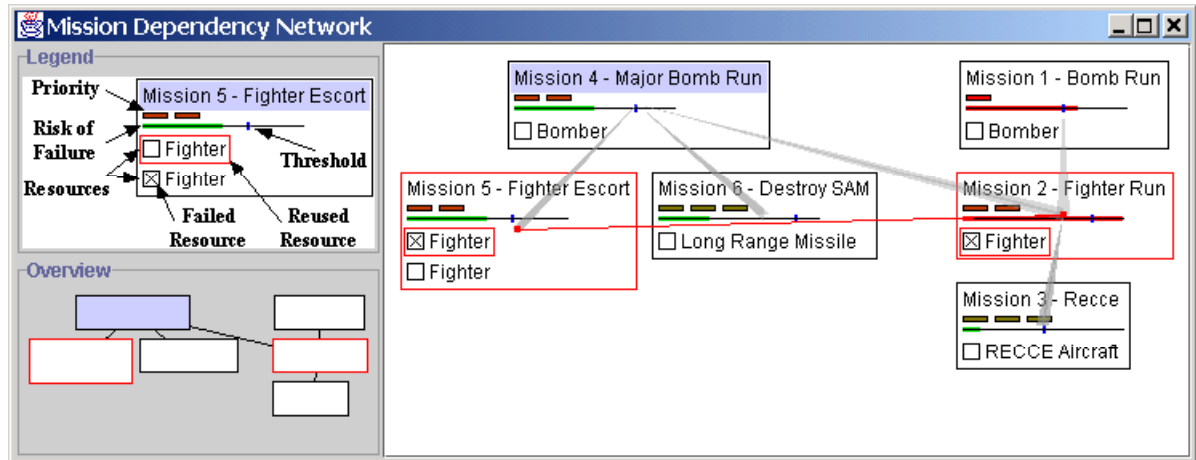


Figure 5.7: Mission Dependencies visualisation.

As can be seen in figure 5.7 each mission, together with its associated resources, is represented as a node in a network. Each node contains information about the mission and the resources that take part in that mission. Nodes are then linked together based on their dependencies to form a network.

The priority of a mission is redundantly encoded as both the number and colour of a small set of bars, one red bar indicating a high priority (one) mission and five green bars indicating a low priority (five) mission. Redundantly encoding the priority in this way helps to reinforce the importance of the mission.

Each mission has an associated risk of failure and a threshold that indicates the acceptable level of risk for that mission. This risk of failure is encoded as a single coloured bar. The length of the bar can be used to judge how close to the threshold level the mission is and the colour of the bar is used to redundantly encode the acceptable risk status of the mission. A green bar informs the user that the mission has an acceptable level of risk and a red bar suggests that the mission is too risky.

Also included in a mission node are the resources used in that mission. Failed resources are marked and increase the risk of mission failure thus the risk bars for all dependent missions grow. Resources that are reused between missions are connected together by a red line and highlighted by a red box.

In figure 5.7 the fighter used in mission 2 is also used in mission 5 (note the connecting red line). Unfortunately this fighter has failed for some reason as indicated by the cross next to the fighter label. Consequently, the risk threshold has been exceeded for mission 1 (the risk bar is red), but mission 4, which is dependent on missions 5 and 6, can still go ahead (the risk bar is

green). Although the risk bar is red in mission 1, it is only just over the threshold value. At this point the user must make a decision as to whether or not the priority one mission 1 should still go ahead despite the risks.

5.4 Application of the Design Evaluation Methods

5.4.1 Motivation

We are proposing a new methodology and in particular a set of design evaluation methods based on a collection of heuristics. It would be useful if we could gather evidence that supports these methods and at some level the heuristics they are based on. To do this the following procedure was adopted. A subset of the visualisations described in section 5.3 were chosen. Two versions of each selected visualisation were created using patterns and heuristics from the methodology appropriate to the tasks. One set of visualisations, referred to as *wow-vis*, would be designed to have high scores for the heuristics. The second set, referred to as *just-a-vis*, would be similar to the *wow-vis* set but designed to produce lower scores. For example, the Resource Checks *wow-vis* would include a legend (corresponding to heuristic 17) whereas the *just-a-vis* version would not have a legend. Thus, the *just-a-vis* has a lower score for heuristic 17. The design evaluation methods could then be applied to visualisations in each set and later each visualisation could be empirically evaluated and a comparison made between the predicted rankings and the actual rankings. If the rankings are similar it could be said that there is evidence to support the design evaluation methods in this case. It should be noted that the design evaluation methods produce a qualitative not quantitative ranking. In other words, we may be able to predict that one visualisation is more usable than another, or possibly even significantly more usable, but we will not be able to say that it is for example 10% more usable (see section 4.4.2.2). It is feasible that the evidence gathered during the empirical evaluations will allow the design evaluation methods to be tuned more accurately.

It is important to note that, due to our focus on the heuristic evaluation, the *just-a-vis* and *wow-vis* designs will be almost identical in terms of the visual features they use. The mapping and metric evaluation techniques, due to their reliance on low-level perceptual features are unlikely to be able to convincingly distinguish between the two visualisation versions. However, if we were to produce designs that were radically different in form then it would not be possible to determine if usability differences between the two versions was due to perceptual differences or differences in the heuristics each version uses. By using the same data dimension to visual feature mappings in both versions we are able to limit this problem. Unfortunately, this means that fewer conclusions will be able to be drawn about the validity of the mapping and metric evaluation techniques.

5.4 Application of the Design Evaluation Methods

The Resource Checks, Parallel Coordinates, and Route Checks visualisations, presented in section 5.3 were chosen to represent the just-a-vis set. This was primarily due to the similarities in the data and tasks that each visualisation could support. Using the methodology and feedback from the cognitive walkthroughs, a wow-vis version of each of these visualisations was designed. A description of the differences between the two versions and the predictions made by the design evaluation techniques is presented in section 5.4.4. Details of the results of the cognitive walkthroughs, using the designs presented in section 5.3, and how those results influenced the design of the wow-vis versions of the visualisations can be found in chapter six.

5.4.2 Design Evaluation Constants

To rate each design using the heuristic evaluation it is necessary to assign weights to each of the usability factors. Command and control personnel need to work quickly but with the minimum of errors, therefore both performance and rate of errors need to be heavily weighted. In contrast, personnel should be given adequate training and use the visualisations daily, which means that the time to learn and remember functionality is less important. Finally military users should feel comfortable using the software. In the case of command and control we decided that the weights shown in table 5.3 were appropriate. Retention has been given a zero weight because it relates to how well users retain knowledge of the system over time. Since the experiment was to be carried out only once retention is not relevant.

Usability Factor	Weight
Time to learn	0.1
Speed of performance	1.0
Rate of errors	1.0
Retention over time	0
Subjective satisfaction	0.5

Table 5.3: Usability factor weight assignments for command and control.

Due to our focus on the heuristic evaluation and the restrictions imposed by the empirical trials certain features had to be held constant across both the wow-vis and just-a-vis versions of each visualisation. In particular the encoding used, the number of data items and the number of dimensions displayed, are the same for each version. The consequence of this constancy is that the mapping evaluation and some of the metric scores return the same result for both the wow-vis and just-a-vis versions.

5.4.3 Heuristic Evaluation Tool

Recall that to rate each design using the heuristic evaluation technique the designer must use

5.4 Application of the Design Evaluation Methods

the following equation:

$$V_{rate} = \sum_{\text{for each u.f.}} (UF_w * \sum_{\text{for each h.}} (H_s * H_{ufe}))$$

Key:

V_{rate} : visualisation rating.

UF_w : usability factor weight.

H_s : heuristic score.

H_{ufe} : heuristic usability factor effect.

This incorporates usability factor weights, heuristic scores, and heuristic usability factor effect values. Although this is not a difficult calculation it can be tedious to perform and is subject to errors. In addition it may be useful for visualisation designers to see the effects of altering the weights, scores, and effect values. For these reasons a heuristic evaluation tool was developed capable of supporting these tasks. This tool is shown in figure 5.8.

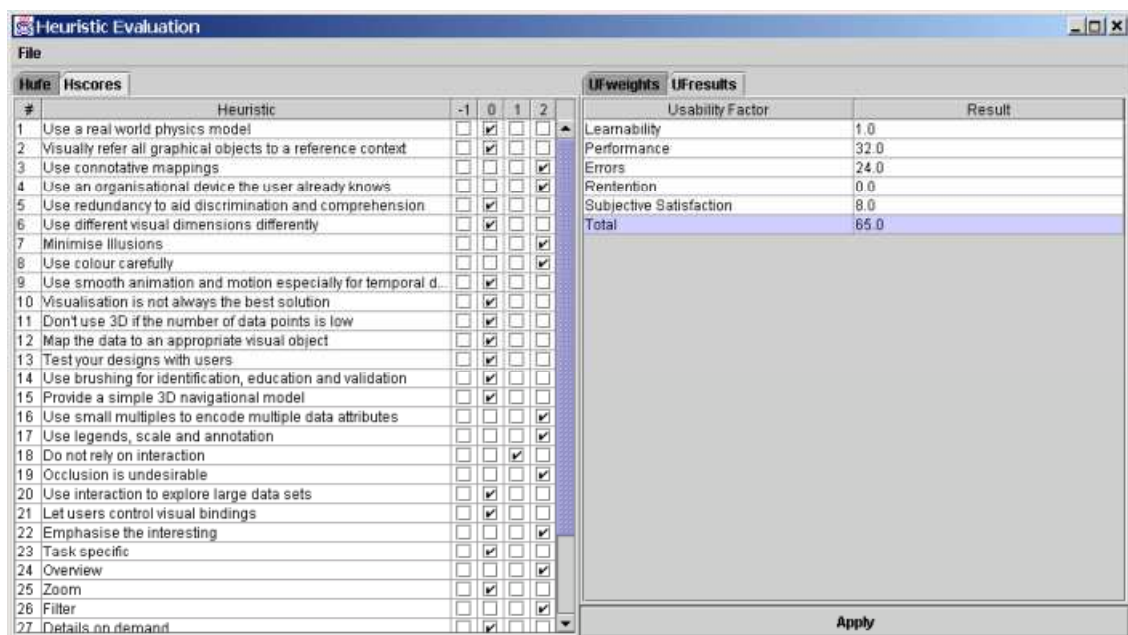


Figure 5.8: Heuristic Evaluation tool.

5.4.4 Visualisations

This section describes the differences between the just-a-vis and wow-vis versions of the chosen visualisations and the predictions made by each of the design evaluation techniques. In each case the predictions made by the mapping evaluation and heuristic evaluation will be presented as a series of tables. However, due to the issues discussed in section 4.4.2.3 a table of results cannot be produced from the metric evaluation. Instead the metrics will be discussed qualitatively.

5.4.4.1 Resource Checks

There are a number of significant differences between the wow-vis version, shown in figure 5.9, and the just-a-vis version, shown in figure 5.10.

5.4 Application of the Design Evaluation Methods

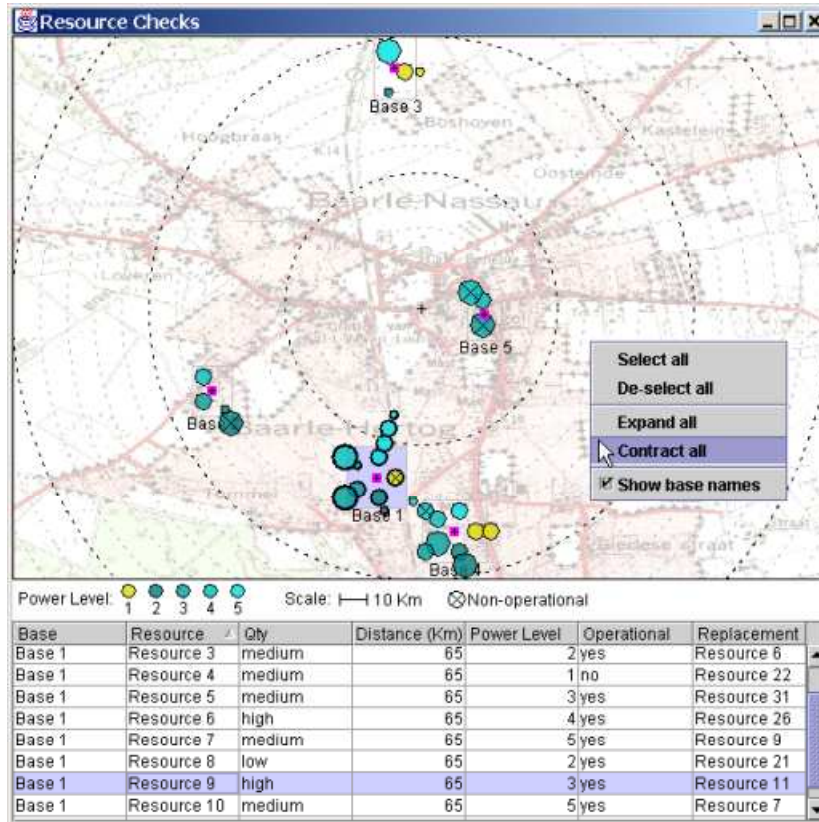


Figure 5.9: Resource Checks – wow-vis version.

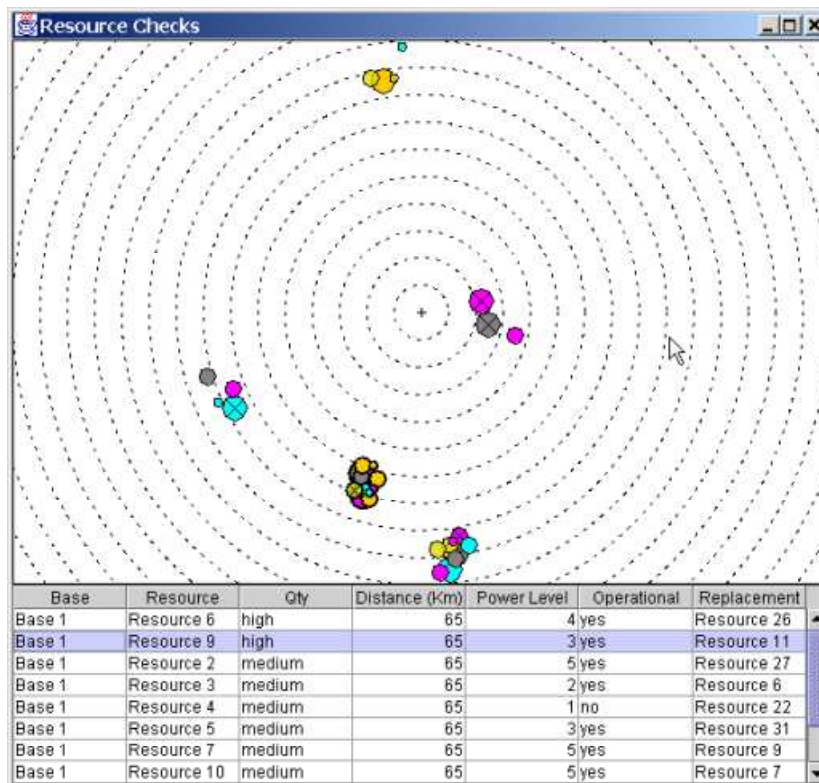


Figure 5.10: Resource Checks – just-a-vis version.

The datatips facility, described in section 2.8.4, is not available in the just-a-vis version. Removing the datatips facility prevents the user from viewing details about bases or resources in context. This relates to heuristic 14, which states that datatips can be used for identification and validation. To obtain this information in the just-a-vis the user must first select the items in question and then view the details in the table. This increases the user workload and causes an eye-shift from the location of the item to the table. Both of these factors have a negative impact on performance.

The wow-vis version includes a legend. This aids the user's memory and can help to reduce errors. In addition, legends can also help the user learn how to interpret the display. Each of these factors is referred to by heuristic 17.

One of the attributes of a resource is its power level. This is a discrete ordinal data type that can be assigned a value ranging from one to five. In both the wow-vis and the just-a-vis version power levels are represented using colour. In the wow-vis the assigned colours are based on varying levels of brightness. The exception to this is power level one resources, which have been assigned a completely different hue. The reason for this is that military convention dictates that power level one resources must be easily identifiable. Using a unique hue satisfies this need. In the just-a-vis the colours used are based on a set of discrete hues with no obvious ordering. The two colour sets used are shown in figure 5.11. According to the ranking of data type to visual feature proposed by Salisbury (2001) and both heuristics 3 and 8, the colour assignments used in the wow-vis should have a number of benefits over those used in the just-a-vis.

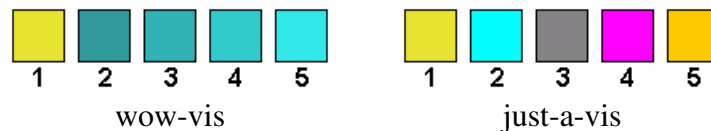


Figure 5.11: Power level colour assignments.

The layout of resources in each visualisation is significantly different. In the just-a-vis version all of the resources belonging to an individual base can be viewed at all times and are positioned randomly within a short distance from the base centre. This results in dense clusters of overlapping resources as can be seen in figure 5.12.c. In the wow-vis version the resources are sorted by quantity and displayed as columns radiating from the centre of the base. Each column displays resources of a particular power level only and are arranged in order of increasing power level clockwise around the base centre. Initially, only maximum quantity resources are shown, however via interaction the user can view all base resources either temporarily or permanently. This layout and interaction technique, which can be seen in figures 5.12.a and 5.12.b, has a number of advantages. When the wow-vis first appears the amount of information presented is less than that of the just-a-vis version. This is beneficial to novice users as it helps prevent them from becoming overwhelmed by the display. This

5.4 Application of the Design Evaluation Methods

reduction in the amount of information displayed was achieved by showing the data most relevant to the tasks, in this case the maximum quantity resources. Finally, the layout approach used in the wow-vis obviously reduces occlusion but still allows comparisons between bases to be carried out. The approaches used in the wow-vis relate directly to a number of the heuristics.

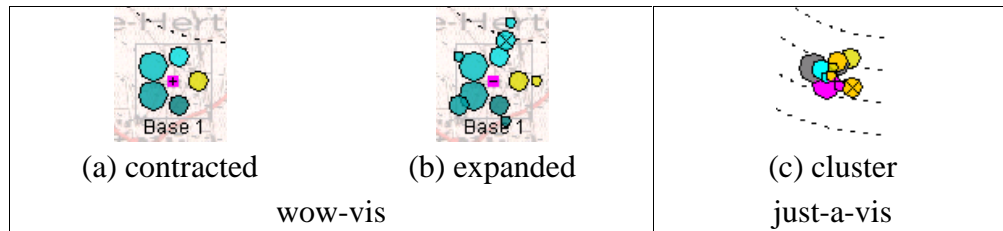


Figure 5.12: Base resource layout comparison.

The visual features used to encode each data dimension and their ranks according to Salisbury (2001) are shown in table 5.4. Only the map part of the visualisation has been rated since the tables are identical for each version. Recall that for the mapping evaluation, the lower the score, the more perceptually effective the visualisation should be. Therefore it can be seen using the mapping evaluation that the wow-vis ranks slightly better than the just-a-vis.

Data Dimension	Type	just-a-vis		wow-vis	
		Encoding(s)	Rank(s)	Encoding(s)	Rank(s)
Base Name	N	Text	6	Text	6
Base Distance	Q	Position	0	Position	0
Base Location	Q	Position	0	Position	0
Resource Name	N	Text	6	Text	6
Resource Quantity	O	Area	10	Area	10
Resource Power Level	O	Cl. Hue	2	Cl. Brightness	1
Resource Operational Status	N	Shape	5	Shape	5
Mapping Evaluation Rate		29		28	

Table 5.4: Mapping evaluation for visual features used in Resource Checks visualisation.

Table 5.5 shows the scores assigned to each heuristic for both versions of the Resource Checks visualisation. These scores, used in conjunction with the heuristic usability factor effect values in tables 5.6 and 5.7, result in the ratings shown in table 5.8. The heuristic evaluation method clearly indicates that the wow-vis version should be more usable.

#	Heuristic	just-a-vis	wow-vis
3	Use connotative mappings	0	1
8	Use colour carefully	0	1
14	Use datatips for identification, education and validation	0	1
17	Use legends, scale and annotation	0	2
19	Occlusion is undesirable	-1	1
22	Emphasise the interesting	0	1

Table 5.5: Heuristic score matrix for Resource Checks visualisation.

Note: Heuristics that are not applicable or that have the same score for both visualisation versions have not been included.

5.4 Application of the Design Evaluation Methods

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
3	Use connotative mappings	0	2	0	2	0	2	0	2	0	2	0
8	Use colour carefully	0	2	0	1	0	2	0	0	0	2	0
14	Use datatips for ident., education and validation	0	2	0	2	0	2	0	0	0	0	0
17	Use legends, scale and annotation	0	2	0	-1	0	2	0	2	0	0	0
19	Occlusion is undesirable	-1	0	0	2	-2	2	-2	0	0	2	-2
22	Emphasise the interesting	0	0	0	2	0	2	0	0	0	0	0
Total * UF_w			0		-2		-2		0		-1	

Table 5.6: Resource Checks – just-a-vis usability scores.

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
3	Use connotative mappings	1	2	2	2	2	2	2	2	2	2	2
8	Use colour carefully	1	2	2	1	1	2	2	0	0	2	2
14	Use datatips for ident., education and validation	1	2	2	2	2	2	2	0	0	0	0
17	Use legends, scale and annotation	2	2	4	-1	-2	2	4	2	4	0	0
19	Occlusion is undesirable	1	0	0	2	2	2	2	0	0	2	2
22	Emphasise the interesting	1	0	0	2	2	2	2	0	0	0	0
Total * UF_w			1.0		7		14		0		3	

Table 5.7: Resource Checks – wow-vis usability scores.

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	0	1
Speed of performance	-2	7
Rate of errors	-2	14
Retention over time	0	0
Subjective Satisfaction	-1	3
Overall rating	-5	25

Table 5.8: Resource Checks visualisation ratings.

The metric evaluation produces identical results across all metric scores. However, if we look at the metrics in qualitative terms we can make some predictions. The total number of data points in both the wow-vis and just-a-vis versions is the same but because base resources can be hidden in the wow-vis version the range of visible data points is greater. Combining this feature with the minimal overlap resource layout algorithm also results in potentially far less occlusion than the just-a-vis. Looking at these metrics we would predict that the wow-vis should be more usable.

To summarise, the mapping evaluation predicts that the wow-vis version will be perceptually slightly more effective than the just-a-vis, although this will probably not be detectable. The heuristic evaluation predicts that the wow-vis version will be more usable than the just-a-vis, especially in terms of accuracy and to a lesser extent performance and subjective satisfaction. Finally, the metric evaluation predicts that the wow-vis should be more usable due primarily to its reduced levels of occlusion. Therefore, all three design evaluation techniques indicate that the wow-vis version should be more usable than the just-a-vis. This is as expected since the wow-vis version was designed using the usability knowledge embedded in the patterns and heuristics.

5.4.4.2 *Parallel Coordinates*

The two versions are shown in figures 5.13 and 5.14.

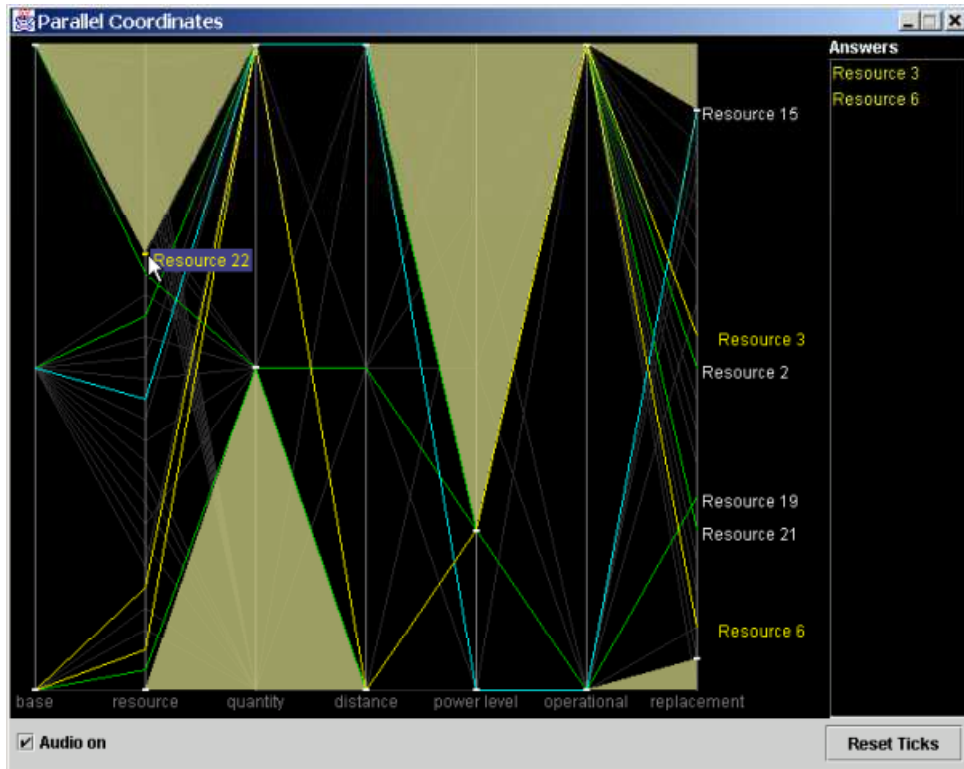


Figure 5.13: Parallel Coordinates – wow-vis version.

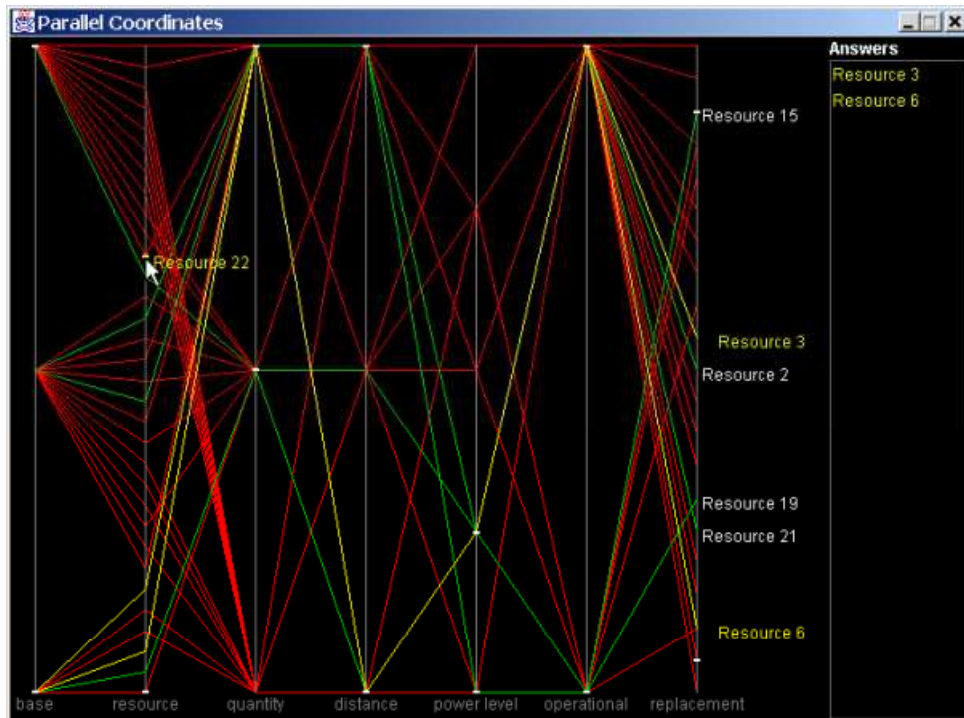


Figure 5.14: Parallel Coordinates – just-a-vis version.

5.4 Application of the Design Evaluation Methods

It can be seen that the just-a-vis version appears far more cluttered than the wow-vis version. At the same time it is more obvious on the wow-vis version which axes have been filtered and the filter values. Specifically, the wow-vis shows filtered items using a semi-transparent line colour, in contrast to the solid red used in the just-a-vis. This reduces visual clutter and occlusion but still maintains the context of non-filtered items. The wow-vis version also fills areas above and below filter values thereby making them more obvious. These differences in design relate to heuristics 5, 8 and 19.

Additional features in the wow-vis version include the ability to swap axis positions, highlight all data items that pass through a particular value (light blue paths in figure 5.13), receive audio confirmation when ticks are grabbed and a facility to reset all filter values.

In the parallel coordinates visualisation all of the data types are mapped to position. According to Mackinlay (1986) and Salisbury (2001) this is the most accurate data type to visual feature mapping and because both the wow-vis and just-a-vis are identical in this respect it is inappropriate to use the mapping evaluation technique. Similarly the metric evaluation produces identical results across all metric scores.

Table 5.9 shows the scores assigned to each heuristic for both versions of the Parallel Coordinates visualisation. These scores, used in conjunction with the heuristic usability factor effect values in table 5.10 and 5.11, result in the ratings shown in table 5.12. The heuristic evaluation method indicates that the wow-vis version should be more usable.

#	Heuristic	just-a-vis	wow-vis
5	Use redundancy to aid discrimination and comprehension	0	1
8	Use colour carefully	0	1
19	Occlusion is undesirable	-1	1
22	Emphasise the interesting	0	2

Table 5.9: Heuristic score matrix for Parallel Coordinates visualisation.

Note: Heuristics that are not applicable or that have the same score for both visualisation versions have not been included.

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
5	Use redundancy to aid discrim. and comprehension.	0	0	0	2	0	2	0	2	0	0	0
8	Use colour carefully	0	2	0	1	0	2	0	0	0	2	0
19	Occlusion is undesirable	-1	0	0	2	-2	2	-2	0	0	2	-2
22	Emphasise the interesting	0	0	0	2	0	2	0	0	0	0	0
Total * UF_w			0		-2		-2		0		-1	

Table 5.10: Parallel Coordinates – just-a-vis usability scores.

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
5	Use redundancy to aid discrim. and comprehension.	1	0	0	2	2	2	2	2	2	0	0
8	Use colour carefully	1	2	2	1	1	2	2	0	0	2	2
19	Occlusion is undesirable	1	0	0	2	2	2	2	0	0	2	2
22	Emphasise the interesting	2	0	0	2	4	2	4	0	0	0	0
Total * UF_w			0.2		9		10		0		2	

Table 5.11: Parallel Coordinates – wow-vis usability scores.

5.4 Application of the Design Evaluation Methods

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	0	0.2
Speed of performance	-2	9
Rate of errors	-2	10
Retention over time	0	0
Subjective Satisfaction	-1	2
Overall rating	<u>-5</u>	<u>21.2</u>

Table 5.12: Parallel Coordinates visualisation ratings.

Looking at table 5.12 the heuristic evaluation indicates that in terms of usability the accuracy, and to a lesser extent performance, are the most significant. The time to learn and subjective satisfaction rating for each version are almost identical and this is expected since both versions have the same structure and function in the same way.

5.4.4.3 Route Checks

The wow-vis version and the just-a-vis version are shown in figures 5.15 and 5.16 respectively. The figures show each visualisation displaying the same data. However, the terrain layer is active in the wow-vis, whereas the danger level layer is active in the just-a-vis. This accounts for the different background colours displayed in the two visualisations.

In both visualisations polygons are used to represent the routes taken by different resources. Unfortunately long thin polygons have a tendency to break up making it difficult for users to accurately identify the end of the polygon. This is a significant problem in command and control visualisations where the user must be certain of the resource destination. The solution used in the wow-vis is to draw a different coloured line around the polygon. This issue relates to heuristic 7, which states that illusions caused by effects such as polygon break up should be minimised.

There are a number of differences in the use of colour between the two versions. In the wow-vis the colours used to represent different values for resource criticality, quantity, value and power level are connotative whereas in the just-a-vis some of the colours are counterintuitive (and in the case of power level colour is not used at all). The colour mappings used in each visualisation are shown in table 5.13. The colour assignments used relate to heuristics 3, 7 and 8.

5.4 Application of the Design Evaluation Methods

	just-a-vis					wow-vis				
	low		medium		high	low		medium		high
Criticality										
Value										
Quantity										
	1	2	3	4	5	1	2	3	4	5
Power Level										

Table 5.13: Colour assignments used in Route Checks visualisations.

In the wow-vis version the resource criticality, value, and quantity are represented visually as small coloured circle icons clustered around the resource at its current location along the route it follows. The same information can only be obtained via interaction in the just-a-vis version. Showing this information immediately has the potential to improve task performance but does increase the likelihood of occlusion and increases visual clutter. However, in this case drilling down to the information causes a small ‘information window’ to be displayed that significantly increases occlusion, as can be seen in figures 5.15 and 5.16. Therefore, the performance benefit of showing the information visually as icons, which increases overall occlusion by a small amount, as well as using the drill down technique, is acceptable. In addition, the wow-vis allows these icons to be hidden, making access to this data equivalent to the just-a-vis. These issues relate to heuristics 16, 18, 19 and 22.

5.4 Application of the Design Evaluation Methods

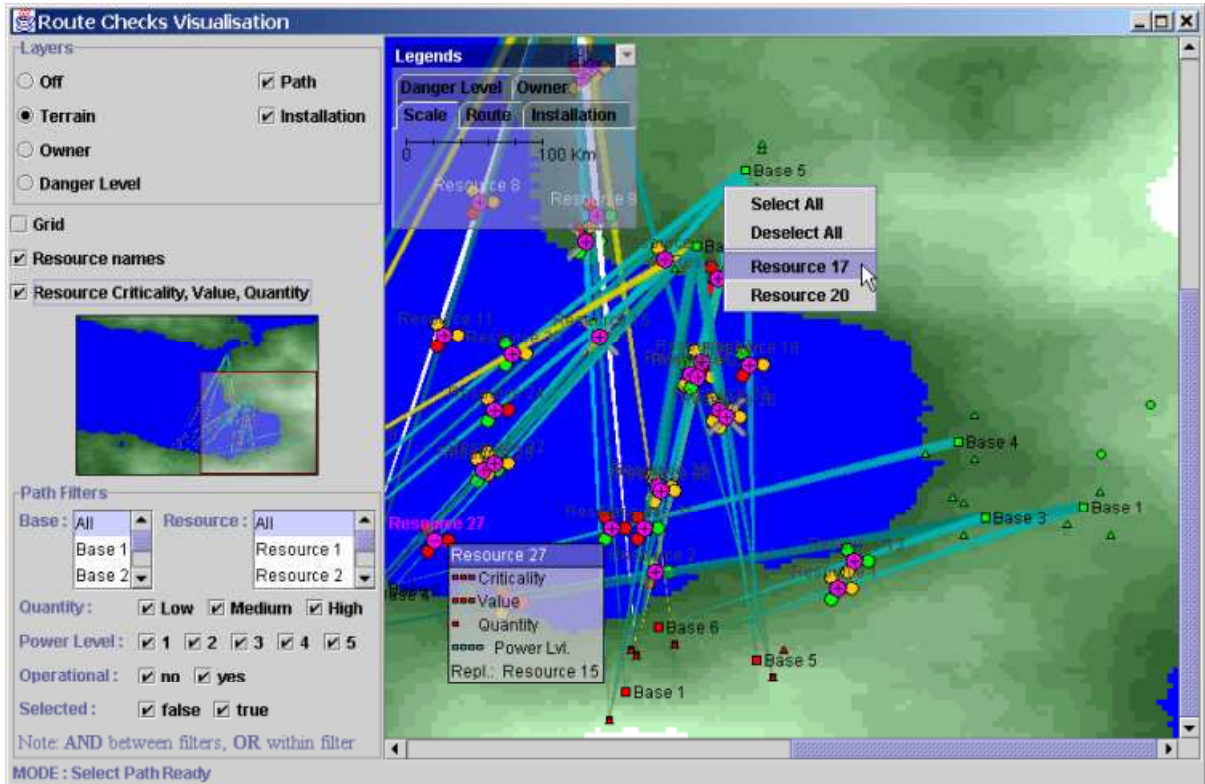


Figure 5.15: Route Checks – wow-vis version.

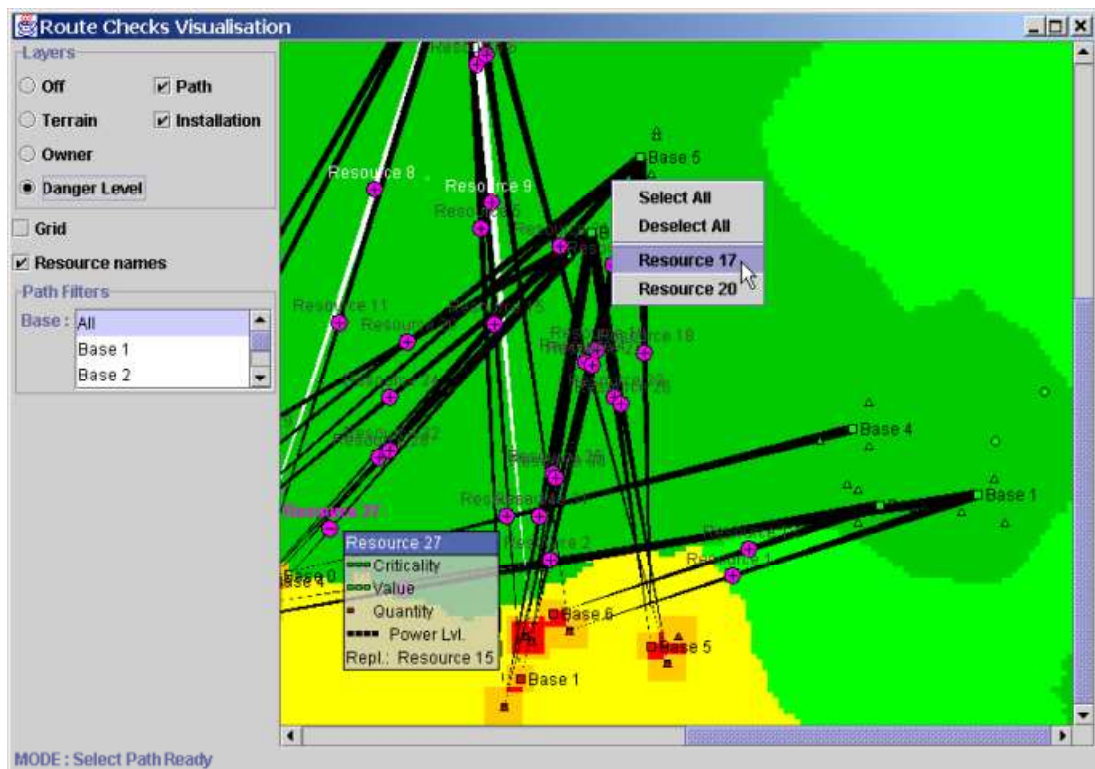


Figure 5.16: Route Checks – just-a-vis version.

5.4 Application of the Design Evaluation Methods

It should be noted that in order to limit occlusion the information windows are semi-transparent and can be moved in the wow-vis version. In the just-a-vis the information windows have a fixed position, which can result in high levels of occlusion.

The wow-vis version includes a legend. As previously stated, this aids the users' memory and helps to reduce errors. Due to the complexity of the visualisation and the encoding used on each of the layers a legend is extremely important. The absence of a legend in the just-a-vis version should have a significant effect. Examples of the legends used in the wow-vis are shown in figures 5.17.a, 5.17.b and 5.17.c. It should be noted that the legend remains on screen at all times and that to reduce occlusion the legend is semi-transparent and can be minimised by the user as required.

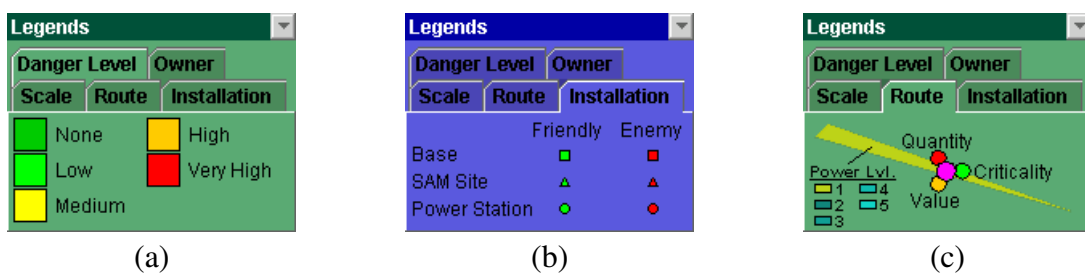


Figure 5.17: Legends used in the wow-vis Route Checks visualisation.

It can be seen in figure 5.15 that the wow-vis version uses the overview and detail technique to help the user navigate the map and at the same time maintain their context. The just-a-vis version does not use an overview, which may cause user performance to degrade. However the size of the map is such that it is unlikely the overview will have a significant affect.

There are a number of filters available in the wow-vis version that are not available in the just-a-vis. Carr (1999), Eick (1995), Foley and Van Dam (1995), and Shneiderman (1996) all state the need for filters in visualisations. The absence of filters in the just-a-vis version should affect user performance, making tasks much more difficult to complete.

The visual features used in the main display to encode each data dimension and their ranks according to Salisbury (2001) are shown in table 5.14. In this case the wow-vis version encodes several more data dimensions on the main display than the just-a-vis version. Consequently a comparison of the two versions based on the common data dimensions gives no clear indication of which visualisation is more effective. However, it is hoped that the heuristic evaluation and metric evaluation techniques can be used to resolve this issue.

5.4 Application of the Design Evaluation Methods

Data Dimension	Type	just-a-vis		wow-vis	
		Encoding(s)	Rank(s)	Encoding(s)	Rank(s)
Installation Type	N	Shape	5	Shape	5
Installation Location	Q	Position	0	Position	0
Installation Owner	N	Cl. Hue	1	Cl. Hue	1
Base Name	N	Text	6	Text	6
Region Danger Level	O	Cl. Hue	2	Cl. Hue	2
Region Owner	N	Cl. Hue	1	Cl. Hue	1
Region Terrain Type	N	Cl. Hue	1	Cl. Hue	1
Route Start Location	Q	Position	0	Position	0
Route End Location	Q	Position	0	Position	0
Route Direction	Q	Slope	3	Slope	3
Route Length	Q	Length	1	Length	1
Resource Name	N	Text	6	Text	6
Resource Criticality	O			Cl. Hue	2
Resource Value	O			Cl. Hue	2
Resource Quantity	O			Cl. Hue	2
Resource Power Level	O			Cl. Brightness	1
Resource Operational Status	N			Shape	5
Mapping Evaluation Rate		26		26 [38]	

Table 5.14: Mapping evaluation for visual features used in Resource Checks visualisation.

Note: The data dimensions missing are not encoded in the main display of the just-a-vis version.

Table 5.15 shows the scores assigned to each heuristic for both versions of the Route Checks visualisation. These scores, used in conjunction with the heuristic usability factor effect values in table 5.16 and 5.17, result in the ratings shown in table 5.18. The heuristic evaluation method clearly indicates that the wow-vis version should be more usable.

#	Heuristic	just-a-vis	wow-vis
3	Use connotative mappings	-1	2
7	Minimise illusions	0	2
8	Use colour carefully	-1	2
16	Use small multiples to encode multiple data attributes	0	2
17	Use legends, scale and annotation	0	2
18	Do not rely on interaction	0	1
19	Occlusion is undesirable	-1	2
22	Emphasise the interesting	0	2
24	Overview	0	2
26	Filter	0	2

Table 5.15: Heuristic score matrix for Route Checks visualisation.

Note: Heuristics that are not applicable or that have the same score for both visualisation versions have not been included.

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
3	Use connotative mappings	-1	2	-2	2	-2	2	-2	2	-2	2	-2
7	Minimise illusions	0	0	0	2	0	2	0	0	0	0	0
8	Use colour carefully	-1	2	-2	1	-1	2	-2	0	0	2	-2
16	Use small multiples to encode multiple data attrs.	0	1	0	1	0	1	0	0	0	0	0
17	Use legends, scale and annotation	0	2	0	-1	0	2	0	2	0	0	0
18	Do not rely on interaction	0	0	0	0	0	0	0	0	0	0	0
19	Occlusion is undesirable	-1	0	0	2	-2	2	-2	0	0	2	-2
22	Emphasise the interesting	0	0	0	2	0	2	0	0	0	0	0
24	Overview	0	-1	0	2	0	2	0	0	0	0	0
26	Filter	0	-1	0	2	0	1	0	0	0	0	0
Total * UF_w			-0.4		-5		-6		0		-3	

Table 5.16: Route Checks – just-a-vis usability scores.

5.4 Application of the Design Evaluation Methods

#	Heuristic	Score	Learn (0.1)		Perf. (1)		Errors (1)		Retnt. (0)		Sub. Sat. (0.5)	
			H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S	H _{UFE}	H _{UFE} *H _S
3	Use connotative mappings	2	2	4	2	4	2	4	2	4	2	4
7	Minimise illusions	2	0	0	2	4	2	4	0	0	0	0
8	Use colour carefully	2	2	4	1	2	2	4	0	0	2	4
16	Use small multiples to encode multiple data attr.	2	1	2	1	2	1	2	0	0	0	0
17	Use legends, scale and annotation	2	2	4	-1	-2	2	4	2	4	0	0
18	Do not rely on interaction	1	0	0	0	0	0	0	0	0	0	0
19	Occlusion is undesirable	2	0	0	2	4	2	4	0	0	2	4
22	Emphasise the interesting	2	0	0	2	4	2	4	0	0	0	0
24	Overview	2	-1	-2	2	4	2	4	0	0	0	0
26	Filter	2	-1	-2	2	4	1	2	0	0	0	0
Total * UF_w			1.0		26		32		0		6	

Table 5.17: Route Checks – wow-vis usability scores.

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	-0.4	1
Speed of performance	-5	26
Rate of errors	-6	32
Retention over time	0	0
Subjective Satisfaction	-3	6
Overall rating	-14.4	65

Table 5.18: Route Checks visualisation ratings.

In terms of the metric evaluation there are a number of differences. Perhaps the most significant difference between the two versions is the number of simultaneous dimensions displayed. The small circle icons used to represent a resources quantity, criticality, and value, in the wow-vis, together with the use of colour on the routes to encode resource power level, and the cross indicating non-operational resources, means that it displays five more dimensions simultaneously than the just-a-vis version. Using this metric alone does not mean that the wow-vis is more usable or more effective; it simply means that potentially the user has access to more information at any given time.

Looking at the occlusion metric the facility in the wow-vis to move the ‘info. windows’ means that for equivalent displays the wow-vis can always achieve a lower occlusion percentage than the just-a-vis. This also has the effect of increasing the percentage of identifiable points.

In this case it is difficult to interpret the metrics as preferring one visualisation over another. However using the occlusion metrics in isolation and knowing that occlusion can seriously inhibit usability, we would predict that the wow-vis should be more usable.

To summarise, although the mapping evaluation could not be used to rank the two versions, the heuristic evaluation and the metric evaluation do predict that the wow-vis will be more usable. The prediction made by the heuristic evaluation is particularly strong indicating that the wow-vis version will be much more usable especially in terms of performance, errors, and

5.4 Application of the Design Evaluation Methods

subjective satisfaction. Since the wow-vis version has been designed using the usability knowledge embedded in the patterns and heuristics, this is to be expected.

5.4.5 Summary of Predictions

The mapping evaluation is based on the ranking of data type to visual feature proposed by Salisbury (2001). Looking at the summary of the mapping evaluation ratings shown in table 5.19 there is no strong indication as to which version of each visualisation is the most effective. This is primarily due to the fact that each version, apart from Resource Checks, uses identical data type to visual feature mappings. Consequently the mapping evaluation produces the same ratings for each version.

Visualisation	Rating	
	just-a-vis	wow-vis
Resource Checks	29	28
Parallel Coordinates	-	-
Route Checks	26	26

Table 5.19: Summary of ratings using the mapping evaluation.

Note: Lower scores indicate the visualisation should be perceptually more effective.

The heuristic evaluation is based on a combination of the classification of the heuristics in terms of their effects on various usability factors, the scores assigned by the designer to each heuristic, and the designer's usability design goals. A summary of the heuristic evaluation predictions is shown in table 5.20. The heuristic evaluation gives a much clearer indication that in each case the wow-vis version will be more usable.

Looking at the more detailed ratings in tables 5.8, 5.12, and 5.18, the heuristic evaluation suggests that the most significant usability differences will be related to speed of performance, rate of errors, and subjective satisfaction.

Visualisation	Rating	
	just-a-vis	wow-vis
Resource Checks	-5	25
Parallel Coordinates	-5	21.2
Route Checks	-14.4	65

Table 5.20: Summary of visualisation ratings using the heuristic evaluation.

Note: Higher scores indicate the visualisation should be more usable.

A summary of the rankings predicted by each of the design evaluation methods is shown in table 5.21. Of the three techniques, the mapping evaluation is the most inconclusive in this case. As stated above, this is primarily due to the similarity in data type to visual feature mappings used in each version. This decision was a result of the need to restrict, as far as possible, differences in the designs to heuristic differences. By contrast the heuristic evaluation gives the strongest indication as to which version of each visualisation is the most

5.4 Application of the Design Evaluation Methods

usable. The metric evaluation, although difficult to apply, does seem to corroborate the predictions made by the heuristic evaluation. None of the evaluation techniques are contradictory. This would suggest that when used together they are a useful tool for determining the relative usability of different visualisation designs.

Visualisation	Method	just-a-vis	wow-vis
Resource Checks	Mapping Evaluation	✘	✓
	Heuristic Evaluation	✘	✓
	Metric Evaluation	✘	✓
Parallel Coordinates	Mapping Evaluation	-	-
	Heuristic Evaluation	✘	✓
	Metric Evaluation	-	-
Route Checks	Mapping Evaluation	-	-
	Heuristic Evaluation	✘	✓
	Metric Evaluation	✘	✓
Key ✓ Ranked as the most usable of the two versions ✘ Ranked as the least usable of the two versions - Usability ranking inconclusive			

Table 5.21: Summary of visualisation rankings using each design evaluation technique.

To conclude, the majority of the evaluation methods described in chapter four and applied here predict that the wow-vis versions of the visualisations should be more usable than their just-a-vis equivalents. To try to confirm these predictions a series of cognitive walkthroughs and empirical evaluations were conducted. The design of these experiments and the results of their execution are described in chapter six.

Chapter 6: EVALUATION, RESULTS, AND ANALYSIS

This chapter discusses the objectives of an evaluation of the methodology and the process used to achieve those objectives. Also included are the results of a series of cognitive walkthroughs conducted using prototype visualisations. This is followed by a detailed description of the main experiment design and an analysis of the results obtained from an empirical evaluation of the visualisations produced in chapter five.

6.1 Objectives and Approach

As discussed in section 5.1, we have decided to concentrate on evaluating key aspects of the proposed methodology. Consequently, the primary objective of this evaluation is to gather evidence that supports the predictions made by the design evaluation methods described in section 4.4.2 and applied in chapter five. In other words, we are looking for empirical evidence that the visualisation rankings shown in table 5.21 are correct. If the data collected supports these rankings it would imply that the design evaluation methods are at least correct in this case. We are particularly interested in validating the heuristic design evaluation technique because, not only is it a new approach to visualisation evaluation, but also because of the heuristic's ability to capture 'fuzzy' design rules, some of which incorporate the low-level perceptual rules used by the other evaluation techniques. In general terms it could be argued that we are seeking evidence that visualisations that make use of the heuristics, patterns, and methodology described in chapter four, have high usability.

To a lesser extent, the evaluation will also help to substantiate the value of the heuristics used and the classification of those heuristics in terms of usability. Obviously it is not possible to apply all the heuristics to a single visualisation or even to a small set of visualisations. However, it is possible to include the more commonly referred to heuristics such as overview and detail, filters, etc. Also, since the heuristics are being used in combination, it is unlikely that we will be able to tell whether any individual heuristic causes a particular effect. However, it should be possible to say whether or not a group of heuristics applied in a certain context have a beneficial effect.

The emphasis on finding evidence of the validity of the heuristics and the heuristic evaluation technique does lead to a number of restrictions on the visualisation designs. If the visualisations are made too dissimilar at a perceptual level and at the same time use very different heuristics then it would not be possible to judge if any effect is caused by the perceptual differences or the differences in heuristics. To overcome this problem the designs are identical in terms of the data attribute to visual feature mappings used. Therefore any effects are more likely to be due to the differences in heuristics.

6.2 Cognitive Walkthroughs

QinetiQ's overall aim is to collect evidence of the appropriateness of using visualisations as tools to aid decision-making in command and control. From the evidence collected they hope to develop recommendations for the production of visualisations specific to command and control.

To achieve these objectives the evaluation process consisted of two distinct stages. The first stage was to conduct a series of cognitive walkthroughs using the task scenarios and prototype visualisations described in section 5.3. Feedback from the walkthroughs was then used to improve the visualisation designs resulting the wow-vis set described in section 5.4. In the second stage, a more rigorous experiment was conducted using a subset of the visualisations used in the cognitive walkthroughs. Each of these stages will now be described in more detail.

6.2 Cognitive Walkthroughs

A cognitive walkthrough is a technique used to evaluate user interface designs. Due primarily to time limitations we found that the task scenarios described in section 5.2.2 were too broad and too complex to be used in the cognitive walkthroughs. To overcome this, each task scenario was refined. This resulted in the six tasks shown in table 6.1, one primary task for each visualisation.

Resource Checks

Given:

- The current time is 09:00.
- Mission *m* is scheduled to start at 10:30.
- Mission *m* requires 100 resources however there are only 20 available.
- It takes an average of 30 minutes to travel 20Km.
- There is a zero waiting time for resources to be loaded onto transport.
- Resources can be used immediately once they arrive.
- All resources displayed in the visualisation are capable of performing the same function as the missing resources.
- Some resources are more powerful than others.

Select: The resources that can be used as replacements and have the same level of power.

Route Checks

Given:

- Each resource travels a particular route.
- Each route may cross several areas that have a danger level classed as one of *none*, *low*, *medium*, *high*, *very high*.
- Each resource has an associated criticality classed as one of *low*, *medium*, *high*.
- Each resource has an associated asset value classed as one of *low*, *medium*, *high*.
- Each resource has an associated quantity classed as one of *low*, *medium*, *high*.
- A route is considered to be too dangerous to travel if it meets all the following criteria:
 1. There are any areas along the route that are categorised as having a *very high* danger level.
 2. The resources meeting criteria 1) have a *high* criticality.
 3. The resources meeting criteria 1) have a *high* value rating.
 4. The resources meeting criteria 1) have a *low* quantity.

Select: The routes that are acceptable given the above criteria.

6.2 Cognitive Walkthroughs

<p>Parallel Coordinates</p> <p>Given:</p> <ul style="list-style-type: none"> • Report <i>rpt</i> states there is an enemy SAM site at location <i>lat/lon</i>. • A report is classed as one of the following, <i>very reliable</i>, <i>reliable</i>, <i>unreliable</i>, <i>very unreliable</i> or <i>unknown</i>. • A report may be considered reliable if: <ol style="list-style-type: none"> 1. Other reports can be found that have the same or higher weights for the same keywords. 2. The reports that meet criteria 1) have a reliability rating of <i>reliable</i> or <i>very reliable</i>. 3. The reports that meet criteria 1) and 2) are not contradictory. • A report may be considered up to date if it is less than or equal to 1 day old. <p>Select:</p> <p>The <i>recent</i> reports that provide supporting evidence for the reliability of <i>rpt</i>.</p> <p>Determine:</p> <p>If the majority of the past reports by the same author have been classed as reliable or very reliable.</p>
<p>Organix</p> <p>Given:</p> <p>All reports can be visually represented to show similarities and differences.</p> <p>Select:</p> <p>The report that is most similar to <i>rpt</i>.</p>
<p>Mission Execution</p> <p>Given:</p> <ul style="list-style-type: none"> • Each resource follows a particular route. • The route may consist of several fixed locations. • Each resource should be near a particular fixed location at a certain time. • A resource may no longer be operational i.e. it has been destroyed <p>Determine:</p> <p>For each resource used in each mission, select those operational resources that are within an acceptable distance of the correct location, heading in the desired direction and are behind schedule.</p>
<p>Mission Dependencies</p> <p>Given:</p> <ul style="list-style-type: none"> • Each mission has a priority from 1 to 5, 1 being the highest, 5 being the lowest. • Each mission has a risk of failure. • Each mission also has an acceptable level of risk of failure. • Each mission has a set of associated resources. • A mission should go ahead if: <ol style="list-style-type: none"> 1. Its risk of failure is less than the acceptable level of risk of failure. OR 2. Its priority is 2 or higher. 3. AND At least one resource in the set of resources has not failed. • Resource <i>r</i> in mission <i>m</i> fails. <p>Question:</p> <p>Which missions can still go ahead?</p>

Table 6.1: Tasks used in cognitive walkthroughs.

In this case the cognitive walkthrough technique was used to assess the appropriateness of the visualisations to command and control, the usability of the visualisations, and to evaluate the comprehensibility and suitability of the tasks.

6.2.1 Procedure

Eight military personnel took part in the cognitive walkthroughs each of whom had some command and control experience and could be considered subject matter experts. Each participant was required to solve each task in turn using the appropriate visualisation. The participants were deliberately given only a few minutes training so that the intuitiveness of each visualisation could be determined. Each participant was asked to give a *verbal protocol* or *narrative*, describing their thoughts, actions, expectations and so on, as they tried to solve a particular task. All the walkthroughs were audio taped and the researcher and a psychologist from QinetiQ observed and made notes. After the walkthrough the participants completed a modified Software Usability Measurement Inventory (SUMI) questionnaire. This consisted of a usability rating, intuitiveness rating, preference ranking and an open-ended section for any additional comments. At the end of each session each participant was asked specific questions by the technical staff based on their observations.

6.2.2 Technical Analysis

For each of the visualisation designs described in section 5.3 (Resource Checks, Parallel Coordinates, Route Checks, Organix, Mission Execution, and Mission Dependencies) the comments recorded by each participant and the observations noted during the cognitive walkthrough were analysed. This analysis highlighted a recurring set of problems across all the visualisations as well as specific problems with individual visualisations. Some of the more interesting and commonly occurring problems are described in this section.

The most commonly recurring problem encountered across all the visualisations was the participants' inability to discover functionality. For example, in the Resource Checks visualisation it is possible to sort the contents of a column by clicking on the column heading. However, only a few of the participants discovered this feature and this was only by fully exploring the interface. With more training this problem could have been overcome but it does highlight the need to avoid 'hidden' functionality. In this case the solution was to alter the mouse pointer shape as it hovered over a column heading. This technique of using a visual cue to indicate that an action can be taken is commonly used in Windows™ applications and should be familiar to the user. For example, when the pointer moves over a window frame its shape changes indicating that the window can be resized.

In many cases the problems related to the more common interface controls, or lack of them, rather than to the view of the data presented in the visualisation. The exception to this was the Route Checks visualisation where two major problems became evident.

The first problem was due to the mappings used to represent the ‘quantity’, ‘criticality’ and ‘value’ of each resource. Each of these attributes is an ordinal data type that can be assigned a value of low, medium or high. These values were encoded both as a number of bars (1, 2 or 3) and as a colour hue (red, orange, green). Thus ‘low’ is mapped to a single red bar and ‘high’ is mapped to 3 green bars. This is the correct encoding for the resource ‘quantity’ attribute. However, it became evident that although the number of bars is correct for all three attributes, the choice of colour encoding was incorrect for both the ‘criticality’ and ‘value’ attributes. Applying the mapping results in highly critical resources being represented as 3 green bars. In western cultures green is normally associated with ‘safe’ or ‘good’ therefore many participants misinterpreted this information, believing that resources were not critical to a mission when in fact they were very critical. Interestingly, some participants noted the incorrect colour encoding and due to the information being redundantly encoded as the number of bars were able to complete the task successfully. This problem has particular significance to two of the heuristics, use redundancy (heuristic 5), and use colour carefully (heuristic 8). This information was used later during the design of the wow-vis version.

The second problem concerned the capability of the visualisation to support the user in performing the required task. The task to be achieved using the Route Checks visualisation required the participants to identify acceptable routes given a set of criteria. During the task several of the participants wrote their own tables on scraps of paper, with each row in the table representing a resource and each column a criterion. The participants would then tick the cells in the table and look for rows that only had ticks in them to determine acceptable resources. Creating a table implies that the visualisation has failed to aid the participant in solving the task. The participants are, in effect, resorting to their own visualisation; one that they feel is more capable of supporting them in the task.

From observation, the cause of the problem was identified as being due to excessive drill down and occlusion. The information required to determine if a route was acceptable or not could only be viewed by popping up a small information window attached to the route. These ‘info. windows’ could not be moved and, due to the proximity of each route, often overlapped. This resulted in information being obscured and unreadable without first closing other ‘info. windows’. The solution the participants used was to open each ‘info. window’ individually and generate the table as described above. In this case the table is being used as a memory aid. It should be noted that this problem was made worse by the fact that routes could not be selected in the visualisation. Had the participants been able to select routes they could have checked if a route was acceptable, selected it if necessary, and then carried on. Therefore it is less likely that they would have needed to resort to pen and paper. However, this does not mean that the problems are not significant. The solution used later in the wow-vis visualisations was to show the relevant information on screen without the need to drill down to it, and to allow the ‘info. windows’ to be moved so as to minimise occlusion.

6.2.3 Usability Analysis

A usability analysis was performed using the data collected from the modified SUMI questionnaires administered at the end of each cognitive walkthrough. This led to three distinct result categories, a profile of the perceived usability, an intuitiveness rating and a preference ranking.

6.2.3.1 Profile of Perceived Usability

The profile attempts to show the user's overall subjective usability rating across all the visualisations based on five usability dimensions. Cook (2002) defines these as:

- *Learnability*: Measures the speed and facility with which the user feels that they have been able to master the system or to learn how to use new features.
- *Efficiency*: Measures the degree to which users feel that the software assists them in their work. Compares the expenditure of resources (mental or physical) to the level of effectiveness achieved.
- *Affect*: Measures the user's general emotional reaction to the software. The degree to which they like or dislike the software.
- *Control*: Measures the extent to which the user feels in control of the software (as opposed to feeling controlled by the software).
- *Helpfulness*: Measures the degree to which the software is self-explanatory. Also considers specific functions such as the adequacy of the help facilities and documentation.

The results show that the visualisations scored well for all dimensions except helpfulness. However this low score is to be expected. The visualisations used in the cognitive walkthroughs and later in the trials are not intended to be commercial products and as such had no help facilities. This, coupled with the minimal training given to the participants, is the likely cause of the low helpfulness score. However, despite this low score, the overall usability rating of 73% is above the average 40-60% range expected of most software (Cook 2002). The complete results are shown in table 6.2.

Dimension	Usability Score (%)
Affect	87
Learnability	87
Efficiency	81
Control	80
Helpfulness	32
Overall	73

Table 6.2: Subjective usability of visualisations.

6.2.3.2 Intuitiveness Rating

The participants were asked to rate the intuitiveness of the functions of each visualisation using a scale from 1 to 5, with 1 indicating that ‘all or almost all’ of the functions were intuitive and 5 indicating that ‘none or almost none’ of the functions were intuitive. The average intuitiveness ratings for each visualisation are shown in table 6.3 with lower scores indicating the visualisation was more intuitive to use.

Visualisation	Mean	Mode	Median	Standard Deviation
Resource Checks	2.38	3.00	2.50	1.11
Route Checks	1.75	1.00	1.00	1.21
Parallel Coordinates	2.38	2.00	2.00	1.40
Organix	1.88	1.00	1.50	1.21
Mission Execution	1.63	1.00	1.00	0.79
Mission Dependencies	2.00	3.00	2.00	1.00

Table 6.3: Average intuitiveness ratings for each visualisation.

Using the mean value as a rough guide, the results in table 6.3 would suggest that in terms of intuitiveness the visualisations could be ordered as in table 6.4.

Visualisation	Mean	Mode	Median	Perceived Intuitiveness
Mission Execution	1.63	1.00	1.00	↑ more intuitive
Route Checks	1.75	1.00	1.00	
Organix	1.88	1.00	1.50	↓ less intuitive
Mission Dependencies	2.00	3.00	2.00	
Parallel Coordinates	2.38	2.00	2.00	
Resource Checks	2.38	3.00	2.50	

Table 6.4: Visualisations ranked by perceived intuitiveness.

Although the ordering shown in table 6.4 cannot be said to be completely accurate it does seem reasonable. The Route Checks visualisation is a map-based display, which is the current non-software based military standard for displaying this type of data. Maps are also the most common format used by non-military individuals trying to plan a route. It is therefore reasonable to assume that the participants past experience will have prepared them to use this type of display. They will expect the visualisation to provide certain functions and be interpreted in a particular way. This relates well to the heuristic 4, ‘use an organisational device the user already knows’.

Interestingly, the Mission Execution visualisation was rated as more intuitive than the Route Checks map-based display. On further investigation it became apparent that the Mission Execution visualisation was similar to a *plan position indicator* (PPI) display already used in command and control. This familiarity with a similar display may be one reason why it was ranked so highly. However, it is equally possible that the high intuitiveness rating was due to the mappings used in the visualisation and the nature of the task. Participants discovered that

they could quickly narrow the search space down to a small set of resources by simply looking for resources that used green only. Resources that used any red or orange failed to meet the criteria for the task. This trivial visual filtering process may have been why the participants found it so intuitive.

The Resource Checks visualisation was found to be the least intuitive. This is surprising because the display is relatively simple and, once bases have been selected, the participant is only expected to interact with a table, which is a familiar organisational device. One reason why this visualisation may have been perceived as unintuitive is that it was always presented first. At this stage participants would not have been used to the cognitive walkthrough procedure and may even have felt a little apprehensive. Another reason is that the base selection method is unusual. The standard method for selecting objects in a graphical display is to click on them. However the Resource Checks visualisation uses the *bounding box* technique. Although this technique has been used in a number of visualisations and is even used in the Windows Explorer™ file manager, it is not the most common selection technique and is often used in addition to the standard click selection mechanism. Since it is the only selection technique used in the Resource Checks visualisation, participants soon became frustrated when they failed to work out how to select a base and saw no means to do so. This frustration, compounded by the fact that this was the first visualisation participants were exposed to, may be the cause of the low intuitiveness rating.

It is also interesting to note that, although the Organix visualisation is probably the most unusual display, it ranked as reasonably intuitive. It is suspected that this was largely due to the limited functionality of the visualisation and the simplicity of the task.

6.2.3.3 Preference Ranking

The participants were asked to rank the visualisations in order of preference by assigning each visualisation a value from 1 to 6, with 1 indicating the most preferred visualisation and 6 indicating the least preferred visualisation. The average preferences for each visualisation are shown in table 6.5 with lower scores indicating more participants preferred the visualisation.

Visualisation	Mean	Mode	Median	Standard Deviation
Resource Checks	3.75	3.00	3.50	1.95
Route Checks	2.63	1.00	2.50	1.11
Parallel Coordinates	3.25	3.00	3.00	1.51
Organix	3.38	1.00	4.00	1.98
Mission Execution	3.75	4.00	4.00	2.06
Mission Dependencies	4.00	5.00	5.00	1.63

Table 6.5: Average preference ratings for each visualisation.

6.2 Cognitive Walkthroughs

Using the mean value as a rough guide, the results in table 6.5 would suggest that in terms of preference the visualisations could be ordered as in table 6.6.

Visualisation	Mean	Mode	Median
Route Checks	2.63	1.00	2.50
Parallel Coordinates	3.25	3.00	3.00
Organix	3.38	1.00	4.00
Resource Checks	3.75	3.00	3.50
Mission Execution	3.75	4.00	4.00
Mission Dependencies	4.00	5.00	5.00

Preference
↑
preferred more

preferred less
↓

Table 6.6: Visualisations ranked by preference.

The Route Checks visualisation was found to be the most preferred and this may be for reasons similar to those stated for its high intuitiveness rating.

The Parallel Coordinates was unfamiliar to all the participants and can appear very complex. Many of the participants initially felt overwhelmed by the display. However, the high preference ranking would seem to indicate that, once the participants had learnt how to use the display, they actually liked it and could see the benefits of using such a tool in command and control.

The least preferred visualisation was Mission Dependencies. There is no clear evidence suggesting why this should be but two reasons are proposed. Mission inter-dependencies and reused resources constitute a complex network structure. This is well suited to visualisation but to generate realistic datasets is a difficult task. The datasets used in the cognitive walkthroughs were simple and as such failed to show the visualisation to its full potential. The other reason why the visualisation may have had such a low rating is to do with established military doctrine. Recall that participants were asked to identify missions that could still go ahead despite their reliance on other failed missions. This situation is unusual in that standard military policy is to abort a mission under such extreme conditions. This may have led to a feeling that the visualisation was not suited to command and control and so was not preferred.

6.2.3.4 Comparison of Intuitiveness to Preference

Although neither ranking can be said to be accurate, a comparison of intuitiveness to preference raises a few interesting questions. The change in rank for each visualisation between the two categories is shown in figure 6.1.

The Mission Execution visualisation although the most intuitive ranked as one of the least preferred. The only reason we can think of for this dramatic change in position is that participants had difficulty decoding part of the visualisation. However, this should have affected the intuitiveness rating rather than the preference.

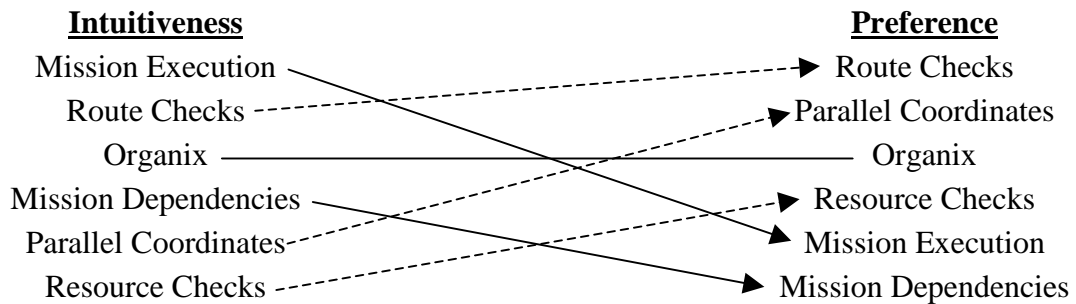


Figure 6.1: Comparison of intuitiveness to preference.

The Parallel Coordinates visualisation, although considered one of the least intuitive, was actually one of the most preferred. This is interesting as it suggests that although the visualisation may be difficult to learn, once learnt it is an efficient tool that people like to use.

6.2.4 Summary

An analysis of the data collected during the walkthroughs showed that there were problems with some of the visualisations. These problems were primarily due to dataset inconsistencies and the fact that the visualisations were early prototypes. Further analysis showed that most of the problems related to the GUI surrounding the visualisation and interaction problems rather than the view of the data within the visualisation. This led to the redesign of the affected components within each visualisation before their use in the trials.

The tasks were felt to be typical of the types of tasks that command and control personnel are faced with, suggesting that the military task requirements analysis had been correct. However, the wording of some of the tasks had to be altered to avoid misinterpretation.

Overall the usability of the software was deemed satisfactory and all of the military personnel who took part felt that visualisations would be useful tools to have in a command and control environment.

6.3 Experiment Design

The experiment was designed jointly with psychology experts from QinetiQ. However, since the trials required military personnel with experience in command and control, the trials were carried out solely by members of QinetiQ.

6.3.1 Aims and Measures

The aim of this experiment was to gather evidence that visualisations designed using the methodology, patterns, heuristics and user feedback, are more usable than those that have not

been designed with these considerations in mind. This evidence would also help to establish the validity of the heuristic evaluation technique.

We decided that the measures to be taken with respect to the aims of the experiment should be:

- *Accuracy*: Assessed by the number of correct responses to the task category questions.
- *Reaction times*: This is a performance measure. Reaction times were recorded from when the question was first displayed on screen to when participants entered the answer. Each question was preceded by a two second count down to alert participants that a question is about to appear.
- *Subjective assessment of Usability*: This would be captured using a modified SUMI questionnaire. This measures the usability dimensions *learnability*, *efficiency*, *affect*, *control*, and *helpfulness*, each of which are defined in section 6.2.3.1.

These are standard measures that correspond well with the usability factors used to classify the heuristics.

6.3.2 Requirements Analysis

Recall from section 4.1.3.3.2 that the task categories include *calculations*, *comparison*, *trends*, *relationships*, *aggregation*, *spatial*, *find*, *lookup*, and *optima*. We decided that these task categories fell into one of two broad classifications that could be used to discuss in more general terms the relative merits of each visualisation. The task classifications are defined as:

- *Simple tasks*: Tasks that are common, that do not require an in-depth analysis or exploration of the data, that can be achieved quickly i.e. do not require a great deal of time investment, and that do not require personal judgement in reaching the answer.
- *Complex tasks*: Tasks that require the user to explore the data, interact with various data attributes and correlate information between data objects and datasets. Complex tasks can often be decomposed into other complex tasks and simple tasks. Complex tasks are often ill-defined and may require significant human judgement. This may mean that different users reach different conclusions.

Examining the task categories and using the above definitions we classified *find*, *lookup*, and *optima* as simple tasks and the rest of the task categories as complex tasks.

In order to compare the abilities of each visualisation to assist users in the completion of different tasks it was necessary to determine a common set of task categories that all the visualisations could support. To do this each visualisation was examined to see which task

6.3 Experiment Design

categories they supported. This is shown in table 6.7. It should be noted that table 6.7 shows those task categories that were currently supported by the visualisations as in the cognitive walkthroughs, not those categories that the visualisations could be made to support if developed further.

	Resource Checks	Route Checks	Parallel Coordinates	Organix	Mission Execution	Mission Dependencies
Calculations	✓	✓	✓	✗	✗	✓
Comparison	✓	✓	✓	✓	✓	✓
Trends	✗	✗	✓	✗	✗	✗
Relationships	✓	✓	✓	✗	✓	✓
Aggregation	✓	✓	✓	✓	✓	✓
Spatial	✓	✓	✗**	✗	✗	✗
Find	✓	✓	✓	✗	✓	✓
Lookup	✓	✓	✓	✗	✓	✓
Optima	✓	✓	✓	✗	✓	✓

**Spatial tasks could be supported using grid references or latitude/longitude coordinates.

Table 6.7: Task categories supported by visualisations.

Grey rows indicate simple task categories.

Looking at table 6.7 it can be seen that the Resource Checks, Route Checks and Parallel Coordinates visualisations support the most task categories and of these task categories only *trends* and *spatial* cannot be supported by all three visualisations. In addition, the Resource Checks and Route Checks visualisations both display information about resources and due to its construction the Parallel Coordinates visualisation could be easily converted to display resource data. We decided to use these three visualisations in the experiment rather than the all six used in the cognitive walkthroughs because of this commonality across both the tasks they support and the data. In addition, reducing the number of visualisations helped to limit the complexity of the experiment.

The three chosen visualisations support all of the simple task categories and were all used in the experiment. Of the complex task categories, only the comparison category was used. This was considered sufficient to make the evaluation meaningful. The chosen task categories, *find*, *lookup*, *optima*, and *comparison*, are representative of the types of tasks carried out by command and control personnel.

Having determined a suitable set of task categories, the tasks shown in table 6.8 were developed. For consistency each task requires the participant to select a resource or set of resources. However to achieve this consistency the *lookup* task is actually a *find-lookup-find* task. Since this task consists only of simple tasks and we are interested in comparing visualisations at a simple vs. complex task level, the *lookup* task was considered acceptable.

6.3 Experiment Design

Task Category	Task
Find	Select the resources that have a power level less than 2.
Lookup	Select the resource that is a replacement for resource 17.
Optima	Select the resources that have the highest power level.
Comparison	Given the relationship between bases 0 and 1, select the base that is the most similar. (This will require you to select all the resources at the base.)

Table 6.8: Task category tasks.

To achieve the primary objectives of this evaluation it was necessary to have two distinct sets of visualisations as discussed in section 5.4. The wow-vis set, designed to produce high usability scores, and the just-a-vis set with lower usability scores. For the purposes of this experiment slightly modified versions of the visualisations used in the cognitive walkthroughs were used for the just-a-vis set. This was because the versions used in the cognitive walkthroughs were not developed enough to be used in an experimental setting. The wow-vis set consisted of the same visualisations but incorporating more heuristics and taking into account the extensive improvements determined from the analysis of the cognitive walkthrough data. Using the visualisations developed for the cognitive walkthroughs as the just-a-vis versions provided a baseline by which a comparison to the improved wow-vis versions could be made. The just-a-vis and wow-vis designs are described in detail in chapter five.

To summarise, the following requirements were identified:

- Two visualisation categories – wow-vis and just-a-vis.
- Three visualisations per visualisation category – Resource Checks, Route Checks, Parallel Coordinates.
- Three simple tasks per visualisation – find, lookup, and optima.
- One complex task per visualisation – comparison.

6.3.3 General Experiment Design

To aid clarity and to simplify the design, the experiment was split into two parts, one part for the simple tasks and one part for the complex tasks. However each part uses the same two (visualisation categories) by three (visualisations) repeated-measures design. The only difference is in the task categories presented to the participants. Repeated-measures means that all participants take part in all conditions i.e. all possible experimental design formats.

It was necessary to counterbalance each experiment in order to avoid *confounding* among variables. This is a learning or priming effect that makes it impossible to determine which variable is the cause of any observations. We used the Latin Square Design to counterbalance the factors in each experiment. The general procedure for the Latin Square Design counterbalancing technique is as follows. Given a set of factors 1 to n, the order of the first

6.3 Experiment Design

row of factors should be 1, 2, n, 3, n-1, 4, n-2, and so on. Subsequent rows simply increase the factor of the previous row by 1. This is illustrated in table 6.9. The specific factors and resulting counterbalanced presentation order used in each section of the experiment are described in sections 6.3.4 and 6.3.5.

Row #	Factor order				
1	A	B	E	C	D
2	B	C	A	D	E
3	C	D	B	E	A
4	D	E	C	A	B
5	E	A	D	B	C
6	B	C	A	D	E
7	C	D	B	E	A
8	D	E	C	A	B
9	E	A	D	B	C
10	A	B	E	C	D

Table 6.9: Latin Square Design counterbalance for five factors (A-E).

In order to make the results statistically powerful it was calculated that at least 52 participants would be required to attain a power level of 0.8 and ensure complete counterbalancing.

Prior to the experiment proper, participants were given instructions on how to use all three visualisations across both visualisation categories. This served as a brief training session before the actual trials commenced.

6.3.4 Simple Task Experiment Design

6.3.4.1 Visualisation Presentation Order

It can be seen from the requirements analysis that there are two distinct presentation blocks corresponding to the two visualisation categories and that within each block there are three visualisations which must each be presented three times, once for each simple task category. This is shown in figure 6.2.

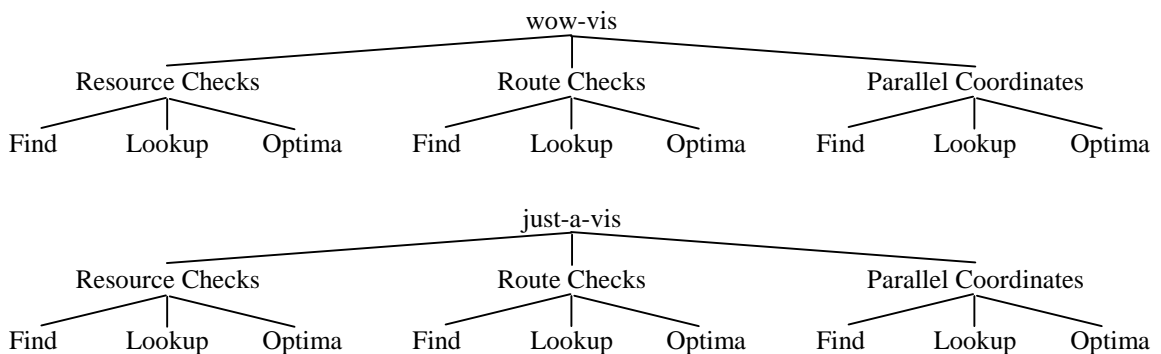


Figure 6.2: Presentation blocks for simple tasks.

As discussed, the order in which the visualisations are presented must be counterbalanced. In this case the factors that need to be balanced are the visualisation category (wow-vis and just-a-vis) and the visualisation format (Resource Checks, Parallel Coordinates and Route Checks). However, the visualisation category is not a factor that can be balanced on its own, therefore it is necessary to combine the categories and the formats before they can be counterbalanced. Referring to wow-vis as WV, just-a-vis as JV, and the three visualisation formats as 1,2 and 3, the factors to be balanced are WV1, WV2, WV3, JV1, JV2 and JV3.

The simple task categories also needed to be ordered to prevent participants becoming familiar with the task presentation order. However this does not need to be a strict ordering because during analysis the individual task categories will be combined under the heading simple tasks. Therefore the presentation order of the simple task categories was random.

It should be noted that unique datasets were used for each task category across all visualisations. This was to prevent users becoming familiar with the data during repeated exposure. In total eighteen datasets were used for the simple task experiment.

6.3.5 Complex Task Experiment Design

6.3.5.1 Visualisation Presentation Order

The complex task experiment differs to the simple task experiment in that only one task category is presented. In this case the task category is *comparison* and the task itself was of the form ‘Given the relationship between bases 0 and 1, select the base that is the most similar. (This will require you to select **all** the resources at the base.)’. This requires the participant to first of all identify the similarities between bases 0 and 1. To do this they must compare several data attributes for each of the resources within the two bases. Having established a relationship between bases 0 and 1 the participant must then examine the resources in the remaining bases to see if a similar relationship exists. However, to maintain the complexity of the task across visualisations it was necessary to choose three different similarity dimensions. If only one similarity dimension had been used the participants would have only had to determine the relationship between bases 0 and 1 once and so would not have needed to complete this part of the task when exposed to subsequent visualisations. The dimensions we chose for this complex task were resource quantity, resource power level and resource operational status. This results in the presentation blocks shown in figure 6.3.

6.3 Experiment Design

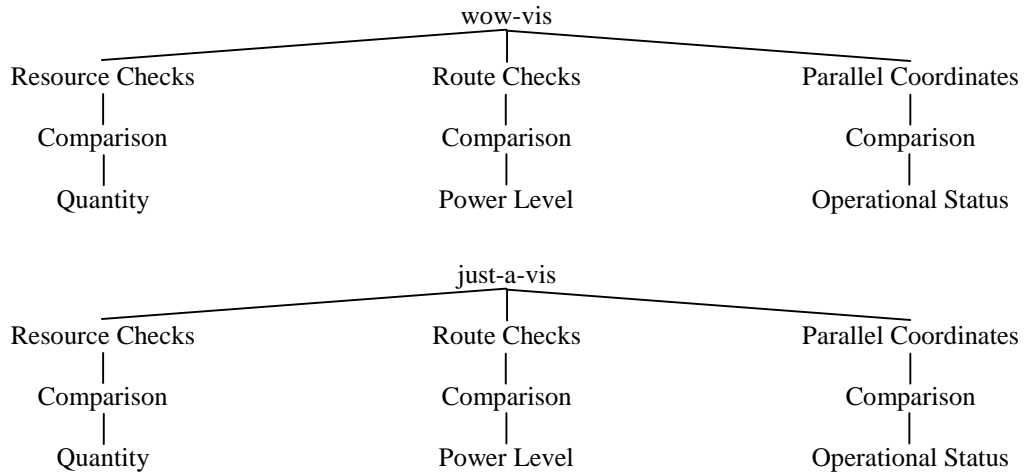


Figure 6.3: Presentation blocks for complex tasks.

As with the simple task experiment, the factors that need to be counterbalanced include the visualisation category and the visualisation format. In addition the three similarity dimensions also needed to be counterbalanced. However, applying the Latin Square Design counterbalancing technique resulted in unbalanced cycles. To compensate for this the first three rows were counterbalanced and then the counterbalancing was flipped so that the order would mirror the first run through. This is illustrated in figure 6.4.

1	Q	PL	OS	OS	Q	PL
2	PL	OS	Q	Q	PL	OS
3	OS	Q	PL	OS	PL	Q
4	PL	Q	OS	OS	PL	Q
5	OS	PL	Q	Q	OS	PL
6	Q	PL	OS	PL	Q	OS

Figure 6.4: Similarity dimension counterbalancing.

Key: Q = Quantity, PL = Power Level, OS = Operational Status

Referring to wow-vis as WV, just-a-vis as JV, the three visualisation formats as 1,2 and 3, and the similarity dimensions as Q, PL and OS, the factors to be balanced are WV1, WV2, WV3, JV1, JV2, JV3, Q, PL and OS. The counterbalanced presentation order for the first ten participants is shown in table 6.10.

1	WV1	Q	WV2	PL	JV3	OS	WV3	OS	JV2	Q	JV1	PL
2	WV2	PL	WV3	OS	WV1	Q	JV1	Q	JV3	PL	JV2	OS
3	WV3	OS	JV1	Q	WV2	PL	JV2	OS	WV1	PL	JV3	Q
4	JV1	PL	JV2	Q	WV3	OS	JV3	OS	WV2	PL	WV1	Q
5	JV2	OS	JV3	PL	JV1	Q	WV1	Q	WV3	OS	WV2	PL
6	JV3	Q	WV1	PL	JV2	OS	WV2	PL	JV1	Q	WV3	OS
7	WV2	Q	WV3	PL	WV1	OS	JV1	OS	JV3	Q	JV2	PL
8	WV3	PL	JV1	OS	WV2	Q	JV2	Q	WV1	PL	JV3	OS
9	JV1	OS	JV2	Q	WV3	PL	JV3	OS	WV2	PL	WV1	Q
10	JV2	PL	JV3	Q	JV1	OS	WV1	OS	WV3	PL	WV2	Q

Table 6.10: Counterbalanced presentation order for the complex task experiment.

First ten participants only.

As with the simple task experiment, to prevent participants becoming familiar with the data a total of six unique datasets were used for each similarity dimension exposure.

6.3.6 Procedure

The procedure for the trials was as follows:

- Participants were given written instructions and briefed on the simple tasks to be completed.
- Participants then completed a practice session to help familiarise themselves with each visualisation format. At this point participants were given the opportunity to ask any questions before commencing the actual trials.
- Participants then completed the actual trial (with an imposed 2 minute time limit).
- On completion of experiment block i.e. exposure to a visualisation across all task categories, the participants completed the modified SUMI questionnaire.
- The procedure was then repeated until the participant had been exposed to all visualisation formats across both visualisation categories. This was done in accordance with the counterbalance ordering.
- After completion of the simple tasks, the participants were given a short refresher course and briefed on the complex tasks to be completed.
- Participants then completed a practice session to help familiarise themselves with each visualisation format.
- Participants then completed the actual trial (with an imposed 5 minute time limit).
- On completion of each visualisation the participants completed the modified SUMI questionnaire.
- Finally the experimenter debriefed the participants. This debriefing covered the main aims of the study and gave the participants an opportunity to comment and ask questions.

6.4 Results

Recall that the measures taken were reaction time, accuracy, and subjective assessment of usability. In this case the subjective assessment was conducted using a modified SUMI questionnaire that measures learnability, efficiency, affect, control, and helpfulness, these measures are defined in section 6.2.3.1. Recall also that the heuristic evaluation is based on heuristics classified by time to learn, speed of performance, rate of errors, retention over time, and subjective satisfaction. For the purpose of this thesis we consider learnability to be equivalent to time to learn, reaction time equivalent to performance, and accuracy used to indicate rate of errors. In addition, the five SUMI measures will be taken to constitute subject satisfaction and give an indication of the overall perceived usability of each visualisation.

6.4 Results

It should be noted that, although the experiment was originally designed to be completed by 52 participants, QinetiQ only carried out the trials with 25 participants. Since the experiment was conducted entirely by QinetiQ, this decision was beyond the control of the researcher. This meant that perfect counterbalancing was not achieved and the power of the experiment was reduced.

The data collected were analysed using the appropriate analysis of variance (ANOVA) tests. In the following sections the 5% significance level was adopted ($p < 0.05$).

6.4.1 Resource Checks

6.4.1.1 Overview of Results

The usability ratings assigned to each visualisation category using the heuristic evaluation technique and the observed results are shown in tables 6.11 and 6.12 respectively.

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	0	1
Speed of performance	-2	7
Rate of errors	-2	14
Retention over time	0	0
Subjective Satisfaction	-1	3
Overall rating	-5	25

Table 6.11: Resource Checks usability ratings using heuristic evaluation.

Measure	Task	just-a-vis		wow-vis	
Reaction Time (secs)	Simple	23.2	>	15.2	$p < 0.001$
	Complex	88.1	>	83.4	
Accuracy (%)	Simple	97.5	<	98.3	
	Complex	88.5	>	84.6	
Overall Usability (%)	Simple	90	<	93	
	Complex	56	<	60	

Table 6.12: Resource Checks observed results for simple and complex tasks.

The majority of the observed results show no significant difference between the two versions in terms of usability. The exception to this is the simple task reaction time measure. In this case the wow-vis appears to be significantly better than the just-a-vis. This was as predicted.

The just-a-vis and wow-vis versions of the Route Checks visualisation have a number of common design features. This may help to explain why there were few significant observations. In particular the table part of each visualisation, which is vital to the simple tasks, is almost identical.

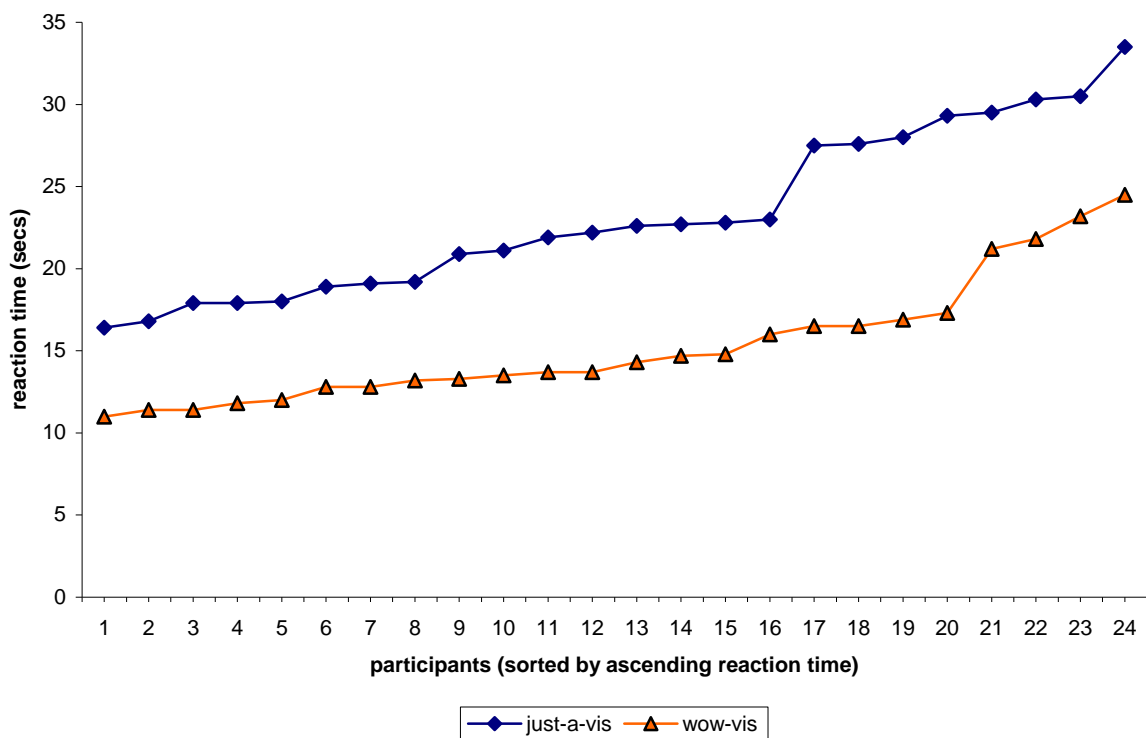
One of the main differences between the two versions is the way in which base resources are

displayed. It was thought that the regular layout used in the wow-vis would assist the user in their tasks. However, some of the heuristics that contributed to this decision such as ‘use connotative mappings’, ‘use colour carefully’, and ‘occlusion is undesirable’, have little or no impact on the complex task but greatly influence the heuristic evaluation ratings. It is possible that for a different set of tasks the heuristics that led to these designs would have greater effect. In this case it could be argued that although applying the heuristics has not resulted in a significantly better design, neither has it resulted in a worse design, but the wow-vis does appear to have the potential to support a wider range of tasks with higher levels of usability. Further investigation would be necessary to confirm this hypothesis.

In terms of overall perceived usability, the small difference in SUMI scores (< 5%) shows that there is no significant difference between the wow-vis and just-a-vis. This corresponds with the small difference in subjective satisfaction scores assigned using the heuristic evaluation.

6.4.1.2 Analysis of Simple Task Results

In terms of reaction time the observed results show that the wow-vis is significantly better than the just-a-vis. This was as predicted and is clearly illustrated in figure 6.5. Due to the high accuracy levels achieved using the Resource Checks visualisation, only correct answers have been shown to maintain the clarity of the graph.



The mean reaction time for each participant was calculated for correct answers across all simple tasks.

The mean reaction times across all participants were then sorted into ascending order.

Figure 6.5: Resource Checks – simple tasks – reaction time (correct answers only).

6.4 Results

The results presented in table 6.12 and figure 6.6 show that on average, for simple tasks, the wow-vis version provides slightly greater accuracy and significantly improved performance.

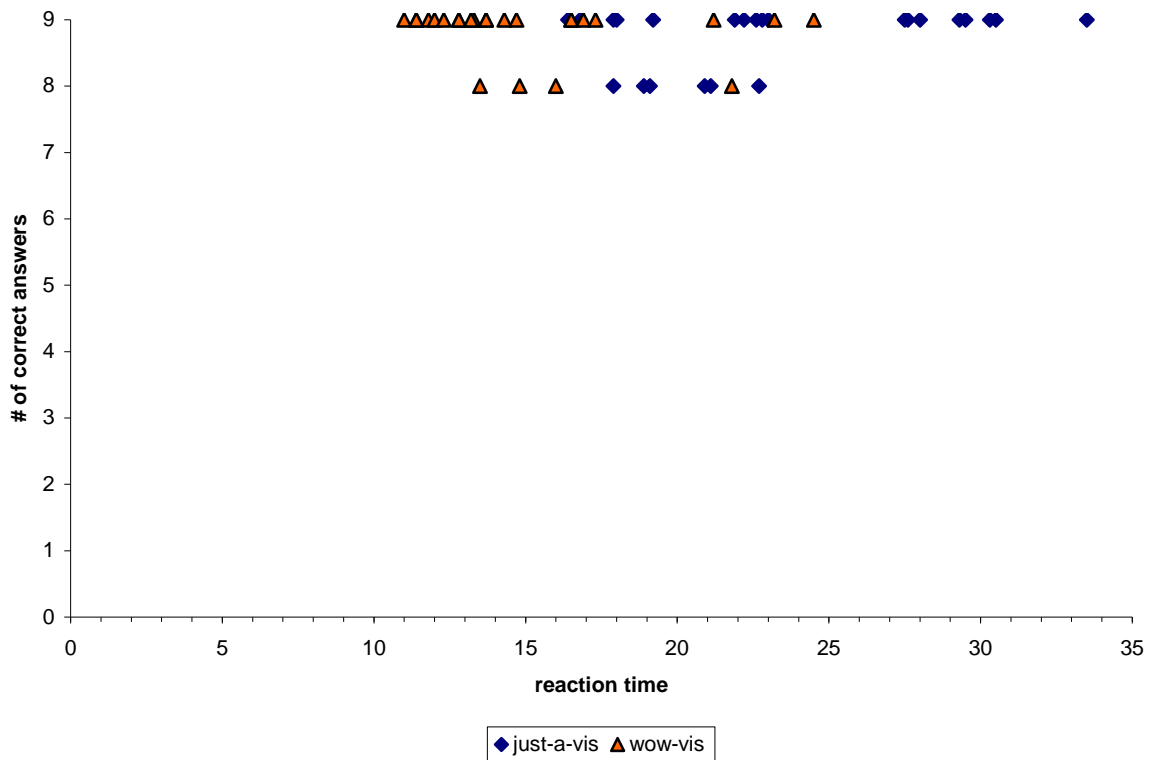


Figure 6.6: Resource Checks – simple tasks – reaction time vs. accuracy.

The SUMI scores presented in figure 6.7 suggest that there is no significant difference in terms of perceived usability between the wow-vis and the just-a-vis. However, the very high scores indicate that both the wow-vis and the just-a-vis are well suited to support command and control personnel in everyday simple tasks.

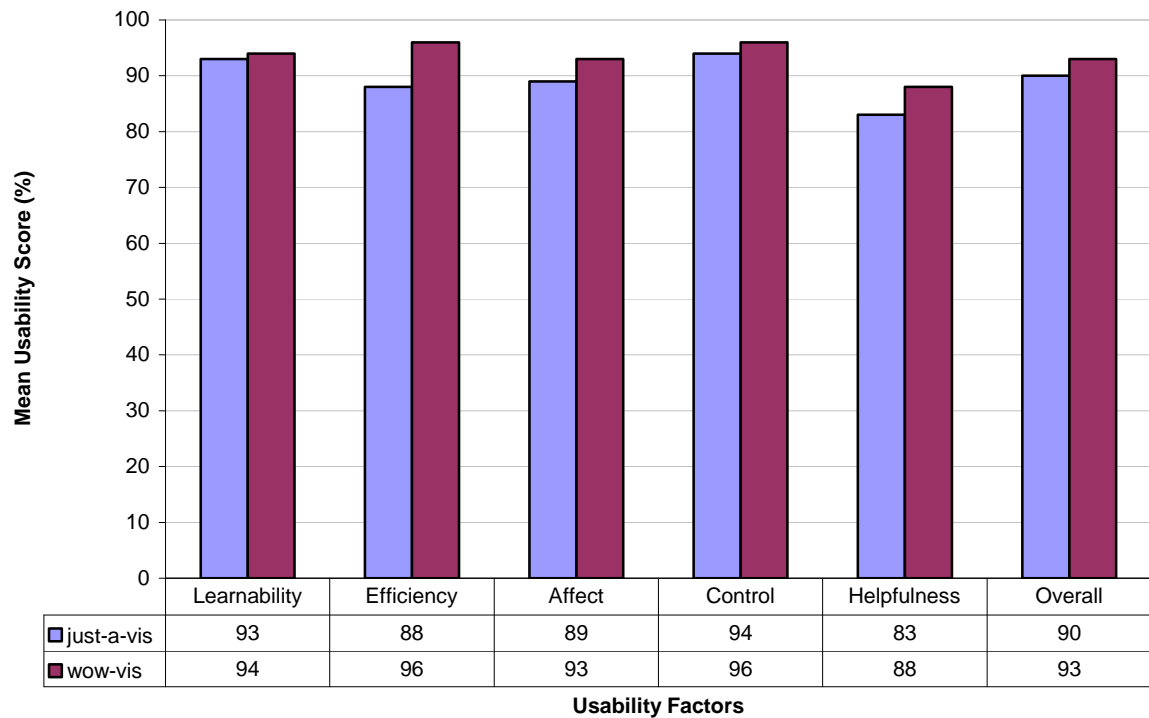


Figure 6.7: Resource Checks – simple tasks – SUMI scores.

6.4.1.3 Analysis of Complex Task Results

The sorted mean reaction times for each participant's correct answers across all complex tasks are presented in figure 6.8. The observed results indicate that the reaction times for each visualisation category are, in general, comparable.

As for the simple tasks, the SUMI scores for the complex tasks, presented in figure 6.9, suggest no perceived difference in usability between the wow-vis and just-a-vis. It is interesting to note that, in terms of learnability, it would appear that the participants feel that the wow-vis is easier to learn. We hypothesise that this may be partly due to the simplified resource selection and base expansion facilities available in the wow-vis.

6.4 Results

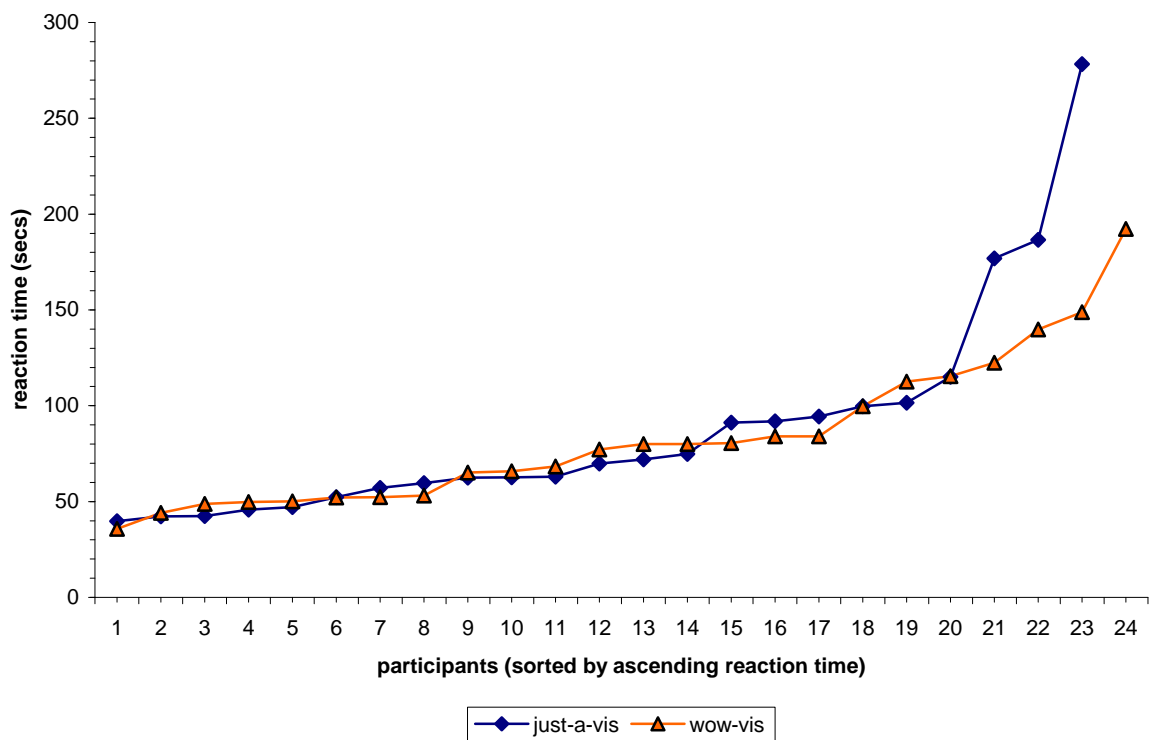


Figure 6.8: Resource Checks – complex tasks – reaction time (correct answers only).

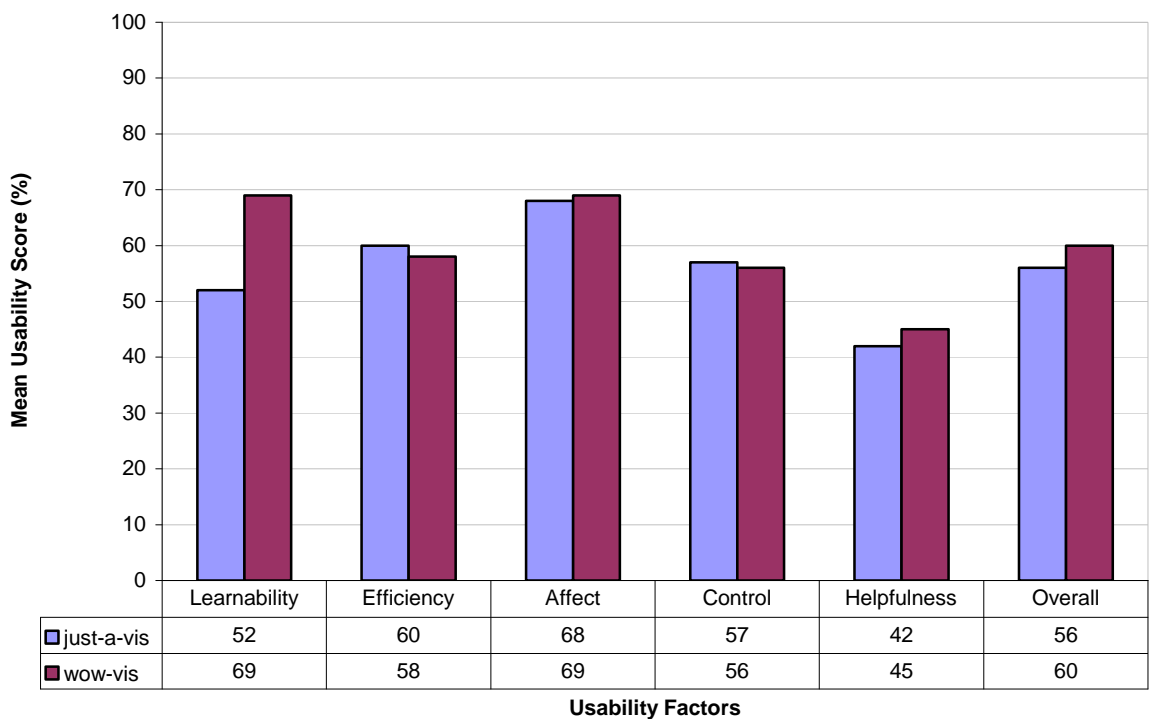


Figure 6.9: Resource Checks – complex tasks – SUMI scores.

6.4.2 Parallel Coordinates

6.4.2.1 Overview of Results

The usability ratings assigned to each visualisation category using the heuristic evaluation technique and the observed results are shown in tables 6.13 and 6.14 respectively.

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	0	0.2
Speed of performance	-2	9
Rate of errors	-2	10
Retention over time	0	0
Subjective Satisfaction	-1	2
Overall rating	-5	21.2

Table 6.13: Parallel Coordinates usability ratings using heuristic evaluation.

Measure	Task	just-a-vis		wow-vis	
Reaction Time (secs)	Simple	23.7	>	22.02	
	Complex	190.6	>	152.7	
Accuracy (%)	Simple	93.7	>	82.2	$p < 0.001$
	Complex	65.4	>	61.5	
Overall Usability (%)	Simple	75	<	79	
	Complex	55	<	57	

Table 6.14: Parallel Coordinates observed results for simple and complex tasks.

The just-a-vis and wow-vis versions of the parallel coordinates visualisation have very similar designs. This is highlighted by the fact that neither the mapping evaluation nor the metric evaluation could conclusively rank the two formats. It may also explain why the majority of the measures taken show no significant difference.

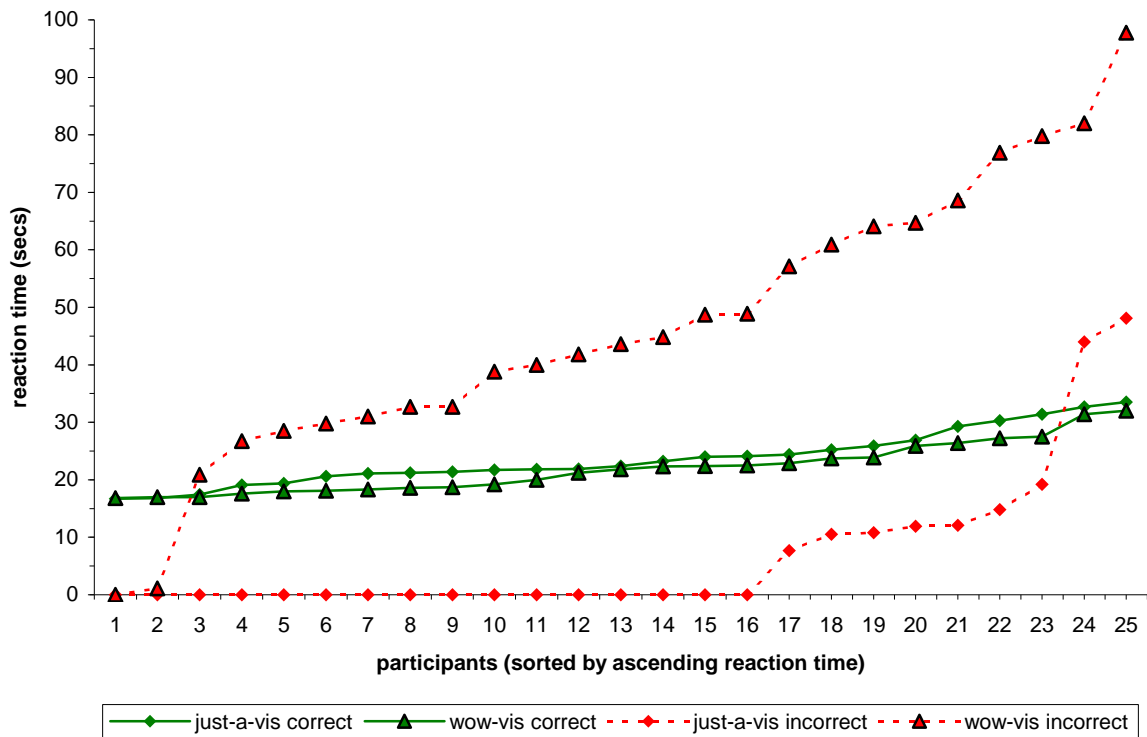
The heuristic evaluation predicted that for each of the usability factors the wow-vis version should be more usable than the just-a-vis version. However the differences between the scores for each factor are small. Given that the majority of the observed usability measures show no significant difference between the two versions we could conclude that the low heuristic scores indicate little or no difference in terms of usability. Conducting further experiments and calibrating the heuristic ratings accordingly would help to verify this hypothesis.

The only significant observation was that the just-a-vis version appears to provide greater accuracy for simple tasks than the wow-vis. This contradicts the prediction made by the heuristic evaluation. The facilities available in the wow-vis version such as being able to swap axis positions, highlight all the data items that pass through a particular value, and receive audio confirmation when ticks are grabbed, do not seem to have improved accuracy. One possible explanation is that the minimal training given during the experiment may have meant

that the participants failed to recall these useful facilities, or used them incorrectly, during task execution.

6.4.2.2 Analysis of Simple Task Results

In figure 6.10 the mean reaction times for each participant’s correct and incorrect answers across all simple tasks have been calculated and then sorted into ascending order. Although not significant, on the whole, the reaction times for executing the simple tasks correctly using the wow-vis are slightly faster than for the just-a-vis version. However, it is clear that nearly all the participants got at least one incorrect answer using the wow-vis. This is reflected in the accuracy values found in table 6.14. Although not presented, the data reveals that the majority of these incorrect answers occurred during the lookup task.



The mean reaction time for each participant was calculated for both correct and incorrect answers across all simple tasks. The mean reaction times across all participants were then sorted into ascending order.

Reaction times of zero seconds indicates no correct/incorrect answers.

Figure 6.10: Parallel Coordinates – simple tasks – reaction time.

The results presented in table 6.14 and figure 6.11 show that for simple tasks the just-a-vis appears to provide greater accuracy with comparable reaction times. This was not as predicted using the heuristic evaluation.

6.4 Results

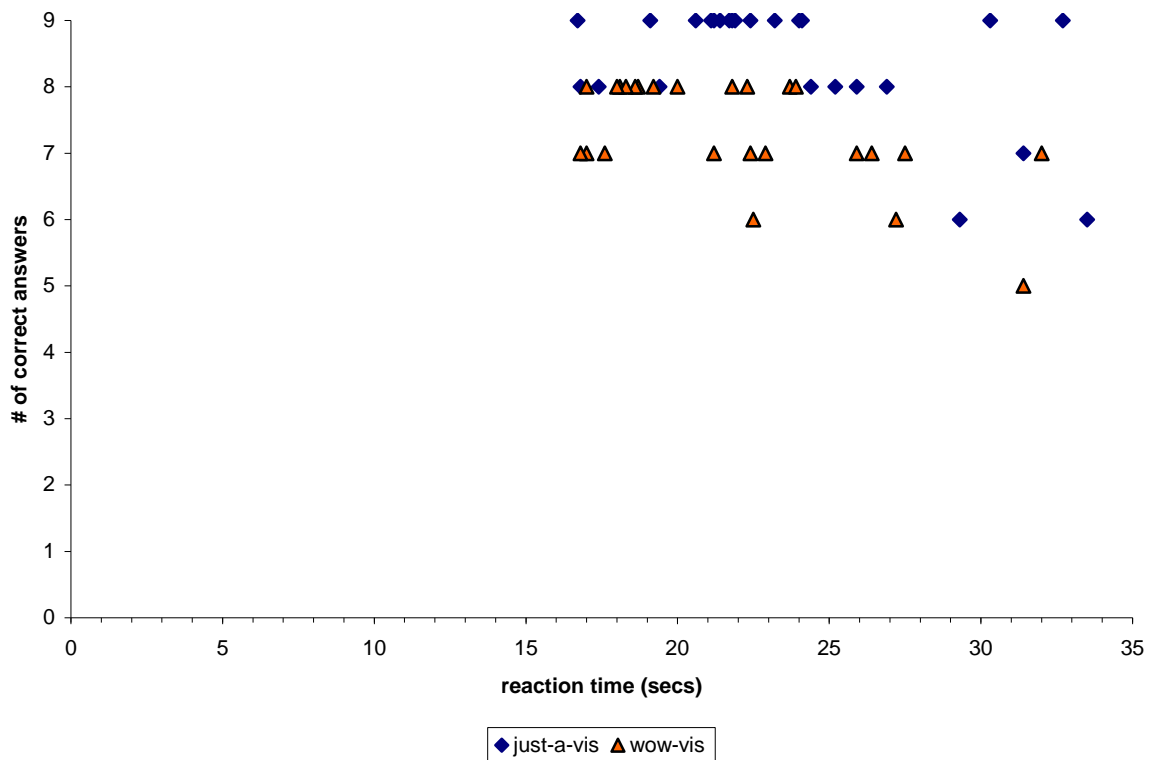


Figure 6.11: Parallel Coordinates – simple tasks – reaction time vs. accuracy.

The SUMI scores presented in figure 6.12 show no significant difference between the just-a-vis and wow-vis formats for the simple tasks.

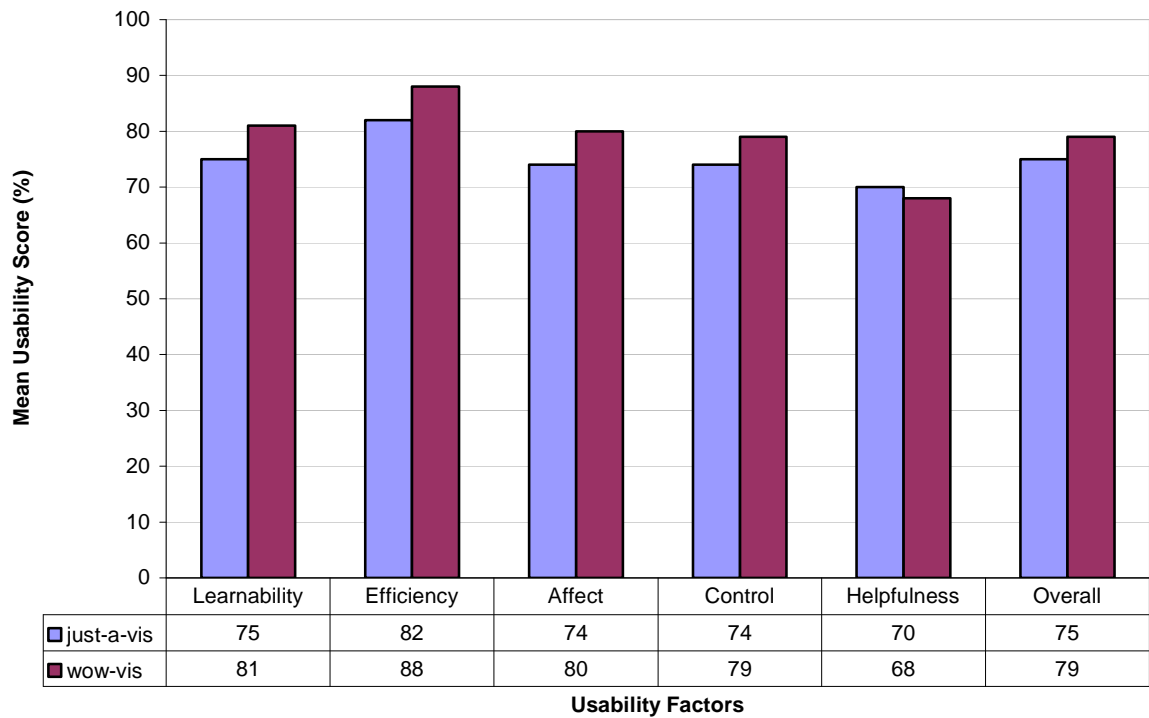
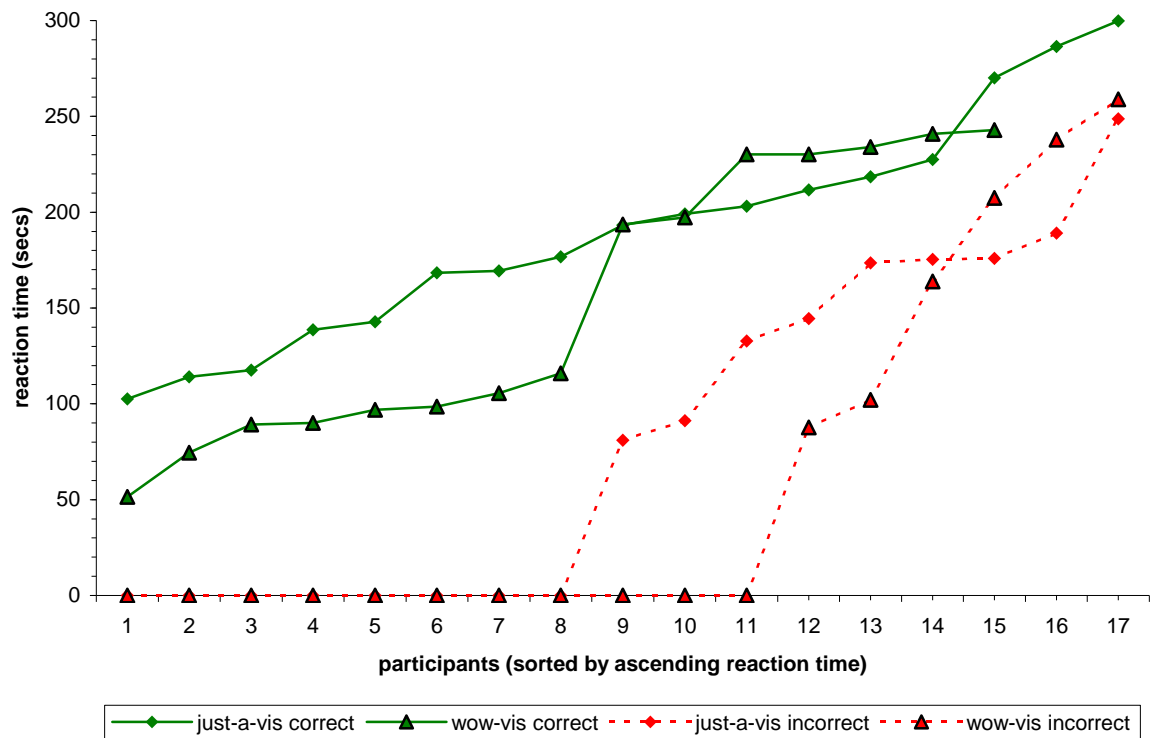


Figure 6.12: Parallel Coordinates – simple tasks – SUMI scores.

6.4.2.3 Analysis of Complex Task Results

The mean reaction times for each participant's correct and incorrect answers across all complex tasks have been calculated and then sorted into ascending order as presented in figure 6.13. The observed results indicate that reaction times for correct answers are generally faster using the wow-vis, but that on the whole the just-a-vis allows tasks to be completed slightly more accurately.



The mean reaction time for each participant was calculated for both correct and incorrect answers across all complex tasks. The mean reaction times across all participants were then sorted into ascending order.

Reaction times of zero seconds indicates no correct/incorrect answers.

Figure 6.13: Parallel Coordinates – complex tasks – reaction time.

6.4 Results

As for the simple tasks, the SUMI scores for the complex tasks, presented in figure 6.14, show no difference between the just-a-vis and wow-vis.

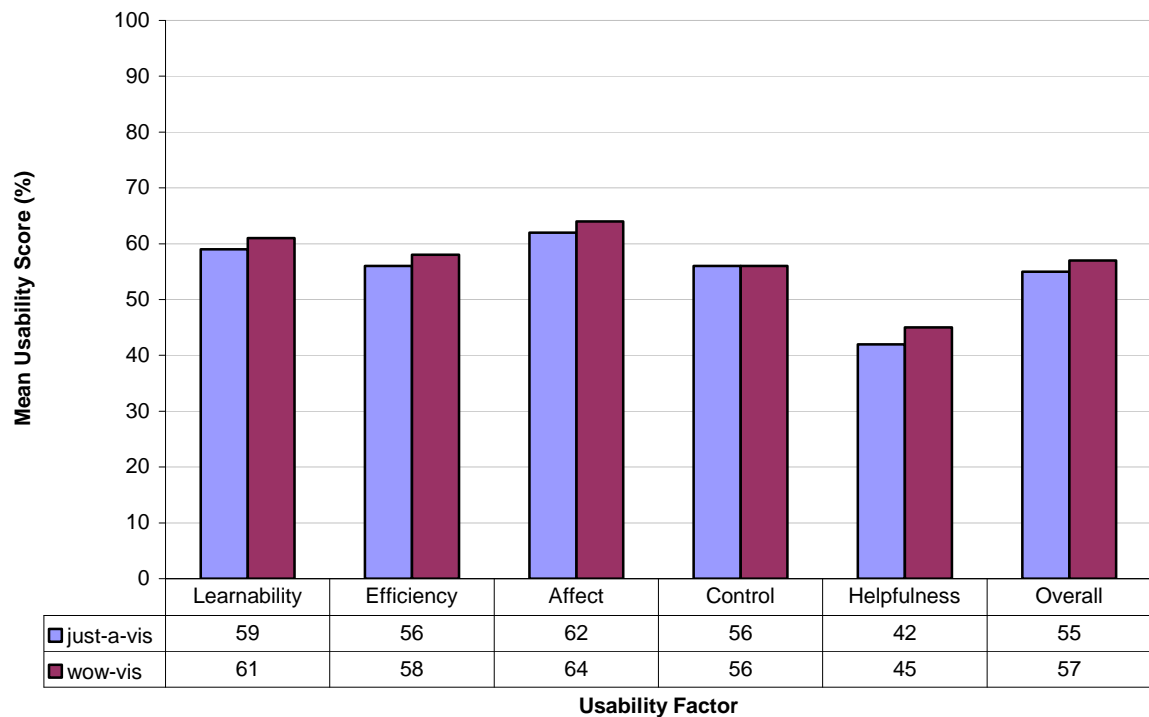


Figure 6.14: Parallel Coordinates – complex tasks – SUMI scores.

6.4.3 Route Checks

6.4.3.1 Overview of Results

The usability ratings assigned to each version using the heuristic evaluation technique and the observed results are shown in tables 6.15 and 6.16 respectively.

Usability Factor	Score	
	just-a-vis	wow-vis
Time to learn	-0.4	1
Speed of performance	-5	26
Rate of errors	-6	32
Retention over time	0	0
Subjective Satisfaction	-3	6
Overall rating	-14.4	65

Table 6.15: Route Checks usability ratings using heuristic evaluation.

Measure	Task	just-a-vis		wow-vis	
Reaction Time (secs)	Simple	74.5	>	25.0	$p < 0.001$
	Complex	190.4	>	164.7	
Accuracy (%)	Simple	42.7	<	92.4	$p < 0.001$
	Complex	26.9	<	53.8	$p < 0.05$
Overall Usability (%)	Simple	56	<	84	$p < 0.001$
	Complex	57	<	58	

Table 6.16: Route Checks observed results for simple and complex tasks.

The majority of the observed results are significant and all match the predictions made by the heuristic evaluation. In contrast to the Parallel Coordinates and Resource Checks visualisations, the just-a-vis and wow-vis versions of Route Checks have markedly different designs in terms of their application of the heuristics. This difference in designs is highlighted by the large difference in overall scores assigned using the heuristic evaluation and judging by the observed results has a significant effect on usability. In this case we can conclude that both the heuristic evaluation and metric evaluation have correctly predicted the most usable version.

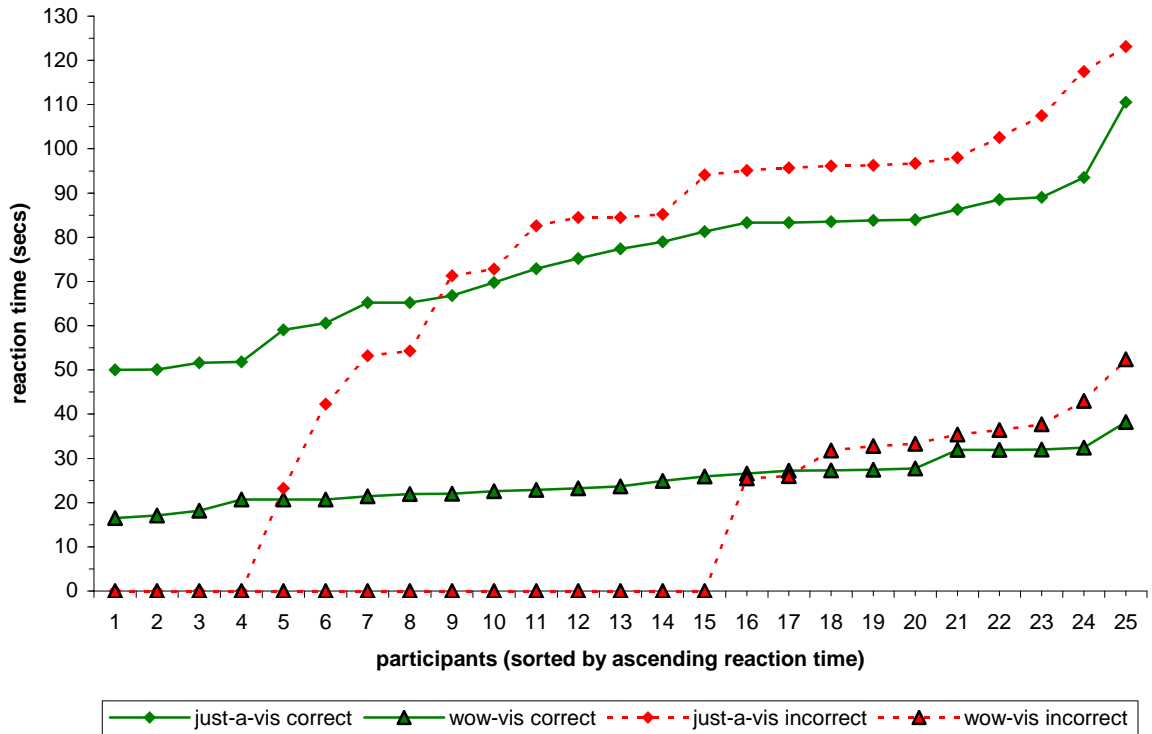
The application of the ‘filter’ heuristic in the wow-vis could be one possible reason why the wow-vis is so much more usable than the just-a-vis, although there are obviously a number of other significant differences. The wow-vis allows the user to filter out a much larger quantity of task irrelevant information than the just-a-vis. This reduction in non-relevant task information would most likely help to reduce reaction time and improve accuracy. Interestingly providing filtering facilities is one of the most commonly proposed heuristics. This suggests that prioritising the heuristics in some way may help to improve the accuracy of the heuristic evaluation.

The overall perceived usability for the simple tasks indicates that the wow-vis is significantly more usable. Conversely, the overall usability scores for the complex tasks are almost identical. This would suggest that, despite significantly faster reaction times and accuracy, participants perceive both visualisations as being equally usable when performing the complex tasks.

6.4.3.2 Analysis of Simple Task Results

The significant differences in reaction times between the two visualisation categories are emphasised in figure 6.15. It is also evident that even when the tasks are completed incorrectly, the wow-vis still allows much faster reaction times. We believe that these significant improvements in performance are primarily due to the filter facilities available in the wow-vis.

6.4 Results



The mean reaction time for each participant was calculated for both correct and incorrect answers across all simple tasks. The mean reaction times across all participants were then sorted into ascending order.

Figure 6.15: Route Checks – simple tasks – reaction time.

The results presented in table 6.16 and figure 6.16 show that, for the simple tasks, both accuracy and reaction time are significantly better in the wow-vis than in the just-a-vis

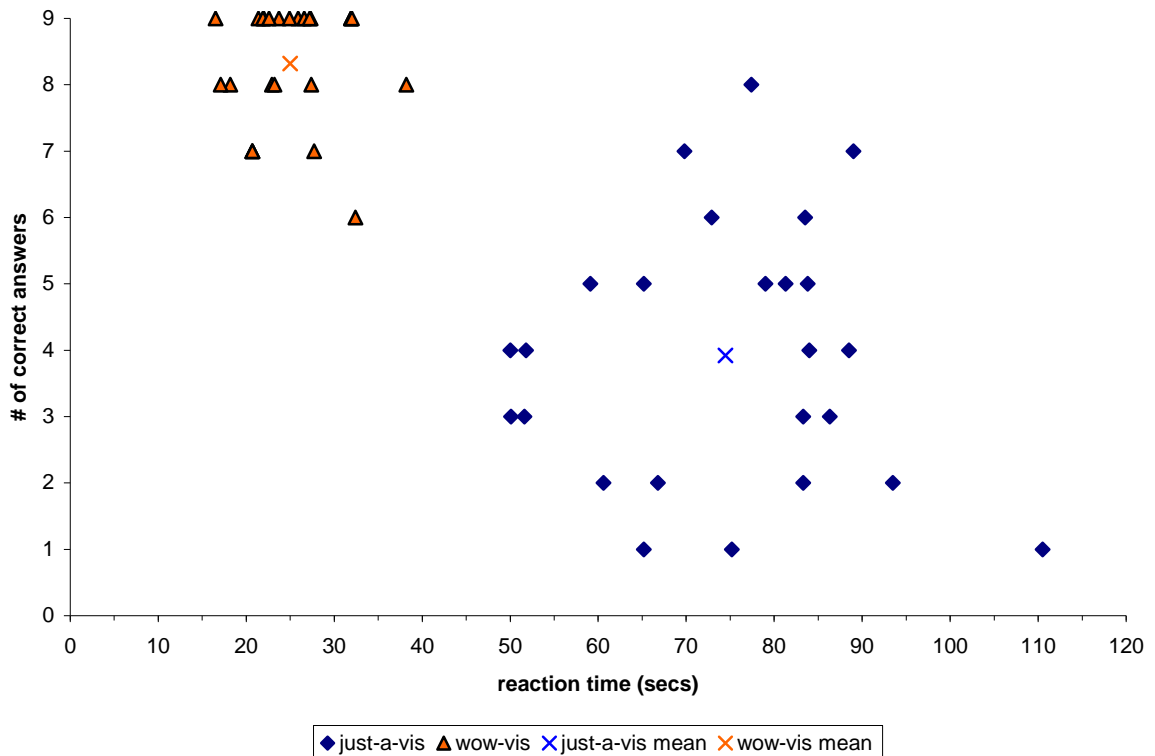


Figure 6.16: Route Checks – simple tasks – reaction time vs. accuracy.

The SUMI scores presented in figure 6.17 confirm the prediction made using the heuristic evaluation that the perceived usability of the wow-vis is higher than the just-a-vis. Unlike the Parallel Coordinates and Resource Checks SUMI scores, the difference in scores for each usability factor is large. These large differences in scores, as well as the large differences in reaction time and accuracy, are hinted at in the values assigned using the heuristic evaluation (table 6.15). As stated, we believe this is due to distinct differences between the just-a-vis and wow-vis designs. In particular, the inclusion of multiple filters in the wow-vis.

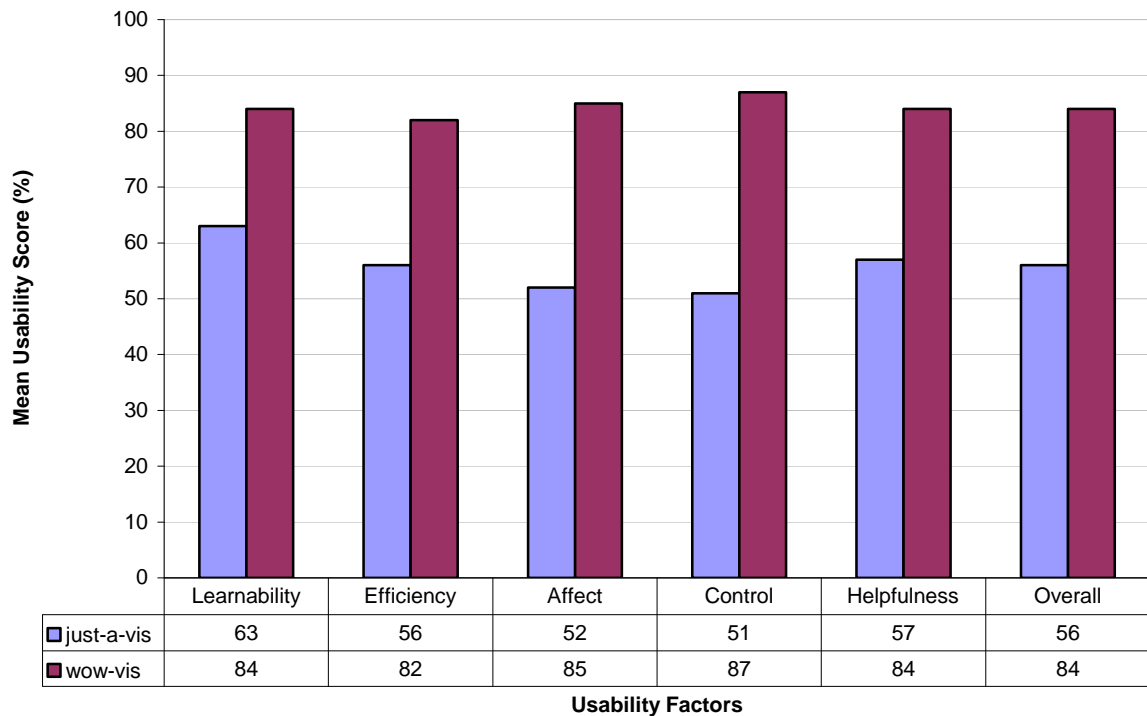


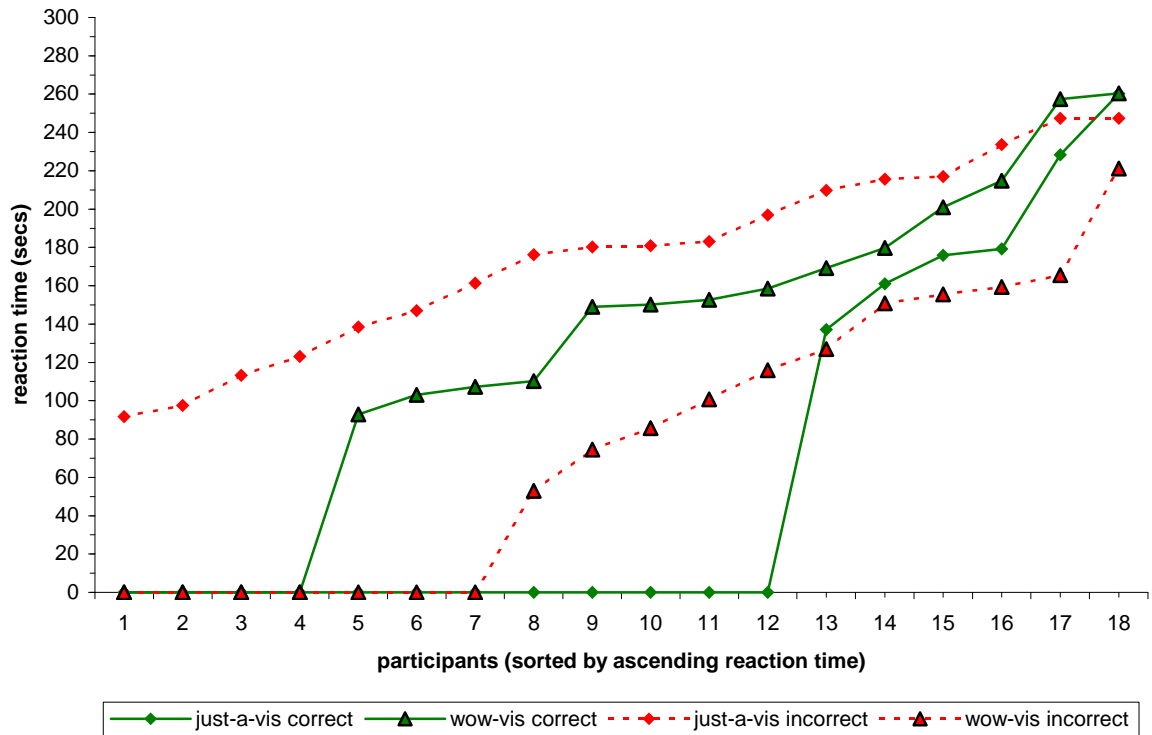
Figure 6.17: Route Checks – simple tasks – SUMI scores.

6.4.3.3 Analysis of Complex Task Results

The mean reaction times for each participant's answers across all complex tasks are presented in figure 6.18. The observed results indicate that reaction times are on average faster, and accuracy significantly higher, when using the wow-vis than when using the just-a-vis.

Unlike the simple tasks, several of the SUMI scores for the complex tasks, presented in figure 6.19, are identical or within one or two percent of each other. As previously stated, this suggests that despite improved reaction time and accuracy, the participants perceived usability of the just-a-vis and wow-vis is approximately equal. We hypothesise that because the wow-vis does not provide any direct facilities to assist with the complex task, although differences in the design do help indirectly (e.g. filters, use of colour, etc.), the participants did not perceive the wow-vis as being more usable.

6.4 Results



The mean reaction time for each participant was calculated for both correct and incorrect answers across all complex tasks. The mean reaction times across all participants were then sorted into ascending order.

Reaction times of zero seconds indicates no correct/incorrect answers.

Figure 6.18: Route Checks – complex tasks – reaction time.

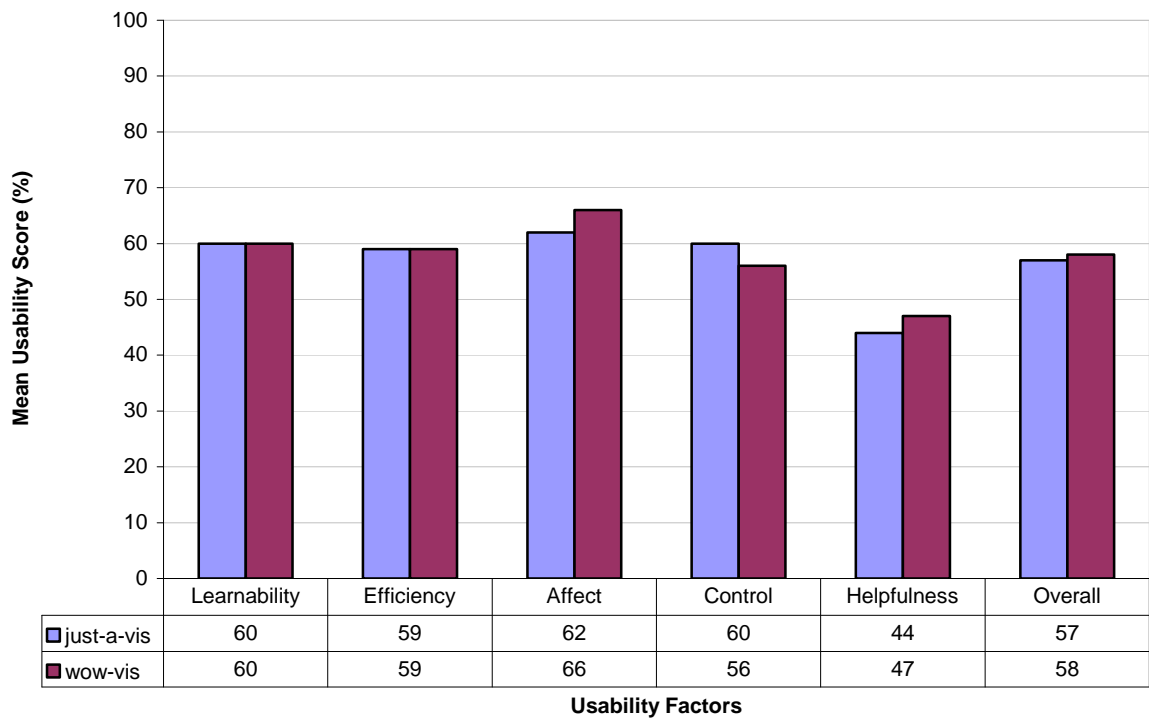


Figure 6.19: Route Checks – complex tasks – SUMI scores.

6.4.4 Ranking of Perceived Usability

Since each visualisation format (Parallel Coordinates, Resource Checks, Route Checks) is capable of supporting the same set of tasks, it is possible to rank the visualisations in terms of their perceived usability scores. This is shown in tables 6.17 and 6.18 for simple and complex tasks respectively.

Visualisation	Version	Overall Mean Usability Score (%)	Perceived Usability
Resource Checks	wow-vis	93	↑ more usable
Resource Checks	just-a-vis	90	
Route Checks	wow-vis	84	↓ less usable
Parallel Coordinates	wow-vis	79	
Parallel Coordinates	just-a-vis	75	
Route Checks	just-a-vis	56	

Table 6.17: Ranking of perceived usability for simple tasks.

Visualisation	Version	Overall Mean Usability Score (%)	Perceived Usability
Resource Checks	wow-vis	60	↑ more usable
Route Checks	wow-vis	58	
Parallel Coordinates	wow-vis	57	↓ less usable
Route Checks	just-a-vis	57	
Resource Checks	just-a-vis	56	
Parallel Coordinates	just-a-vis	55	

Table 6.18: Ranking of perceived usability for complex tasks.

The heuristic evaluation predicted that in terms of usability the wow-vis would be more usable in each case. The overall perceived usability scores for both the simple and complex tasks show that the wow-vis version of each visualisation is perceived as more usable than its just-a-vis equivalent. Therefore, there is evidence to suggest that the prediction made by the heuristic evaluation is correct in this case.

The Resource Checks wow-vis has the highest usability scores for both the simple and complex tasks. This suggests that it is particularly appropriate for the command and control domain.

6.5 Conclusion

The main objectives of these evaluations was to find evidence that supports the predictions made by the design evaluation techniques presented in section 4.4.2, in particular the heuristic evaluation, and the suitability of visualisation to command and control.

Initially we analysed the command and control domain and developed a set of prototype visualisations capable of supporting both the data and tasks found in command and control.

We then conducted a series of cognitive walkthroughs using the prototypes. Feedback from these walkthroughs indicated that the task requirements analysis was valid and that the visualisations were felt to be appropriate tools to use in a command and control environment.

A second, more rigorous, experiment was then conducted, aimed at finding evidence to support the hypothesis that visualisations designed using the visualisation heuristics and patterns are more usable than those that do not, as predicted by the design evaluation techniques. Two sets of visualisations were developed, a basic set referred to as just-a-vis, and an equivalent, but theoretically more usable, set referred to as wow-vis. So that any usability differences measured would be due to differences in heuristics, the two sets of designs used the same perceptual encodings, which resulted in perceptually similar designs. Participants were then asked to complete a series of simple and complex tasks using visualisations from each set.

Looking at the results of the empirical evaluation and comparing them to the predictions made by the design evaluation techniques, especially the heuristic evaluation, a number of issues have become apparent.

One such issue is that of design similarity. The results show that similar designs, such as those used for the Parallel Coordinates and Resource Checks visualisations, are difficult to rank in terms of their usability using the design evaluation techniques presented in section 4.4.2. In other words the designs could be thought of as being equally usable. In some cases this was predicted by the mapping and metric evaluation techniques. In contrast, the heuristic evaluation always predicted that the wow-vis would be more usable than the just-a-vis. This may still be correct. It may be that the wow-vis is actually more usable than the just-a-vis but by such a small amount that it cannot be accurately measured. However, it is more likely that the scores generated by the heuristic evaluation need to be calibrated. At present the heuristic evaluation can only rank designs qualitatively, i.e. it can only be used to state whether or not one visualisation is only slightly more usable or much more usable than another.

Unfortunately it is not known how big a difference in visualisation scores constitutes a 'slightly' more usable visualisation, a 'much' more usable visualisation, or if there is 'no' difference in usability. It is possible that the individual usability scores for the Parallel Coordinates and Resource Checks visualisations, which are all less than fifteen, constitute no difference in usability. Only further experience using the heuristic evaluation technique will be able to determine this calibration.

Looking at the tasks used in the evaluation it is evident that in some cases it would be immaterial whether or not the heuristics were adhered to. This is especially true for the complex comparison task. For example, when trying to identify similar bases in the Resource Checks visualisation, the actual colour of the resources has no affect on the task as long as the

colours are unique. Thus the heuristic ‘use colours carefully’ has no influence even though it increases the usability score for the wow-vis version. Alternatively, if the task had been to always identify resources of power level one, then highlighting those resources by using a very distinct colour would be appropriate. This suggests that applying the heuristic evaluation on a task-by-task basis may be beneficial. However, the heuristic evaluation was not designed to be used in this way and in fact providing an overall usability score rather than a task-specific usability score may be better since different tasks may lead to contradictory heuristics.

Finally, using the empirical evidence we can say that for significantly different designs in which a number of heuristics have been violated or poorly implemented, the heuristic evaluation technique appears to be able to predict which visualisation design should be more usable.

Chapter 7: CONCLUSIONS AND FUTURE WORK

This chapter provides a brief summary of the research, the conclusions that can be drawn, and directions for future work.

7.1 Summary

It has been established that visualisations can be used to help users cope with increasingly large quantities of complex data. In essence acting as tools that they can use to achieve the goals relevant to the domain in which they work. However, as with any interactive software, designing these visualisations is a difficult task.

Visualisation design relies on the fact that the human visual system is capable of processing large quantities of information rapidly and accurately. As a result, visualisation designers make use of principles established by perceptual psychologists in order to design perceptually effective visualisations. However, the human visual system does have limitations and our knowledge of how it works is incomplete. In addition, human memory and reasoning also play an important part when interpreting information and performing tasks. Thus perception cannot be the only driving force behind visualisation design.

Visualisation designers have developed a variety of techniques that help the user to interpret large quantities of complex data. Recently designers have started to gather and document these techniques in the form of heuristics. Unfortunately these disparate sources tend to be unstructured and may lack important information such as when to use the technique, the rationale behind the technique, examples of the technique in use, etc. Architecture, software engineering, and HCI have encountered similar problems with heuristics and have begun to use patterns as a way of recording all of the meta-information related to a technique. In short patterns capture experience based design knowledge, that has been proven to work, in a structured format. In addition because each pattern has a unique title and includes examples of how the pattern has been applied they can act as a common language between all members of a design team including users.

Visualisation is a form of human-computer interaction and visualisation systems are software systems. This suggests that some of the methodologies and techniques used in these disciplines could also be used in visualisation design. Of particular relevance are the structured approach taken in software engineering, the application of HCI techniques throughout the design process, and the emphasis of HCI on producing usable systems.

An analysis of existing methodologies identified a set of common stages and that the techniques employed by each methodology depended heavily on the value-system to which it

is tailored. Since visualisation systems are interactive software systems, visualisation design is strongly influenced by the methods, techniques, and value-systems used in software engineering and HCI. Obviously methods, techniques, and a value-system specific to visualisation must also be considered. Thus the basic form of a visualisation specific methodology was established.

We propose an iterative methodology that consists of five main stages, *information gathering, analysis, design, implementation, and evaluation*. The analysis, design, and implementation stages are each supported by heuristics and patterns, which have been used as a means of providing the designer with experience-based knowledge and assisting them in making design decisions. In addition the patterns can be used to promote cross-discipline communication, something that is relevant to visualisation design teams because of the involvement of experts from several domains. Also, since visualisation is a form of HCI, each stage has an associated set of relevant HCI techniques.

To assist the visualisation designer we have compiled a collection of experience-based visualisation heuristics. We have also classified these heuristics by their effect on various usability factors. This classification allows the heuristics to be used as part of a heuristic evaluation, which in turn can be used to rank alternative visualisation designs.

To evaluate the methodology, the heuristics, the patterns, and in particular the heuristic evaluation, we have developed a small set of novel visualisations that support command and control teams. Initially these visualisations were tested with users in a series of cognitive walkthroughs. This helped to test the applicability of the visualisations to command and control and provided important user feedback. Two sets of visualisations were produced, a wow-vis set and a theoretically less usable just-a-vis set. The heuristic evaluation was then used to rank each visualisation, in each case predicting that the wow-vis would be more usable than the just-a-vis. These two sets of visualisations were then empirically tested.

The experiments used to evaluate the just-a-vis and wow-vis designs imposed a number of constraints. In particular because of the focus of this research on evaluating the heuristic evaluation the two versions of each design had to use the same data dimension to visual feature mappings. If this restriction had not been imposed it would not have been possible to determine if usability differences were due to perceptual differences or heuristic differences.

Another constraint imposed by the experiments that is of particular relevance was the need to use tasks that had specific answers i.e. non-judgmental. This is necessary since the experimenters must know whether or not the users are solving the tasks correctly. Unfortunately this is in conflict with the idea that visualisations are particularly useful in assisting users with tasks that require subjective interpretation of the data. Tasks that are well

defined, such as those used in the experiment, are often better solved using algorithms. In contrast to this, tasks that are difficult to code algorithmically i.e. those that rely on human judgement, experience and intuition e.g. pattern recognition, are much better suited to visualisation.

The observed results were then analysed and compared to the predictions made by the heuristic evaluation.

7.2 Conclusions

The primary goal of this research was to develop a formal methodology that covered the main aspects of the design process, identified the factors relevant to visualisation design, and assisted the designer in making design decisions by supporting each stage of the design process with the relevant methods and design knowledge. Following the methodology should produce visualisations that use established techniques and that have high usability levels. Having developed and evaluated a small set of visualisations for the command and control domain we have found evidence that supports this hypothesis. From this we can say that the methodology represents a step towards the complete integration of techniques from a number of disciplines, which is necessary to produce usable systems.

Evaluating a methodology such as the one proposed is a difficult task. First we must decide what criteria to use. For example, HCI specialists may evaluate an interface based on its usability, using criteria such as performance, rate of errors, subjective satisfaction, etc. With respect to a methodology, criteria may include development time, development costs, management, developer, and user satisfaction with the process, ease with which the methodology can be learned, the amount the methodology needed to be adapted to meet project requirements, scalability, and so on, as well as criteria for the final product such as usability, flexibility, maintainability, etc.

To evaluate the methodology it would then be necessary for a complete design team, including visualisation designers, software engineers, HCI specialists, and users, to implement and evaluate a set of visualisations and take the appropriate measurements during the development of each visualisation. Unfortunately due to the nature of this research this is not possible. An alternative approach may be to use case studies as a way of comparing methodologies and identifying the advantages and disadvantages of each. However, this is unlikely to take into account the extent to which the development teams were able to follow the methodology or their satisfaction with the overall development process.

The visualisations developed during this research have made extensive use of the patterns and heuristics identified as supporting each stage of the methodology. Since there is evidence that

those visualisations that make use of this design knowledge are more usable than those that do not, we feel that the use of such patterns and heuristics, and in general the knowledge upon which they are based, is valid. However, writing patterns is difficult and the visualisation patterns do need to be reviewed by other researchers. In addition, due to the nature of this research, the use of these patterns as an aid to communication between different design team members has not been tested.

The use of a formal structure and the collection of the heuristics into a single catalogue has proved extremely beneficial during the design process. The catalogue acts as a useful tool, both as a memory aid and as a way of finding related heuristics. Also, by summarising the ideas of several researchers and bringing them together into a single source, the designer no longer has to search through multiple references to find the information they require. This helps to reduce development time. However, the catalogue is far from complete and it is hoped that future researchers can continue to add to this resource.

Another aim of this research was to provide designers with a mechanism for evaluating alternative designers. To this end three different design evaluation techniques were proposed, a mapping evaluation technique, a heuristic evaluation, and a metric evaluation.

The mapping evaluation, which is similar to the evaluation strategies used in several automatic visualisation generators, has not been used convincingly in the evaluation of this research. The problem in this case lies in the fact that the wow-vis and just-a-vis versions are too similar in terms of the data dimension to visual feature mappings used. This similarity results in almost identical ratings for each visualisation. Therefore it cannot be confidently said that one visualisation is perceptually more effective than the other. Despite this we still feel that under different circumstances the mapping evaluation approach is a valid strategy for rating visualisations and has been used successfully in a number of automatic visualisation generators.

The mapping evaluation is also limited by the fact that the encoding methods are ranked independent of their use within the visualisation. The metric evaluation technique attempts to overcome this limitation by assigning a dimensional score to each encoding method based on the cognitive workload required to decode the mapping. We feel that this model is too simplistic because it does not take into account a number of confounding factors such as interaction, the point at which information is displayed, visualisations consisting of multiple views, and so on. By themselves none of the metrics indicate which visualisation to choose, however when used together the result is clearer.

The reasons why one visualisation is more usable than another are difficult to define. Why is something easier to use or easier to learn? Why does one approach promote better

performance and another greater accuracy? Visualisations rely on perceptual models, cognitive models, and interaction, and can therefore be regarded as complex systems. Looking at the perceptual effectiveness of a visualisation as with the mapping evaluation is simply not enough. Performing a metric evaluation can be useful but again the complexity of the visualisation may not have been captured and so looking only at the low-level components of a visualisation may be inadequate. Heuristics, of the type presented in appendix A, take a different approach.

Essentially each heuristic is a combination of rules, none of which may be specifically defined or understood, but which incorporate knowledge from perception, cognitive modelling, and interaction, and, when brought together as a heuristic, help guide the designer to develop an effective visualisation. The heuristics encapsulate years of design experience that have proved useful in overcoming some of the complexity issues previously mentioned. We feel this makes them a suitable compliment to the other design evaluation techniques proposed.

The heuristic evaluation relies on the classification of the heuristics by their effect on various usability factors. To achieve this we have simplified each factor as far as possible and assigned a set of scores to each heuristic based on their gross effect on each factor. Using this simple model the heuristic evaluation predicted that each wow-vis would be more usable than the equivalent just-a-vis. Unfortunately, in the majority of cases the empirical evidence could not conclusively establish which version was more usable. A number of reasons for this are presented in sections 6.4 and 6.5. Despite this we still feel that with additional tuning the heuristic evaluation is a viable technique for evaluating visualisation designs in terms of their usability and the model upon which it is based is a reasonable first attempt.

Several novel visualisations have been developed. Despite being restricted to the requirements of this research, evidence from the cognitive walkthroughs suggests that visualisation is an appropriate tool for command and control.

As stated the evaluations have provided practical evidence that the visualisations developed are applicable to command and control. Moreover they indicate that those visualisations that implement the design knowledge embedded in the patterns and heuristics are indeed more usable than those that do not. This helps to support the validity of the patterns, heuristics and the methodology as a whole.

7.3 *Future Work*

7.3.1 *Methodology*

The methodology is a first attempt at integrating methods and techniques from several disciplines in a way similar to the integration of software and HCI methodologies. However, the research has focused on visualisation, in particular the collection and development of several visualisation specific techniques. This has resulted in certain aspects of the methodology having more depth than others. Specifically, although elements from the usability engineering life cycle, and HCI techniques in general, have been assigned to each stage of the methodology, details regarding which specific techniques to use are ambiguous. This is partly due to the focus on factors relevant to visualisation and partly due to the way in which research of this kind is conducted i.e. empirical restrictions, time and resource limitations, etc.

In the short term both the ‘information gathering’ and ‘evaluation’ stages can be expanded. Information gathering should include more details regarding how raw data can be organised, data indexing strategies, techniques for identifying erroneous data, etc., and the evaluation stage should assist the designer in the selection and execution of HCI techniques based on their goals. The methodology also fails to address project management criteria such as cost constraints, time limitations, resource management, etc.

It is felt that in the long term it may be possible to develop a *grand unified methodology* which as well as combining methods and techniques from HCI, software engineering, visualisation, perceptual psychology, computer graphics, project management, etc., also describes in detail which techniques to use, when to use them, and how to use them, as well as the roles that various development team members play. It is possible that patterns are a mechanism that can be used to present this information and so achieve this goal.

The methodology needs to be more rigorously implemented in a more realistic situation. So far only a single individual has applied the methodology in a limited way. This has been necessary due to restrictions imposed by the empirical evaluation and the nature of this style of research. Ideally a visualisation design team consisting of experts from disciplines such as human computer interaction, visualisation, and software engineering, as well as users, should be brought together. The tasks to be solved could then be strictly defined and the methodology used to produce visualisations that solve those tasks without the need to conform to the requirements of a research style experiment. Verifying the methodology in this way would help to prove not only its validity but also the validity of the heuristics and patterns that it uses. This would also help determine the applicability of the patterns as a means of cross-discipline communication.

7.3.2 *Patterns*

At present the ‘information gathering’ and ‘evaluation’ stages do not make use of the patterns concept. We feel that there must be common solutions to recurring problems in both of these stages. In particular, techniques exist which handle missing or erroneous data, and evaluation procedures often take the same basic form. Discovering and applying these patterns would help to form a more complete methodology.

More visualisation patterns need to be discovered. As with finding additional heuristics, one approach is to look at existing visualisation systems and try to determine the solutions that they have in common. This *pattern mining* is a difficult task since it is not always clear if something is a pattern or not. In addition to finding patterns that describe good solutions, visualisation *anti-patterns* should be searched for and recorded in a similar way. This will help to prevent designers from using inappropriate design techniques.

Further verification of the proposed patterns is also required. Exposure of the patterns to other researchers and application of the patterns either individually or as part of the methodology should help to achieve this.

7.3.3 *Design Evaluation Techniques*

The mapping and metric evaluation techniques are both limited by their failure to adequately take into account the complexity of visualisations. However, it is possible that some combination of these two approaches could result in an evaluation technique based on low-level features of the visualisation that compliments the more high-level approach taken by the heuristic evaluation. One way of achieving this may be to weight the perceptual rankings by the dimensional score.

Due to the constraints imposed by the experiment and the emphasis of this research on verifying the heuristic evaluation, both the mapping and metric evaluation techniques have not been thoroughly tested. We hypothesise that for equivalent visualisations the mapping and metric evaluation techniques may be better suited to those visualisations with very different mappings, and the heuristic evaluation for those visualisations that use different visualisation techniques. This is something that needs to be investigated in future research.

There is some evidence that the heuristic evaluation works, but as with other aspects of this research further application of this technique is required to verify its validity. The model upon which the evaluation is based and the scores assigned to each heuristic have deliberately been kept simple. Although as a first attempt this is valid, it may be useful to try to refine both the model and the scores so as to improve the accuracy of the evaluation. This could be achieved

by breaking down the usability factors into more atomic units, adding more heuristics, or increasing the granularity of the scoring system. Some of the heuristics we have used are in fact compound heuristics. For example, the heuristic *multiple linked views* can actually be broken down into a number of distinct heuristics. It is likely that assigning scores to these sub-heuristics would improve the accuracy of the evaluation. In addition these sub-heuristics should be included as part of the heuristic catalogue.

So far the heuristics have only been classified by their effect on a small number of usability factors. It is likely that there are other factors, e.g. articulatory load, which can also be used to classify the heuristics. This would also help to form a more complete model and is more likely to satisfy the aims of the designer.

The heuristic evaluation relies heavily on the usability scores assigned to each heuristic. At present these scores are based primarily on the literature reviewed and discussions with other researchers. It is possible that these scores could be verified by performing a series of experiments in which only one heuristic is altered at a time and the effect of that alteration on the usability of the system recorded. However, this would be a laborious task and it is not clear if all the heuristics could be tested in this way.

At present the heuristic evaluation can only be used to judge qualitatively if one visualisation is more usable than another. A large difference in scores indicates a large difference in the comparative usability of each visualisation. It is clear from the evaluation that it would be useful to calibrate score differences. So for example, two visualisations with a difference in score of less than ten may be considered equal in terms of usability, a difference of less than twenty may mean that one visualisation is only slightly more usable than the other, and so on. Further experience applying the heuristics and additional empirical evaluations would help to achieve this calibration.

It is also apparent that some heuristics have a greater influence on the usability of a visualisation than others, although this may be task dependent. For example, a find task is often best supported by the heuristic 'filter' rather than 'use colour carefully'. Therefore, it would be useful to weight the heuristics accordingly. In the long term this could be achieved by conducting further experiments. However, in the short term it may be possible to use the number of supporting researchers as a means of indicating the relative importance of each heuristic, i.e. heuristics that are proposed by many researchers would be given a higher weighting.

Some of the visualisations developed could be classed as compound visualisations, i.e. visualisations that are made up of multiple views. However the strict definition of multiple linked views has not been implemented as part of this research. It is not clear how the design

evaluation techniques cope with these types of visualisations. It may be necessary to evaluate each view separately and then perform a weighted combination of the results (e.g. based on the time spent interacting with each view), to obtain a final rating. Since multi-view visualisations are becoming increasingly common this is an important area of research.

7.3.4 *Visualisation*

Visualisations rely heavily on graphical transforms, i.e. changes to the appearance of objects in a graphical scene, however existing graphical transforms tend to change one visual feature only, e.g. colour, size, etc. We feel that the development of new graphical transform techniques that alter more than one visual feature at a time may be beneficial in making relevant task specific visual elements more salient. Similarly analysis of the effects of using less common graphical transforms e.g. motion would also be useful.

A number of researchers advocate the use of traditional chart-based visualisations and as such have even developed systems that can automatically generate chart-based displays based on the data and the tasks to be supported. We have argued that these automatic visualisation generators have a number of limitations. However it is clear that they do have a place in visualisation design. The question remains as to when a table-based display is more appropriate than a chart-based display, is more appropriate than some other novel visual representation. To try to answer this question we are already considering a second experiment that compares the visualisations developed in this research to chart-based displays within the command and control domain.

The Mission Dependency network visualisation tries to assist the user in the formulation of alternative mission plans when some unforeseen event prevents the proposed plan from proceeding. This is a complex task that is well suited to the type of problems that visualisations are good at helping users solve. Unfortunately the Mission Dependency visualisation did not fit well within the framework of the empirical evaluations. Effective network visualisations are inherently difficult to design so it is important to conduct research into alternatives to network visualisations and the development of network visualisation specific techniques. Continuing work on the Mission Dependency visualisation is particularly important for command and control.

The Organix visualisation is perhaps the most generic visualisation of those developed. Although Organix suffers severely from occlusion and is limited by the number of items that can be successfully compared at any one time, we feel that it is a truly novel visualisation that has significant potential. Future work related to Organix includes improving the growth algorithms, using different growth models, combining it with other visualisations, providing

greater interaction and drill down mechanisms, applying functions to Organix e.g. visual difference, and so on.

Current visualisations tend to be designed for a specific task with a specific type of user in mind, and once implemented are fairly inflexible. Research needs to be undertaken into the development of *adaptive visualisations* that can intelligently detect the skill level of the user and what they are trying to achieve and then present the required information in a suitable format. In other words adapting to the abilities and requirements of the current user and task. It is hoped that such a system would be more usable, improve subjective satisfaction and increase work output. However it is likely that this would require several years research.

Empirically evaluating visualisations with users is often difficult to organise and is costly in terms of time, money, and human resources. It would be useful to develop a comprehensive cognitive model of the user that could then be implemented and used as a way of cheaply testing the usability, robustness, and appropriateness of visualisation designs. This is an extremely difficult task that would require expertise not only from visualisation but also artificial intelligence, HCI, psychology, etc., etc. However, the benefits from such a model and the lessons learnt during its development would be tremendous.

Visualisation is still a relatively new research area and there is definitely scope for the development of novel visualisations and generic visualisation techniques. With ever-increasing amounts of complex data being collected and analysed, visualisation research is vital.

Appendix A: HEURISTICS

The following is a short description of each heuristic, what it means and its effect when used in a visualisation.

1.0 Use a real world physics model

We feel more ‘comfortable’ with representations that match our cognitive model of the real world. Our daily experiences lead us to have preconceived ideas about phenomena such as lighting, shadows, surfaces, etc., violating these ideas means the representation does not seem ‘natural’.

Specifically:

- *Lighting*. Should always be from above (Rheingans and Landreth (1995) cite Ramachandran (1988)).
- *Shadows*. Help to identify an objects spatial location, aid comparison between objects and make it easier to determine an objects structure.
- *Hidden line and surface removal*. Using wire frame models or other techniques that allow the far side of an object to be seen as well as the near side confuses the user and makes it difficult to judge the spatial location and depth of objects within the scene. This is especially true when objects overlap.

Author(s): Rheingans and Landreth (1995)

See also: 2.0, 3.0, 4.0

2.0 Visually refer all graphical objects to a reference context

A reference context is a mechanism that aids the user in spatially locating and comparing visual objects. This can be achieved by using grids, planes or aligning points in a particular dimension e.g. time. For example, anchor lines can be used to attach objects to planes or shadows can be used to help place objects in a scene. Using a reference context is particularly important when using three-dimensional representations.

Author(s): Brath (1999)

See also: 1.0

3.0 Use connotative mappings

Connotative mappings help to reduce the cognitive effort required in remembering the mapping between data attribute and visual dimension. For example (from Brath (1999)):

- “size” and “amount” map connotatively to scale (e.g. length, width, height)
- “price” maps connotatively to a vertical axis supporting concepts such as “high price” and “low price”
- “time” maps connotatively to motion and animation; as well as to a left-right orientation (dependent on culture), where left is earlier and right is later.
- “good” and “bad” may map connotatively to colours (dependent on culture). Typically, red may be used for bad and green for good.

Author(s): Brath (1999)

See also: 1.0, 4.0

4.0 Use an organisational device the user already knows

An organisational device is a pre-defined representational structure that can be applied to the data. Chart based representations e.g. bar charts, scatter plots, etc., are good examples of structured representations that many users are familiar with from early education.

Organisational devices can also be derived from the procedures and environment in which the user already works, e.g. floor plans, specialised financial charts, etc. Other organisational devices include maps, grids, hierarchies, time series, rooms, etc.

Using an organisational device the user is familiar with reduces the learning time required and may be more readily accepted by the user if they are happy with their current working practices.

Author(s): Brath (1999)

See also: 3.0, 12.0

5.0 Use redundancy to aid discrimination and comprehension

Widdel and Post (1992) define *partial redundancy* as mapping one data attribute to one visual feature and *full redundancy* as mapping one data attribute to several visual features. For instance mapping friend or foe status to either a green or red triangle is an example of partial redundancy whereas mapping friendly troops to green triangles and enemy troops to red squares is an example of full redundancy. Using full redundancy aids visual search tasks

because there are more features to identify. Full redundancy is normally referred to simply as *redundant encoding*, and is the most commonly accepted definition.

Redundant encoding can be used to visually reinforce the similarities and differences between visual objects. It also has the advantage of overcoming any visual deficiencies the user may have. For example, by redundantly encoding shape with colour, users who are colour blind can still use the shape of the object as the identifying feature. A common example of redundant encoding is to combine colour and height e.g. ThemeScapes (Wise et al. 1995). Another advantage of redundant encoding is that an object's context can be maintained even if a visual feature is altered. This is useful when making relevant data salient.

Brath's (1999) definition of redundancy also includes any extra visual objects that are not directly related to data values. The reference context described in heuristic 2.0 is an example of a redundant graphical object that helps the user comprehend the visualisation and compare objects. This definition of redundancy is similar to the *control level of detail* heuristic proposed by Rheingans and Landreth (1995).

Another form of redundant encoding is to use several identical graphical objects each of which displays a single attribute of the data but the encoding of that attribute is different for each object. For example, one object may use a grey scale, another a HSV scale, another an RGB scale, and so on. Rheingans and Landreth (1995) call this *explicit redundancy*.

Author(s): Brath (1999), Rheingans and Landreth (1995)

See also: 2.0, 6.0

6.0 Use different visual dimensions differently

Different visual dimensions can be perceived with different degrees of accuracy. Mackinlay (1986) used this fact to map and rank visual features to data types. This ranking has since been used in numerous visualisations. However, other data characteristics e.g. value ranges, structure, etc., may have an effect on this ranking as do connotative mappings and organisational devices. Unfortunately multiple mappings can cause visual interference, which will increase visual search time and may even cause illusions.

An important question arises from this heuristic. Is there a limit on the number of data attribute to visual feature mappings that the user can remember? Obviously factors such as connotative mappings, cultural influences, etc., have an effect, but in general is the limit the same as the number of items that can be held in short term memory? These questions may need to be addressed in future research.

Author(s): Brath (1999)

See also: 5.0, 7.0, 8.0

7.0 Minimise Illusions

Illusions confuse the user which leads to increased cognitive load and reduced accuracy when determining values and discriminating between objects. The most common illusion effects occur when combining colour and size. Rheingans and Landreth (1995) summarise these effects as follows:

- The perceived size of an object can be influenced by its colour.
- The perceived hue of a colour can be influenced by its saturation.
- The perceived saturation of a colour can be influenced by its hue.
- The perceived depth of an object may be influenced by its colour.
- The perceived colour of an object may be influenced by the colour of surrounding objects.

Other illusion effects to avoid as listed by Brath (1999) include:

- *Moiré*. An effect caused by a large number of parallel lines e.g. dense grids.
- *Alignment and overlap*. In 3D visualisations occlusion is a major problem that may cause several objects to be hidden behind other objects. Using transparency or emphasising the outline of an object can help to overcome these effects.
- *Break-up*. Long thin polygons tend to break-up when viewed from a distance, drawing a border around polygons of this type can help.

Author(s): Brath (1999), Rheingans and Landreth (1995)

See also: 5.0, 6.0, 8.0

8.0 Use colour carefully

Colour is a complex visual dimension to use. A complete description of when, where, and how to use colour in visualisations is a difficult task that requires further research.

A small set of guidelines includes:

- The number of colours that can be used to represent discrete data is debatable, typical figures range from 4-7 to 10-12 discrete colours.
- Avoid using blue especially for small objects or with a black background.
- Only represent one or two continuous data dimensions with colour.

- With the exception of nominal data, colour brightness is perceptually more dominant than colour hue.
- Avoid using very low brightness values when both hue and brightness are used to represent data dimensions. Low brightness makes it difficult to differentiate hue values.
- Colour can be effectively used to highlight information e.g. making salient objects red.
- Beware of different cultural interpretations of colour. For example, in Western cultures red means stop or danger, and green means go, but in Eastern cultures the opposite is true.
- Consider the needs of colour-blind users. Red-green colour blindness is common therefore these colours may need to be avoided.
- Use colour consistently.
- Ensure that the colour coding supports the task.

Author(s): Brath (1999), Shneiderman (1996,1998)

See also: 7.0

9.0 Use smooth animation and motion especially for temporal data

Animation or motion is an effective means of representing data that has some temporal aspect. The most important consideration when using this technique is to make the transition from one state to another as smooth as possible. Animation or motion that produces a large change of state distracts the user from the information and may cause important features to be missed. Large jumps from one location in a 3D environment to another may cause the user to become disoriented a smooth transition helps them to maintain their context in the environment. If possible the user should be able to control the rate of change and direction of the animation, this may allow them to rapidly and accurately explore the data.

There are arguments for not using smooth animation or motion. If the task is to monitor data that may change only gradually over long periods of time, then using large discrete time steps may help the user to identify significant changes in the data.

This implies that this heuristic may be task dependent.

Author(s): Eick (1995), Foley and Van Dam (1995)

See also: 15.0, 25.0

10.0 Visualisation is not always the best solution

One definition of a visualisation is a representation that allows tasks with low specificity to be performed on large quantities of data. The requirement that tasks be difficult to define is perhaps the most important. If a task has a high specificity e.g. find the oldest file on a PC,

then it is much more efficient to write an algorithm to accomplish this task than it is to develop an effective visualisation. This reasoning is used extensively in Chuah's (2000) AVID system. However, if the task is to find a file that has no unique distinguishing characteristics other than its partially remembered location and file type, then a visualisation is more appropriate.

The second requirement, that a visualisation should only be used for large quantities of data is debatable. Is a bar chart a visualisation? How many data points are required for a representation to be called a visualisation? Can ten items in a list or menu be called a visualisation? The requirement itself is less important, when to use the appropriate technique, an algorithm or a visualisation, is what matters.

Author(s): Carr (1999)

See also: 11.0

11.0 Don't use 3D if the number of data points is low

In our everyday lives we are used to working with two-dimensional mediums such as pen and paper or keyboard and screen. This suggests that users would have a pre-familiarity with a 2D visualisation making them more acceptable. Two-dimensional visualisations are computationally less expensive in terms of computer processing although this is becoming less of a factor as graphics hardware improves.

There also arguments for 3D visualisations. We live in a 3D environment, we understand depth perception and how it effects what we see. Using the third dimension allows a visualisation to access an almost infinite amount of space, the more space, the more data can be displayed. This ability to display more data corresponds to Foley's (1995) heuristic for increasing information content. However, Brath (1999) states that 3D visualisations should not be used when there are a low number of data points.

Using the third dimension also has many disadvantages. Several of these disadvantages are to do with readability issues, the worst of which is the problem of occlusion where objects partially or completely obscure other objects. There are also several problems associated with navigation in 3D space (see heuristic 15.0).

From this brief discussion it is evident that additional heuristics are required that state under what circumstances 2D or 3D visualisations should be used.

Author(s): Brath (1999), Carr (1999), Foley and Van Dam (1995)

See also: 10.0, 15.0, 19.0

12.0 Map the data to an appropriate visual object

There are a number of data factors that can influence the design of visual objects to represent data items. These factors include the data *source*, *structure*, *type*, *range*, *dimensionality*, and *quantity*.

For scientific visualisations the data source can often be used as a model for the visualisation. For example, MRI data can often be presented using a model of the organ from which the data was gathered. Unfortunately this is not always possible, especially for information visualisation where the data source tends to be less concrete.

The structure of the data can also be used as a basis for the visualisation. For example hierarchical data is often presented using a tree structure. These ‘natural’ structures can help to reinforce the users mental model of the data.

When mapping data attributes to visual features the data type and range are both significant. Mackinlay (1986) and Salisbury (2001) have both ranked data types to visual features in terms of how efficiently the visual feature can encode the data type. The data range can also affect the choice of visual feature since some visual features are limited in the number of discrete values that they can represent.

The task to be solved is also important. For example, if the relative locations of various cities is not important then there would be no reason to display the cities on a map.

Author(s): Carr (1999)

See also: 3.0, 4.0, 23.0

13.0 Test your designs with users

Before prototypes are developed mock-ups should be shown to the users as a cheap and effective way of making sure that the tasks and requirements have been correctly understood. Prototypes can then be developed. Prototypes should then be tested with users and repeatedly refined as required. This approach will help to avoid costly changes late in the development stage.

Author(s): Carr (1999), Graham (2000), Wiss et al. (1998)

See also:

14.0 Use datatips for identification, education and validation

Datatips is an interaction technique where selecting an object reveals detailed information about that object in a way similar to tooltips. This mechanism can also be used as an effective ‘drill down’ technique, especially as the context of the selected item is maintained. Datatips allow the user to quickly identify an item and so confirm and strengthen their cognitive model of how the visualisation should be interpreted.

Author(s): Brath (1999)

See also: 28.0

15.0 Provide a simple 3D navigational model

Navigating in three dimensions using the standard mouse input device can be difficult. If complete freedom of movement is allowed then users can quickly and easily become disorientated e.g. rotating the view so that it appears upside down, this should be avoided. The model should also stop users from seeing empty space, in other words users should not be allowed to move into a position in which they can no longer see the data. Finally, the method of movement should be smooth and simple to use, aircraft style controls may not be appropriate.

Author(s): Brath (1999)

See also: 9.0, 11.0

16.0 Use small multiples to encode multiple data attributes into graphical objects

A small multiple or glyph is a graphical object that represents multiple data attributes. Typically many small multiples are used together to show variations in the attributes they present. A common example is to use arrows placed on a weather map, the arrow length shows wind speed and arrow direction shows wind direction. Using small multiples can help to increase the task expressiveness of a visualisation i.e. increase the number of tasks that can be solved. They are also useful for finding relations in the data. However, small multiples can be difficult to design and care must be taken to limit the number of mappings used within each small multiple. The more data attributes each small multiple represents, the more cognitive effort required by the user in remembering the mappings from data attribute to visual feature. Complex small multiples may also require more space. If screen space is at a premium and a large number of items need to be displayed, occlusion may be a problem, in this case multiple views may be a better solution.

Author(s): Brath (1999)

See also: 31.0

17.0 Use legends, scale and annotation to aid memory and increase accuracy

Legends and annotations help to reduce some of the cognitive load on the user. Datatips is a good annotation technique that only requires attentive viewing when the user requires it and only temporarily increases clutter. Legends tend to be on screen all the time and display fixed information. However, if screen space is at a premium it should be possible to hide the legend. Legends and annotations also help the user learn how to interpret the display and are also useful when interaction with the visualisation is not possible e.g. a printed screen capture.

Referring to legends may incur eye-shift, which can result in the user losing the items of interest. However with increased exposure to the visualisation the need to refer to the legend should be reduced.

Labels in close proximity to objects also helps to increase identification accuracy but at the potential cost of more attentive viewing and an increase in visual clutter.

Author(s): Brath (1999)

See also: 14.0, 18.0

18.0 Do not rely on interaction

Visualisation results or screen captures are often printed out and distributed to relevant individuals. This ‘paper copy’ of the screen does not allow any interaction. This limitation means that visualisations need to be as comprehensible as possible without relying on interaction to overcome issues such as readability or allowing access to further information.

Author(s): Brath (1999)

See also: 17.0, 20.0

19.0 Occlusion is undesirable

Occlusion makes it difficult for the user to read information or discriminate between different visual objects. Often a certain amount of occlusion cannot be avoided however it must be kept to a minimum. There are a limited number of techniques that can help to overcome occlusion. Transparency can be used to help show the relative locations of objects although transparency can make object boundaries less distinct. Navigation in the scene can also be used as can standard filtering techniques to reduce the number of displayed objects.

Author(s): Brath (1999)

See also: 15.0

20.0 Use interaction to explore large data sets

Standard interaction techniques such as zooming, filtering, selection, navigation, etc., all help the user improve their mental map of the data, help them find specific items and help determine relationships between items. Interaction also overcomes the limitations of available screen space thus increasing the amount of data that can be explored. The importance of interaction requires that it be easy and intuitive to use.

Author(s): Brath (1999)

See also: 15.0, 18.0

21.0 Let users control visual bindings

Supporting this guideline may enable the users to increase their subjective satisfaction of the system. If applied to the user interface rather than the visualisation contents, this guideline may be useful. However, if the user is allowed to choose the mappings between data attributes and visual dimensions, it may be possible to alter the visualisation in such a way that it is difficult to comprehend or leads to misinterpretation of the data. This facility places great responsibility on the user and should only be available to expert users and then with extreme care.

Author(s): Foley and Van Dam (1995)

See also:

22.0 Emphasise the interesting

Having performed a task analysis and determined the tasks it is necessary to make sure that the visualisation highlights the data necessary to solve those tasks. Emphasising the data of interest occurs at two levels. Firstly the typical scene displayed to the user should show all the relevant information. For example, if a user is interested in the population of cities but not their location then using a map of the relevant country as part of the visualisation is not necessary. Secondly, data should be emphasised as a result of user interaction e.g. filters, search results, etc. Typically this emphasis will result in a change in the relevant visual objects features. For example, in response to a query all matching items may change colour, size, etc.

Author(s): Rheingans and Landreth (1995)

See also: 12.0, 23.0, 26.0

23.0 Task specific

The visualisation must support the tasks that the user is trying to solve. Macklin and Dudfield (2001) verify this requirement with respect to command and control visualisations. The data requirements, the task category, specificity, flexibility and frequency, all need to be considered. A visualisation should present only the appropriate data necessary to solve the tasks. Redundant data increases the complexity of the design and forces the user to try to filter out irrelevant information. This filtering process will lead to an increase in articulatory, perceptual and cognitive processing. A visualisation that meets these requirements is said to be *task expressive*.

Author(s): Brath (1999), Carr (1999), Casner (1991), Chuah (2000), Eick (1995)

See also:

24.0 Overview

The visualisation should provide an overview of the entire data set under consideration. Eick (1995) refers to the overview technique as a *reduced representation*. Since the overview tends to display a higher number of data items than any more detailed view it is often necessary to use simple glyphs that minimise clutter, maximise use of screen space and portrait the data attributes most relevant to the task.

An overview provides several functions, it reduces the visual space the user has to search, it shows the user their current location and therefore context within the data set, and it acts as a navigation aid by helping them to decide which area of the data space to explore next.

Author(s): Carr (1999), Eick (1995), Foley and Van Dam (1995), Shneiderman (1996)

See also: 25.0, 31.0

25.0 Zoom

A zoom facility allows the user to see a more detailed view of a subset of the data. There are two types of zoom that can be performed. A *spatial zoom* produces an enlarged version of part of the dataset, whereas a *semantic zoom* leaves the content the same but generates a view with a different appearance to that of the original. The zoom process should be smooth so that the user does not lose their overall location or context. It is important to emphasise that this new zoomed view may still display a large amount of data and therefore should not show fine

grained details about each item displayed. In other words, zooming is not the same as showing details on demand.

Author(s): Carr (1999), Foley and Van Dam (1995), Shneiderman (1996)

See also: 26.0, 27.0

26.0 Filter

A visualisation should provide filtering facilities. Filtering helps to reduce the number of data items displayed to those of specific interest to the user and their current task. This helps to reduce the visual complexity of the display. Typically widgets such as sliders, buttons, menus, etc., are attached to different attributes of the data. Manipulating these controls to specify the desired attribute values should cause a rapid update of the main display. This technique is known as *dynamic queries*. The choice of which attributes to allow the user to filter by may be determined from the task analysis stage.

One potential problem with filtering is that because items are removed from the display, the context of the remaining items may be lost. One solution is to change a visual feature of the surviving items e.g. colour, rather than removing those that don't pass the filter. In effect making the relevant items more salient. This means that the surviving items context is preserved but the visual complexity of the display stays high. In addition altering a visual feature in such a way may alter or lose the meaning of the associated data value.

Author(s): Carr (1999), Eick (1995), Foley and Van Dam (1995), Shneiderman (1996)

See also: 22.0, 32.0

27.0 Details on demand

A visualisation should allow the user to view data item attribute values. The datatips can be used to show a temporary details on demand window. The advantages of the datatips technique are that no mouse clicks are required, just pointing, and since the window is temporary it only interferes with the display for a short time. Both Brath (1999) and Eick (1995) recommend this approach.

An alternative is to display the details in a completely separate window. This more permanent window is useful if the details need to be frequently referred to but at the cost of increasing screen clutter. So far, a mechanism for converting the popup window into a separate permanent window has not been described but this may be a useful feature.

Author(s): Brath (1999), Carr (1999), Eick (1995), Foley and Van Dam (1995), Shneiderman (1996)

See also: 28.0, 31.0

28.0 Relate

Finding relations between items is a common user task. When a user selects an item it should be possible for them to see related items. Related items do not necessarily have to contain the same type of data. For example, if a user selects a particular city, they may wish to find cities with similar populations or they may wish to see various tourist reviews about the selected city. Relate is similar in some ways to linked-brushing, however relate is not limited to simply highlighting similar items. Relate can allow more complex relationships to be viewed that may require different visualisations. Shneiderman (1996) comments on the difficulty of determining the relationships required and the interface actions that will produce them.

Author(s): Carr (1999), Shneiderman (1996)

See also: 14.0, 32.0

29.0 History

Visualisations deal with data exploration. While exploring data users will often go down the 'wrong path' or wish to revisit a particular point during their exploration and take a different path. Having facilities similar to those in word processing applications like undo and redo would aid the user in the exploration process.

Author(s): Carr (1999), Shneiderman (1996)

See also: 30.0

30.0 Extract

Users should be able to save the results of their explorations. In addition it may be useful to save the control widget values that lead to the results at each stage of exploration. This facility would allow the saved items to be used in other visualisations for further analysis or be printed out or emailed to others who may be interested.

Author(s): Carr (1999), Shneiderman (1996)

See also: 31.0

31.0 *Multiple linked (co-ordinated) views*

Multiple co-ordinated views of the data help to provide insight into the relationships between various data items. If necessary the user can also focus on one view at a time to solve view/task specific problems or use them in a combined way to solve more complex problems. Performing actions in one view should propagate to other views. For example, clicking on a directory in a tree style file browser may cause details of the individual files within that directory to be listed in a table view, clicking on a file in the table view may then cause the tree view to scroll to the relevant directory. A more typical example is dragging a rectangle to a new location in an overview window that then updates the corresponding detailed view.

Successfully implementing multiple co-ordinated views is a difficult task. The layout management, co-ordination methods and situations in which they should and should not be used all need to be considered. Layout management is covered in depth by North and Shneiderman (1999) as are co-ordination issues. Baldonado et al. (2000) provides a detailed set of heuristics on the use of multiple co-ordinated views.

Author(s): Carr (1999), Eick (1995), Foley and Van Dam (1995), North and Shneiderman (1999), Baldonado et al. (2000)

See also: 14.0, 28.0

32.0 *Direct manipulation*

The technique of interacting directly with objects in the scene rather than via menus or other controls is known as *direct manipulation*. Dynamic queries are an excellent mechanism for providing rapid, easily reversible actions on the scene. However, sometimes it is difficult or less intuitive to map an action onto a control. Instead the action should be applied directly to the object or objects in the scene. Selecting objects by using a bounding box is a good example. The classification that the user has determined by looking at the objects may be difficult to describe in terms of the values of those object's attributes and therefore difficult to set the associated controls correctly. However, by using a bounding box or by clicking on individual items, the user can easily select the desired set.

Chuah et al. (1995) have developed a comprehensive set of interactive techniques called *selective dynamic manipulation* (SDM). In the SDM system objects or sets of objects can be selected by clicking on them or using sliders, this alters their appearance e.g. a change in colour, to allow easy identification. Sets of selected objects can be saved, scaled, translated or manipulated in several other ways to help analyse the data.

Interestingly Chuah et al. (1995) have taken this idea a step further by attaching handles to objects. By pushing and pulling the handles the appearance of the selected object, and any other objects in the selected set, changes dynamically. This allows the user to make the selected set of objects more prominent.

The ability to manipulate the visualisation using the SDM approach can help to overcome some of the problems associated with 3D views, for example translating the selected objects can overcome occlusion.

Author(s): Chuah et al. (1995), Eick (1995)

See also: 9.0, 15.0, 26.0, 27.0

Appendix B: HEURISTIC CLASSIFICATION

Classifying the heuristics by various criteria may help the designer to determine which heuristics are appropriate and when to use them, this may then lead to a more effective visualisation in terms of those criteria. We have attempted to classify the heuristics by their effect of various usability factors.

B.1 Heuristic effect on Usability Factors

The main factors that affect usability according to Shneiderman (1998, p15) are as follows:

- *Time to learn.* How long does it take for typical members of the user community to learn how to use the commands relevant to a set of tasks?
- *Speed of performance.* How long does it take to carry out the benchmark tasks?
- *Rate of errors by users.* How many and what kinds of errors do people make in carrying out the benchmark tasks? Although time to make and correct errors might be incorporated into the speed of performance, error handling is such a critical component of system usage that it deserves extensive study.
- *Retention over time.* How well do users maintain their knowledge after an hour, a day, or a week? Retention may be linked closely to time to learn, and frequency of use plays an important role.
- *Subjective satisfaction.* How much did users like using various aspects of the system? The answer can be ascertained by interview or by written surveys that include satisfaction scales and space for free-form comments.

Usability refers to the effectiveness of a software product in allowing a user to accomplish a task. It is important to note that to classify the heuristics these definitions refer to the visualisation **not** the data contained within the visualisation. For example, the time to learn refers to the time it takes the user to learn how to use the visualisation, not how long it takes them to interpret what the data means.

In order to determine the effect of each heuristic on each usability factor it is necessary to simplify the factor as much as possible. For example, learnability consists of a number of dimensions including the time to learn how to use the system, the ease with which tasks can be carried out, the experience level of the user, the levels of functionality of the system, the frequency of use, and so on. To analyse each factor in detail and to determine the effect of each heuristic on each dimension of each factor is beyond the scope of this research. It is also unnecessary; a simplification of each factor should allow the relative merits of each heuristic to be determined. In addition, it is often not possible to perform a completely accurate analysis simply because we do not have enough knowledge about how people learn and use systems. For example it is not known exactly what factors make something easy or difficult to learn. Therefore simplifying the usability factors is a valid solution.

Appendix B: HEURISTIC CLASSIFICATION

Instead of trying to quantitatively determine the effect of each heuristic a qualitative approach has been used. This fits well with the idea of using a simplified model. It should be noted that these assignments are not definitive, as stated, each factor is extremely complex, which together with the fact that heuristics may influence each other means that no assignment can be said to always be correct.

		Learn	Perf.	Errors	Retnt.	Sb.St
1	Use a real world physics model	+	+	+	+	?
2	Visually refer all graphical objects to a reference context	0	+	+	0	+
3	Use connotative mappings	+	+	+	+	+
4	Use an organisational device the user already knows	+	+	+	+	+
5	Use redundancy to aid discrimination and comprehension	0	+	+	+	?
6	Use different visual dimensions differently	-	+ & -	+ & -	0	?
7	Minimise Illusions	0	+	+	0	?
8	Use colour carefully	+	+ & -	+	0	+
9	Use smooth animation and motion especially for temporal data	0	0	+	0	+ & -
10	Visualisation is not always the best solution	+	+	+	+	?
11	Don't use 3D if the number of data points is low	+	+	+	+	?
12	Map the data to an appropriate visual object	+	+	+	+	?
13	Test your designs with users	0	+	+	+	+
14	Use datatips for identification, education and validation	+	+	+	0	?
15	Provide a simple 3D navigational model	+	+	+	+	+
16	Use small multiples to encode multiple data attributes	+ & -	+ & -	+ & -	?	?
17	Use legends, scale and annotation	+	-	+	+	?
18	Do not rely on interaction	N/A	N/A	N/A	N/A	N/A
19	Occlusion is undesirable	0	+	+	0	+
20	Use interaction to explore large data sets	+ & -	+	+	0	?
21	Let users control visual bindings	+ & -	+	+ & -	+	+
22	Emphasise the interesting	0	+	+	0	?
23	Task specific	+	+	+	+	+
24	Overview	-	+	+	0	?
25	Zoom	-	+	+	0	?
26	Filter	-	+	+ & -	0	?
27	Details on demand	0	0	+	0	+ & -
28	Relate	0	+	+	0	?
29	History	+	+	+	0	+
30	Extract	N/A	N/A	N/A	N/A	+
31	Multiple linked (co-ordinated) views	-	-	+ & -	0	?
32	Direct manipulation	+	+	+ & -	+	?

Key

- 0 *No effect.* The heuristic has no known effect on the factor.
- + *Positive effect.* The heuristic has a positive effect on the factor e.g. increases performance, reduces the number of errors, etc.
- *Negative effect.* The heuristic has a negative effect on the factor e.g. decreases performance, increased the number of errors, etc.
- ? *Undefined effect.* The heuristic has an undefined effect on the factor. This is typically for subjective satisfaction where it is not possible to tell if a heuristic will have a positive or negative effect. However, occasionally it is difficult to determine the effect on other factors.
- N/A *Not applicable.* The heuristic does not apply to the factor.

Table B.1: Heuristic effect on usability factors.

B.2 Rationale

The following describes the reasoning behind the values assigned to each heuristic in table B.1.

1.0 Use a real world physics model

Learnability (+): Using a physics model which is based on the real world means the user already knows what to expect when they take an action rather than having to learn the consequences of an action.

Performance (+): The users familiarity with the model should allow them to perform more efficiently.

Errors (+): Having a good understanding of the model being used should help to prevent errors.

Retention (+): The user can concentrate on remembering unfamiliar aspects of the visualisation since the physics model is already known.

Subjective Satisfaction (?):

2.0 Visually refer all graphical objects to a reference context

Learnability (0):

Performance (+): Locating and comparing visual objects is easier therefore performance is improved.

Errors (+): The increased accuracy in correctly identifying or comparing visual objects means fewer errors should be made.

Retention (0):

Subjective Satisfaction (+): The visualisation is potentially easier to use and therefore more satisfying.

3.0 Use connotative mappings

Learnability (+): Connotative mappings are already familiar and therefore do not have to be learned from scratch.

Performance (+): Less effort is required in interpreting the mapping, which means performance is improved.

Errors (+): Familiar mappings are well understood and therefore less likely to be misinterpreted.

Retention (+): Familiar mappings may have been used for a long period of time, which leads to accurate recall. The user can concentrate on remembering unfamiliar aspects of the visualisation.

Subjective Satisfaction (+): The visualisation is potentially easier to use and therefore more satisfying.

4.0 Use an organisational device the user already knows

Learnability (+): Organisational devices are already familiar and therefore do not have to be learned from scratch.

Performance (+): Less effort is required in interpreting the visualisation, which means that performance is improved.

Errors (+): Familiar organisational devices are well understood and therefore less likely to be misinterpreted.

Retention (+): Familiar mappings may have been used for a long period of time, which leads to accurate recall. The user can concentrate on remembering unfamiliar aspects of the visualisation.

Subjective Satisfaction (+): An organisational device the user is familiar will tend to be more satisfying if the user is already happy with it. However, if the user is unhappy with the current working practice then forcing them to use it in a new system will have a negative effect on subjective satisfaction.

5.0 Use redundancy to aid discrimination and comprehension

Learnability (0): Using redundancy has no known effect on the time it takes to learn how to use the visualisation.

Performance (+): It is possible that the user may only have to recognise one visual feature to identify an object therefore performance is improved.

Errors (+): Redundant mappings reinforce each other which helps the user to confirm their interpretation of a visual object therefore reducing the chances of making an error.

Retention (+): Using redundancy gives the user a greater chance of remembering at least one of the mappings used thus increasing their chances of being able to use the system at a later date.

Subjective Satisfaction (?):

6.0 Use different visual dimensions differently

Learnability (-): Using multiple different mappings, especially non-connotative, between data attributes and visual features increases the cognitive load because there are more mappings to remember, this will therefore increase learning time.

Performance (+ & -): Mappings can be chosen which make items relevant to the current task more salient, this has a positive effect on performance because the time to locate such items is reduced. Multiple mappings increase cognitive load, which may have a negative effect.

Errors (+ & -): Mappings that make relevant items more salient reduce the possibility of including items irrelevant to the current task. Multiple mappings can be more easily misinterpreted.

Retention (0): Using multiple different mappings has no known effect on how the user maintains their knowledge of the visualisation over time.

Subjective Satisfaction (?):

7.0 Minimise Illusions

Learnability (0): Minimising illusions has no known effect on the amount of time it takes to learn how to use the visualisation.

Performance (+): Minimising illusions should make interpreting the scene quicker.

Errors (+): Illusions make interpreting the scene more difficult therefore mistakes can more easily be made. Thus minimising illusions helps to reduce errors.

Retention (0): Minimising illusions has no known effect on how the user maintains their knowledge of the visualisation over time.

Subjective Satisfaction (?):

8.0 Use colour carefully

Learnability (+): If colours are used which reflect cultural affordances it may help to reduce the time to learn. For example, in Western cultures red means danger, if a visualisation uses red to represent critical values, this may be learned in less time.

Performance (+ & -): If colour is used to highlight anomalous data but the task is to find data in the middle of the normal data range then this colour coding may hinder task performance. If however the task is to find anomalous data then performance may be improved since the relevant data is more obvious. In other words the colour should support the task.

Errors (+): Inappropriate use of colour can lead to misinterpretations of the data. Colour can also affect the perceived size of an object. Colour-blind users may have many difficulties with the colour mappings used. Avoiding these issues by using colour carefully may result in fewer errors.

Retention (0): Using colour can help users remember features of the data, but not how to use the visualisation. However, it could be argued that using colour in the interface e.g. colour coding buttons such as red for a button that performs a 'dangerous' action and green for an action that will not effect the data, may help the user remember how to use the visualisation interface.

Subjective Satisfaction (+): Users often prefer to use a colour display even if there is no functional benefit.

9.0 Use smooth animation and motion especially for temporal data

Learnability (0): Smooth animation and motion has no known affect on the time it takes to learn how to use the visualisation.

Performance (0): Smooth animation and motion has no known affect on the time it takes users to perform benchmark tasks.

Errors (+): There is less chance of important information being missed if changes from one state to another are taken in small steps therefore errors will be reduced.

Retention (0): Smooth animation and motion has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (+ & -): When we view objects in motion in the real world we are used to seeing a continuous path from one location to another. Replicating this effect in a visualisation should lead to positive subjective satisfaction. However, if the animation is too slow the user may become frustrated, this may lead to less satisfaction.

10.0 Visualisation is not always the best solution

Learnability (+): An algorithm that determines the results and is activated by a simple mouse click is more easily learned than a visualisation that performs the same function via user interaction.

Performance (+): An algorithm can be optimised; a visualisation will always have some dependency on the user.

Errors (+): An algorithm can be thoroughly tested so that there is minimal chance of errors users on the other hand can always make mistakes.

Retention (+): If an algorithm is used, it is not necessary for the user to understand how the algorithm works, just the inputs it requires. These inputs are usually limited and are associated with specific controls.

Subjective Satisfaction (?):

11.0 Don't use 3D if the number of data points is low

Learnability (+): 2D visualisations are often less complex therefore they take less time to learn.

Performance (+): Using a 3D visualisation for a low number of data points may have a negative effect on performance therefore using 2D will have a positive effect.

Errors (+): 2D visualisations are often less complex especially with respect to interaction, the simpler something is to use, the less likely errors will be made.

Retention (+): 2D visualisations are often less complex especially with respect to interaction, the simpler something is to use, the more easily it will be remembered.

Subjective Satisfaction (?):

12.0 Map the data to an appropriate visual object

Learnability (+): The user is likely to be familiar with the visual object which means it requires less time to learn.

Performance (+): Appropriate visual objects may be closely linked to the data structure and so may more closely match the users mental model of the data therefore improving performance.

Errors (+): As with performance using an obvious visual object to represent the data should help reduce errors.

Retention (+): Using an appropriate visual object the user recognises may allow them to recall more easily how to interpret and use that object thus improving retention.

Subjective Satisfaction (?): One person might like a traditional tree view, another might like using a Treemap as developed by Johnson and Shneiderman (1991).

13.0 Test your designs with users

Learnability (0):

Performance (+): Repeatedly testing the designs with users helps to find areas in which performance can be improved.

Errors (+): Reasons for errors and ways to correct them can be determined during testing.

Retention (+): Although designs may change, increasing the frequency of exposure to designs helps the user retain information between sessions.

Subjective Satisfaction (+): Incorporating changes suggested by users can increase user satisfaction.

14.0 Use datatips for identification, education and validation

Learnability (+): Datatips helps users learn and interpret the meaning of the visual objects used.

Performance (+): The ability to quickly identify items will increase user performance.

Errors (+): Datatips is an effective technique for confirming the user has found the appropriate item.

Retention (0): Datatips does not help the user remember how to use the visualisation.

Subjective Satisfaction (?):

15.0 Provide a simple 3D navigational model

Learnability (+): Less cognitive load should be required to learn the effects of actions taken.

Performance (+): The users performance should improve if navigating the space is made simple.

Errors (+): Fewer misinterpretations can be made with a simple model.

Retention (+): The actions and effects of a simple model are more easily triggered in memory. This is especially true if the model is based on a real-world physics model as in heuristic 1.0.

Subjective Satisfaction (+): The users subjective satisfaction should be improved because navigating the space is easier.

16.0 Use small multiples to encode multiple data attributes into graphical objects

Learnability (+ & -): Simple connotative small multiples can be easily learned. However, if the small multiples are made too complex, the mappings between data attribute and visual feature can be difficult to remember and learn.

Performance (+ & -): Numerous compact small multiples can help to improve performance because the close spatial location of each requires only a small scanning distance. Small multiples also have a high information load, which means several tasks can be performed using individual small multiples. However, as the number of small multiples increases so does the cognitive load on the user, which can then have a negative effect on performance. The complexity of each small multiple i.e. the number of data attributes to visual feature mappings, may also have a negative effect.

Errors (+ & -): Small multiples can help to reduce the number of errors. Unfortunately complex small multiples that use multiple mappings or have non-intuitive mappings are less easy to interpret which may lead to errors.

Retention (?):

Subjective Satisfaction (?):

17.0 Use legends, scale and annotation to aid memory and increase accuracy

Learnability (+): Legends help the user to remember and so learn what different symbols or mappings mean.

Performance (-): Initially the user may need to frequently refer to the legend therefore reducing performance. However, as the user becomes more familiar with the visualisation the legend should be referred to less often. At this point the legend should have less impact on performance.

Errors (+): Annotations such as labels increase user accuracy when reading information therefore reducing the number errors.

Retention (+): Legends do not have to be remembered thus freeing space that can be used to remember other information.

Subjective Satisfaction (?):

18.0 Do not rely on interaction

A visualisation should try to generate clear presentations of the data that can be easily interpreted without the need for interaction. This is a recommendation only and the usability factors are not strictly applicable to this heuristic.

Learnability (N/A):

Performance (N/A):

Errors (N/A):

Retention (N/A):

Subjective Satisfaction (N/A):

19.0 Occlusion is undesirable

Learnability (0): Occlusion has no known affect on the time it takes to learn how to use the visualisation.

Performance (+): Less occlusion results in increased speed in identifying and comparing objects.

Errors (+): Less occlusion results in increased accuracy in identifying objects.

Retention (0): Occlusion has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (+): A more readable display should reduce user frustration and so make the visualisation more satisfying to use.

20.0 Use interaction to explore large data sets

Learnability (+ & -): Although interaction may not help in learning the system, it can help to learn features of the data. As the number of controls increases so does the amount of information to be learned.

Performance (+): Interaction may allow rapid exploration of data.

Errors (+): Using interaction it is possible to confirm ideas and correct mistakes.

Retention (0): Allowing interaction with large data sets has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (?):

21.0 Let users control visual bindings

Learnability (+ & -): Once the user has chosen the mappings they should be quickly learnt. However, as well as learning how the visualisation works, the user would also have to learn how to control the data attribute to visual feature mappings.

Performance (+): Since the user has chosen the mapping between data attribute and visual feature they should be able to decode the mapping rapidly there performance should be improved.

Errors (+ & -): The user should have less difficulty in interpreting the mappings, however, each mapping has to be carefully chosen to avoid illusions and other perceptual problems. Inappropriate mappings can lead to an increase in the number of errors made.

Retention (+): Each mapping would reflect the users choice therefore they should be easy to recall.

Subjective Satisfaction (+): Users like to be in control, having the ability to choose their own visual bindings should improve subjective satisfaction.

22.0 *Emphasise the interesting*

Learnability (0): Emphasising interesting data has no known effect on the time it takes to learn how to use the visualisation.

Performance (+): The user can concentrate on the important information and is less distracted by any irrelevant information therefore improving performance.

Errors (+): Errors caused by using the wrong information are less likely.

Retention (0): Emphasising interesting data has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (?):

23.0 *Task specific*

Learnability (+): If the user understands the task and the visualisation supports that task or only a small number of tasks then learning time should be reduced. The more tasks the visualisation supports the more complex it is likely to be and therefore the longer it will take to learn.

Performance (+): Making the visualisation task specific should reduce the amount of irrelevant information that is displayed. This means that the user does not have to do any extra work e.g. filtering, to find the data relevant to the task therefore performance is improved.

Errors (+): Only showing information relevant to the task should reduce the chances of including inappropriate data in any analysis or results.

Retention (+): If the user's memory is not cluttered by irrelevant tasks or procedures presented by the visualisation their recall of how to use the visualisation should be improved.

Subjective Satisfaction (+): If a visualisation supports the tasks the user is interested in then their subjective satisfaction should be improved.

24.0 Overview

Learnability (-): Including an overview will require more time to learn how it relates to the main view.

Performance (+): An overview allows rapid navigation of the data set therefore improving performance.

Errors (+): An overview helps to reduce the possibility of the user becoming “lost” in the data.

Retention (0): Having an overview may help the user to recall features of the data but does not help them remember how to use the visualisation.

Subjective Satisfaction (?):

25.0 Zoom

Learnability (-): Users may have difficulty learning how different zoomed views of the data relate to each other. This is especially true when semantic zooming is used.

Performance (+): A zoom facility can help the user to rapidly and accurately identify items therefore improving performance.

Errors (+): A zoom facility should allow the user to accurately identify items therefore reducing the likelihood of errors.

Retention (0): Having a zoom facility may help the user to recall features of the data but does not help them remember how to use the visualisation.

Subjective Satisfaction (?):

26.0 Filter

Learnability (-): Multiple filter controls may be difficult to learn.

Performance (+): The ability to focus on items of interest, either by reducing the number of items displayed or making relevant items salient, should allow the user to perform tasks more efficiently.

Errors (+ & -): The ability to focus on items of interest, either by reducing the number of items displayed or making relevant items salient, should reduce the chances of users missing items. However, if the context of those items is lost, or incorrect filters are applied, errors may be introduced.

Retention (0): Having filter facilities may help the user to recall aspects of the data, because there is less data to remember, but does not help them remember how to use the visualisation.

Subjective Satisfaction (?):

27.0 Details on demand

Learnability (0): Providing details on demand has no known affect on the time it takes to learn how to use the visualisation.

Performance (0): Providing details on demand has no known effect on the user's performance using the visualisation.

Errors (+): Allowing the user to see the details of individual items should help to reduce errors.

Retention (0): Providing details on demand has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (+ & -): Users often have a need to see specific details about an individual item, providing this facility helps increase their satisfaction with the visualisation. However, some users like to be able to see both a high level overview of the data and specific details, providing several views rather than 'drilling down' to details may prevent users from becoming 'lost' in the data, therefore only showing very detailed information may have a negative effect on subjective satisfaction.

28.0 Relate

Learnability (0): Providing relate facilities has no known effect on the time it takes to learn how to use the visualisation.

Performance (+): Allowing the user to view the relationships between items may help them to perform their tasks more efficiently.

Errors (+): Allowing the user to view the relationships between items should help them to confirm their mental model of the data and so reduce errors.

Retention (0): Providing relate facilities has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (?):

29.0 History

Learnability (+): The history facility encourages the user to explore different aspects of the system without the fear that they may not be able to recover from their actions.

Performance (+): The ability to store and replay a set of actions rather than having to manual repeat a set of actions should improve performance.

Errors (+): Supporting an undo feature allows errors to be quickly and easily rectified.

Retention (0): Providing a history facility has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (+): The history facility lowers user apprehension. If they make a mistake it is simple to fix. This confidence in being able to explore and use the system will increase user satisfaction.

30.0 *Extract*

This is a recommendation only and the usability factors are not strictly applicable to this heuristic.

Learnability (N/A):

Performance (N/A):

Errors (N/A):

Retention (N/A):

Subjective Satisfaction (+): Providing the users with the facility to save the results of their explorations should improve subjective satisfaction.

31.0 *Multiple linked (co-ordinated) views*

Learnability (-): Setting up and understanding the relationships between multiple co-ordinated views can be a difficult process.

Performance (-): Integrating information between multiple views increases cognitive load therefore reducing performance.

Errors (+ & -): Using multiple views to analyse various aspects of the data can help to reduce errors. The complexity involved in co-ordinating multiple views may result in more errors.

Retention (0): Providing multiple linked views has no known effect on how well the user maintains their knowledge of how to use the visualisation.

Subjective Satisfaction (?): Some people like to use multiple views, other like to use details on demand techniques.

32.0 *Direct manipulation*

Learnability (+): Users are used to manipulating objects in the real world directly. Even when using a steering wheel 'control' to alter the direction of a car, we still have a direct physical connection rather than say trying to drive the car remotely. This close connection between action taken and resulting event means that direct manipulation takes less time to learn.

Performance (+): Selecting objects of interest by clicking on them may be faster than using control widgets. This is especially true if the selected objects are difficult to define using the controls.

Errors (+ & -): If the actions that can be performed on objects are visually obvious then errors may be reduced. If however the actions are hidden or are represented in some abstract way then the user may incorrectly manipulate the object, which may cause errors.

Retention (+): The effects of direct manipulation are often more easily remembered than consequences as a result of indirect actions.

Subjective Satisfaction (?): Some people like direct manipulation, others like controls or other techniques.

Appendix C: VISUALISATION PATTERNS

The patterns presented here are a first attempt at a set of visualisation specific patterns and are not meant to be definitive. Some of the patterns, such as *Appropriate Visual Objects*, *2D Representation*, *3D Representation*, *Dynamic Queries*, *Direct Manipulation*, *Interaction*, *Selection*, *Level of Detail*, and *Spatial Navigation*, are less concrete than others. Often these more abstract patterns are identifiable by their lack of examples. However, these patterns do help to form a more complete pattern language, often setting the context and forces that apply to lower level patterns. Different researchers may feel that these patterns should be removed or combined and since these are new patterns this may be justified.

Visualisation patterns fall into three main categories (although there is some overlap):

- *Structure Patterns*: Focus on defining the form and content of the visualisation i.e. the graphical scene, and as such relate primarily to data and mapping transforms.
- *Interaction Patterns*: Focus on the interaction mechanisms that can be used to achieve tasks and the visual effects they have on the scene, as such they relate primarily to graphical and rendering transforms.
- *Composition Patterns*: Provide solutions for the effective layout of multiple visualisations and the communication between them. Composition patterns are essentially an extension of GUI design patterns and should conform to similar GUI design principles such as consistency, feedback, use of shortcuts, etc. However, composition patterns do include solutions that are specific to visualisation and therefore the distinction is justified.

The pattern language shown in figure C.1 consists primarily of structure patterns and will be referred to as the *structure pattern language* (SPL). Obviously this pattern language is incomplete and there are undoubtedly more patterns at and between each level. As with other pattern collections new techniques and new solutions are constantly being developed and this refinement and extension to existing pattern languages is simply part of the natural process of pattern writing. It can be seen that several of the visualisation patterns exist at the same level and can be applied to most visualisation designs. Note also that some of Tidwell's (1999) GUI design patterns fit within this pattern language.

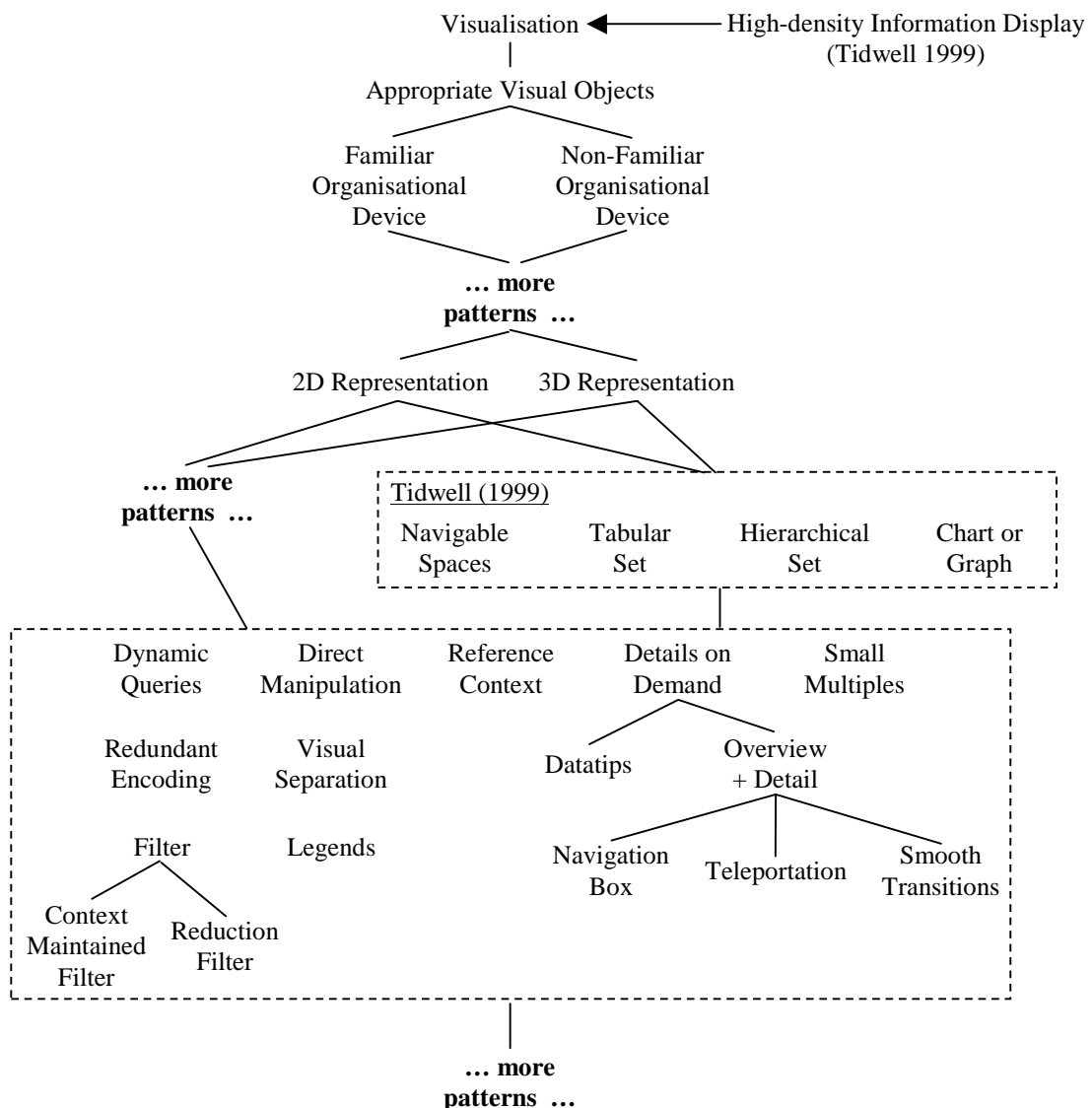


Figure C.1: Structure pattern language.

The pattern language shown in figure C.2 consists primarily of interaction patterns although there are links to the SPL. This will be referred to as the *interaction pattern language* (IPL). It is likely that additional patterns exist within the IPL however to maintain the clarity of the figure these additional patterns have been omitted. It should be noted that the grey nodes in the IPL are not actually patterns but instead help to define the context for lower level patterns and assist the designer in navigating the pattern language. Note also that some of Tidwell's (1999) GUI design patterns fit within this pattern language e.g. controls can be used to set filter parameter values, scrollbars can be used to view different areas of the data space, etc.

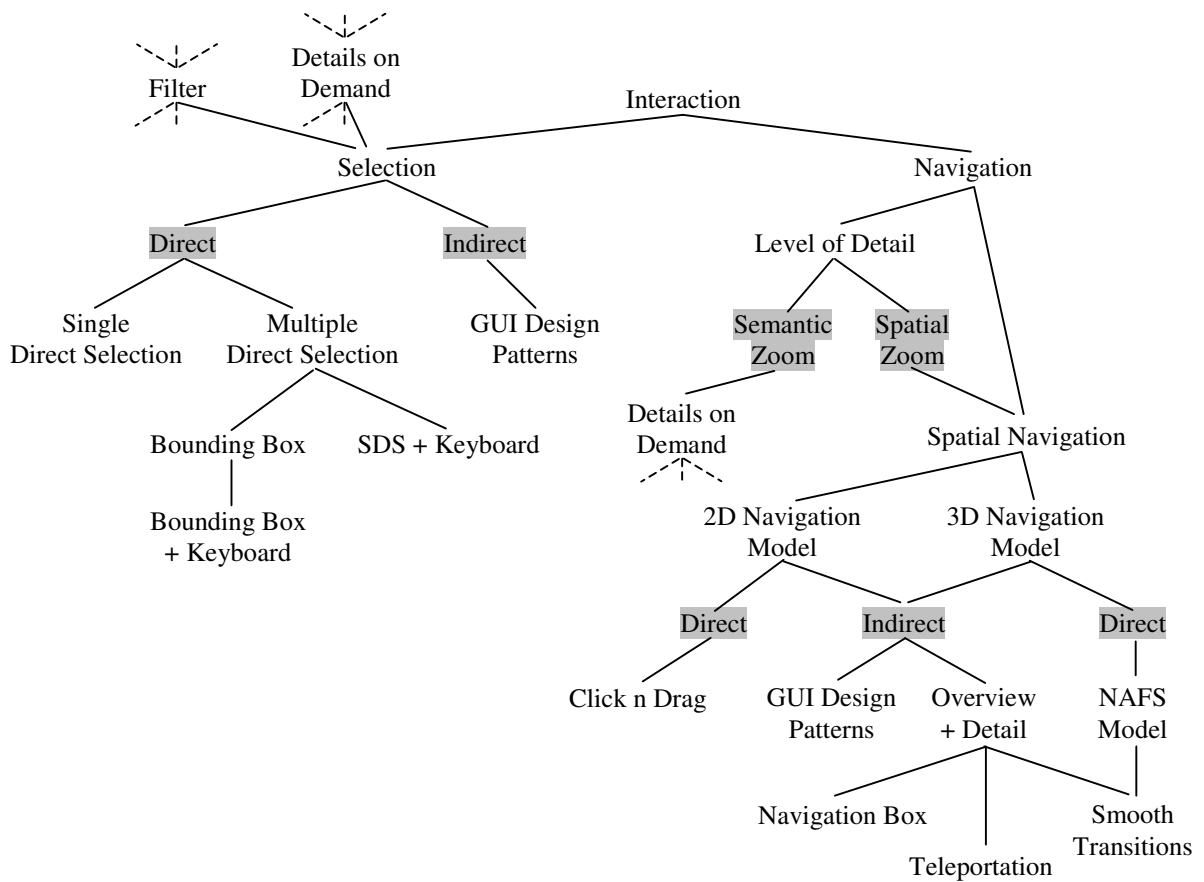


Figure C.2: Interaction pattern language.

Grey nodes act as “context” guides.

Appendix C: VISUALISATION PATTERNS

<i>Title</i>	Visualisation
<i>Context</i>	There is a need to visually represent information in such a way as to amplify user cognition.
<i>Problem</i>	How to represent the data in visual form.
<i>Forces</i>	<ul style="list-style-type: none"> • The user needs to be supported in the tasks relevant to their domain. • Typical tasks require human judgement i.e. cannot be easily coded algorithmically. • The user should be able to interpret the data effectively.
<i>Solution</i>	<p>Choose a visual representation that accurately represents the structure and content of the data and at the same time supports the tasks required by the user.</p> <p>This is a vague pattern that does not really assist the visualisation designer in choosing a specific visual form. However, it does set the scene for when a visualisation might be the appropriate solution.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • The Information Mural (Jerding and Stasko 1997) • CyberGeo Maps (Holmquist et al. 1998) • Narcissus (Hendley et al. 1995) • Numerous other examples.
<i>Related Patterns</i>	Appropriate Visual Objects

<i>Title</i>	Appropriate Visual Objects
<i>Context</i>	There is a need to visually represent information in such a way as to amplify user cognition.
<i>Problem</i>	How to represent the data in some visual form.
<i>Forces</i>	<ul style="list-style-type: none"> • The user needs to be supported in the tasks relevant to their domain. • Typical tasks require human judgement i.e. cannot be easily coded algorithmically. • The user should be able to interpret the data effectively.
<i>Solution</i>	<p>Map the data to a set of appropriate visual objects and organise those objects to form a graphical scene.</p> <p>There are a number of data factors that can influence the design of visual objects to represent data items. These factors include the data source, structure, type, range, dimensionality, and quantity.</p> <p>For scientific visualisations the data source can often be used as a model for the visualisation. For example, MRI data can often be presented using a model of the organ from which the data was gathered.</p>

	<p>Unfortunately this is not always possible, especially for information visualisation where the data source tends to be less concrete.</p> <p>The structure of the data can also be used as a basis for the visualisation. For example hierarchical data is often presented using a tree structure. These ‘natural’ structures can help to reinforce the users mental model of the data.</p> <p>When mapping data attributes to visual features the data type and range are both significant. Mackinlay (1986) and Salisbury (2001) have both ranked data types to visual features in terms of how efficiently the visual feature can encode the data type. The data range can also affect the choice of visual feature since some visual features are limited in the number of discrete values that they can represent.</p> <p>The task to be solved is also important. For example, if the relative locations of various cities is not important then there would be no reason to display the cities on a map.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • As well as those for Visualisation • Chart, Tree, Network, MRI displays, etc. • Numerous other examples.
<i>Related Patterns</i>	Familiar Organisational Device, Non-Familiar Organisational Device

<i>Title</i>	Familiar Organisational Device
<i>Context</i>	A system exists that the users are familiar with, that can support the data required to solve the task, and that is replicable in a software application.
<i>Problem</i>	How should the data be organised?
<i>Forces</i>	<ul style="list-style-type: none"> • The users may only be given minimal training. • The users should be able to anticipate what actions can be carried out using the display. • A system exists that the user is familiar with. • The existing system is replicable in a software application. • The existing system is an appropriate metaphor for the data and tasks. • There is an appropriate physical model upon which to base the representation.
<i>Solution</i>	<p>Use an organisational device the user already knows.</p> <p>An organisational device is a pre-defined representational structure that</p>

	<p>can be applied to the data. Chart based representations e.g. bar charts, scatter plots, etc., are good examples of structured representations that many users are familiar with from early education.</p> <p>Organisational devices can also be derived from the procedures and environment in which the user already works, e.g. floor plans, specialised financial charts, etc. Other organisational devices include maps, grids, hierarchies, time series, rooms, etc.</p> <p>Using an organisational device the user is familiar with reduces the learning time required since they are already familiar with how it works. The user may also more readily accept the device if they are happy with their current working practices.</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> • Head Trader (Brath 1999) <div data-bbox="715 846 1198 1160" data-label="Image"> </div> • Urban View (Salisbury 2001) <div data-bbox="715 1211 1198 1518" data-label="Figure"> </div> • AVID (Chuah 2000) • MacEachren et al. (1998) • The Table Lens (Rao and Card 1994) • Numerous other examples.
<p><i>Related Patterns</i></p>	<p>Visualisation patterns, GUI design patterns</p>

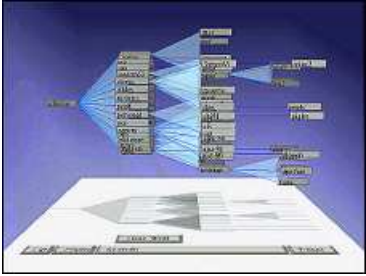
<p><i>Title</i></p>	<p>Non-Familiar Organisational Device</p>
<p><i>Context</i></p>	<p>No established system currently exists in the user domain that effectively supports the data and tasks within that domain.</p>
<p><i>Problem</i></p>	<p>How should the data be organised?</p>

Appendix C: VISUALISATION PATTERNS

<i>Forces</i>	<ul style="list-style-type: none"> • The users may require additional training. • The users should be able to anticipate what actions can be carried out using the display. • There is a need to develop a new visual representation due to inadequacies in existing systems, or because no such systems exist. • There is no physical model upon which to base the representation.
<i>Solution</i>	<p>Develop a novel organisational device.</p> <p>Since an existing organisation does not exist or there are deficiencies with existing systems, it may be necessary to develop a novel visual representation. The data, task, user, and usability factors will all influence the design of this representation.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • CyberGeo Maps (Holmquist et al. 1998) • Narcissus (Hendley et al. 1995) <div data-bbox="737 860 1174 1115" data-label="Image"> </div> <ul style="list-style-type: none"> • Numerous other examples.
<i>Related Patterns</i>	Visualisation patterns

<i>Title</i>	2D Representation
<i>Context</i>	There are a number of data items, from one or more datasets, that need to be viewed by the user.
<i>Problem</i>	How to represent the data in visual form?
<i>Forces</i>	<p>The number of data items is relatively low.</p> <ul style="list-style-type: none"> • The structure of the data or the source from which it was gathered does not require the use of the third dimension. • The dimensionality of the data is such that it does not warrant the use of the third dimension.
<i>Solution</i>	<p>Use a 2D representation.</p> <p>If the quantity of data items is relatively low, or the number of data dimensions that need to be displayed simultaneously can easily be incorporated into one or more views, or if the data fits naturally into a 2D space, then a 2D representation may be appropriate.</p>

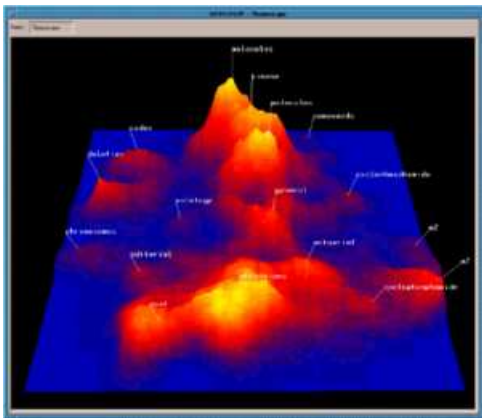
	2D representations have a number of advantages over 3D representations. In particular comparison and selection of individual data items is often much easier, navigation <i>models are easier to use and implement</i> , depth cues are not necessary, etc.
<i>Examples</i>	<ul style="list-style-type: none"> • CyberGeo Maps (Holmquist et al. 1998) • Urban View (Salisbury 2001) • The Table Lens (Rao and Card 1994) • Numerous other examples.
<i>Related Patterns</i>	Visualisation patterns, GUI design patterns.

<i>Title</i>	3D Representation
<i>Context</i>	There are a number of data items, from one or more datasets, that need to be viewed by the user.
<i>Problem</i>	How to represent the data in visual form?
<i>Forces</i>	<ul style="list-style-type: none"> • The number of data items may be high. • The structure of the data or the source from which it was gathered requires the use of the third dimension. • The dimensionality of the data is such that it does warrant the use of the third dimension. • Lack of screen space.
<i>Solution</i>	<p>Use a 3D representation.</p> <p>Using three dimensions provides access to a much large space in which to place the data items. However, 3D representations may require substantial additional work due to the problems of occlusion, the need for depth cues, the need for a simple 3D navigational model, etc. These considerations must be taken into account before the visualisation designer decided to use a 3D representation.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Arc Map (Cox et al. 1996) • Cone Tree (Robertson et al. 1993)  <ul style="list-style-type: none"> • Narcissus (Hendley et al. 1995) • Numerous other examples.
<i>Related Patterns</i>	Visualisation patterns, GUI design patterns.

Appendix C: VISUALISATION PATTERNS

<i>Title</i>	Reference Context
<i>Context</i>	There is a need for the user to accurately identify and compare individual objects but the display is such that the position of these objects in relation to each other is not clear.
<i>Problem</i>	How to show the relative spatial locations of different visual objects in a scene.
<i>Forces</i>	<ul style="list-style-type: none"> • The user needs to identify and compare objects. • The depth (or position) of those objects is difficult to judge.
<i>Solution</i>	<p>Visually refer all graphical objects to a reference context.</p> <p>A reference context is a mechanism that aids the user in spatially locating and comparing visual objects. This can be achieved by using grids, planes or aligning points in a particular dimension e.g. time. For example, anchor lines can be used to attach objects to planes or shadows can be used to help place objects in a scene. Using a reference context is particularly important when using three-dimensional representations.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • SeeIT (Brath 1999) <div data-bbox="715 987 1198 1294" data-label="Figure"> </div> • Topic Islands (Miller et al. 1998) <div data-bbox="727 1346 1187 1644" data-label="Figure"> </div> • Bermudez et al. (2000): Physiologic data visualisation.
<i>Related Patterns</i>	

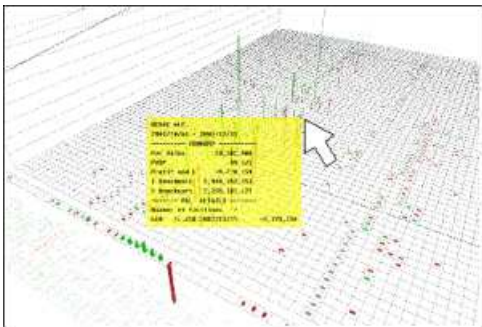
<i>Title</i>	Redundant Encoding
<i>Context</i>	There is a need to ensure accurate interpretation of information. Users may have visual deficiencies e.g. colour blindness. The visual features of a visual object may change but there is still a need to interpret the attached data value.

<i>Problem</i>	How to reinforce the similarities and differences between visual objects.
<i>Forces</i>	<ul style="list-style-type: none"> • Accuracy in data interpretation may be more important than performance. • Data dimensions can be mapped to several different visual features. • Users may need to gain confidence in their interpretation of the display.
<i>Solution</i>	<p>Encode the data dimensions using several visual features i.e. redundant encoding.</p> <p>A typical encoding may be to use triangles i.e. shape to represent troops and the colour of the triangles to represent troop status e.g. friendly troops green triangles, enemy troops red triangles. However, the troop status could be redundantly encoded in the shape, so enemy troops become red squares. This redundant encoding is then visually reinforcing similar troops and at the same time it still allows colour-blind users to correctly identify the troop status.</p> <p>Another advantage of redundant encoding is that an object's context can be maintained even if a visual feature is altered. This is useful when making relevant data salient. For example, the colour of selected troops could be yellow but their status could still be identified from their shape.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • ThemeScapes (Wise et al. 1995) 
<i>Related Patterns</i>	

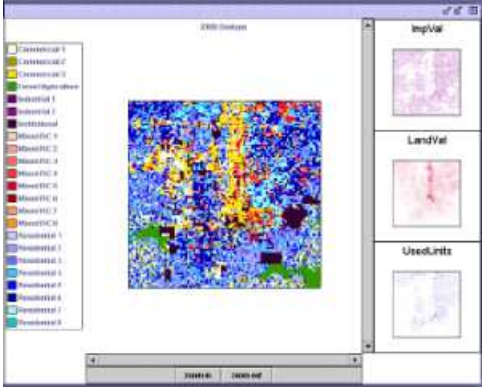
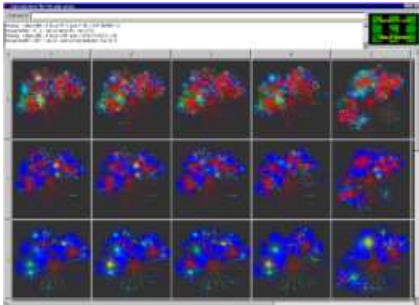
<i>Title</i>	Smooth Transitions
<i>Context</i>	The data is represented in such a way that the entire dataset cannot be viewed at one time. However, the user has a mechanism for spatial movement from one part of the dataset to another. Alternatively a large number of data items captured frequently over a period of time are being viewed using animation.

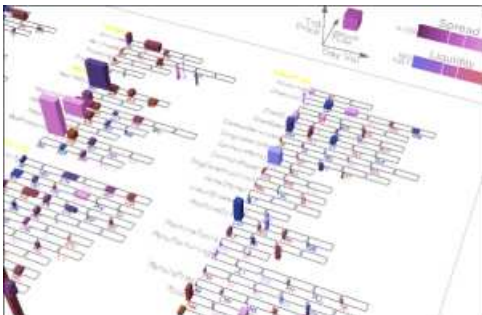
<i>Problem</i>	How to maintain the users context when travelling through the data space.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to maintain context in data space. • Moving from one part of the dataset to another. • Looking for changes in data over time.
<i>Solution</i>	<p>Use smooth animation and motion. This is especially true for temporal data.</p> <p>Animation or motion is an effective means of representing data that has some temporal aspect. The most important consideration when using this technique is to make the transition from one state to another as smooth as possible. Animation or motion that produces a large change of state distracts the user from the information and may cause important features to be missed. Large jumps from one location in a 3D environment to another may cause the user to become disoriented a smooth transition helps them to maintain their context in the environment. If possible the user should be able to control the rate of change and direction of the animation, this may allow them to rapidly and accurately explore the data.</p> <p>There are arguments for not using smooth animation or motion. If the task is to monitor data that may change only gradually over long periods of time, then using large discrete time steps may help the user to identify significant changes in the data. This implies that the application of this pattern may be task dependent.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Brath (1999) <p>Note it is difficult to find examples since researchers tend not to describe the rate at which transitions occur.</p>
<i>Related Patterns</i>	

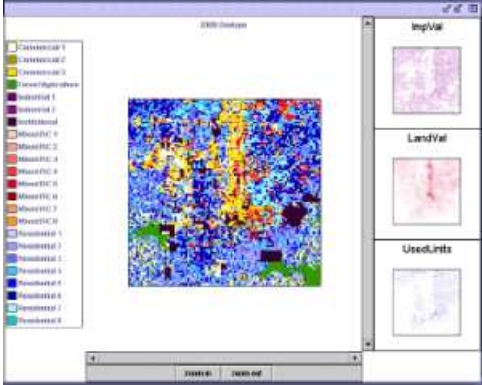
<i>Title</i>	Datatips
<i>Context</i>	Detailed information about individual data items is hidden.
<i>Problem</i>	How to show detailed information about a data item as the user requests it without permanently cluttering the display or using a separate view.
<i>Forces</i>	<ul style="list-style-type: none"> • Displayed items fail to show low level details. • User needs to see details about data items in order to confirm item content. • Detailed information should not be permanently displayed.
<i>Solution</i>	Use datatips for identification, education and validation.

	<p>Datatips is an interaction technique that can be used to temporarily view details about individual data items. Typically the details ‘pop up’ in a window at the location of the pointer thus maintaining the context between the item and the details. However alternatives, such as having a small permanent space somewhere on screen, are possible.</p> <p>Datatips allows the user to quickly identify an item and so confirm and strengthen their cognitive model of how the visualisation should be interpreted.</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> Inventory Viewer, SeeIt, Risk Movies (Brath 1999) 
<p><i>Related Patterns</i></p>	

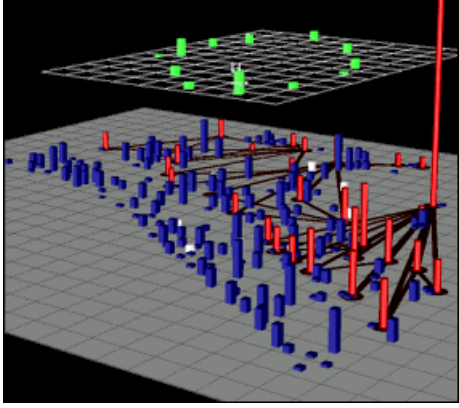
<p><i>Title</i></p>	<p>Small Multiples</p>
<p><i>Context</i></p>	<p>There are a number of related data dimensions per individual data item and at the same time collections of data items can be viewed.</p>
<p><i>Problem</i></p>	<p>How to package multidimensional data items into complete units that can be compared across data items or datasets.</p>
<p><i>Forces</i></p>	<ul style="list-style-type: none"> Multidimensional data. Need to compare items and possibly datasets.
<p><i>Solution</i></p>	<p>Use small multiples to encode multiple data objects into packaged graphical objects.</p> <p>A small multiple is a combination of visual features, each one representing a data dimension, packaged together and designed to function as a complete unit. For example, each ‘mark’ in a scatter plot is an example of a small multiple that uses position, shape, colour, and size to encode different data dimensions. However, if multiple scatter plots are used to represent either different data dimensions of a single dataset or the same data dimensions of multiple datasets, then each scatter plot can also be thought of as a small multiple. In this way several data dimensions or several datasets can be viewed</p>

	<p>simultaneously.</p> <p>As well as presenting multiple data dimensions small multiples <i>also provide rapid access to</i> large quantities of data. They are also a useful way of comparing datasets or visualising temporal data, especially if presented using a regular structured layout. However, small multiples can occupy relatively large amounts of space and designing effective small multiples can be difficult. Brath (1999) describes a serious case where the mapping of data dimension to visual feature was poorly chosen making it difficult for the user to accurately interpret the values associated with each small multiple.</p> <p>To distinguish small multiples from multiple linked views it is important to note that small multiples all use the same visual features and are all constructed in the same way, whereas multiple linked views are typically different but complementary.</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> • Urban View (Salisbury 2001)  <ul style="list-style-type: none"> • Chi and Card (1999)  <ul style="list-style-type: none"> • Brath (1999) • Keim and Kriegel (1993, 1994), Kiem et al. (1995)
<p><i>Related Patterns</i></p>	

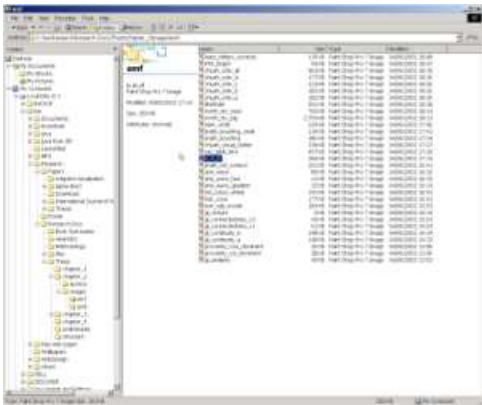
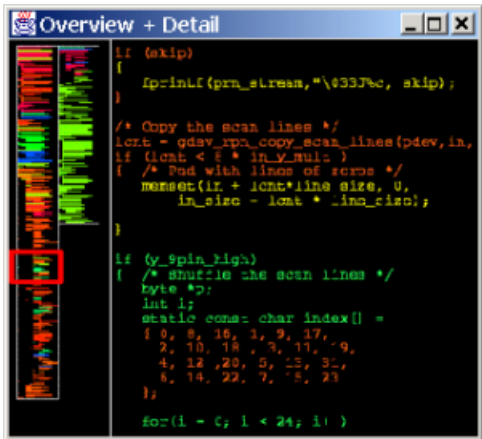
<i>Title</i>	Legends
<i>Context</i>	Data is encoded in the form of visual objects that are then organised in a scene.
<i>Problem</i>	How to remind the user how to interpret the display.
<i>Forces</i>	<ul style="list-style-type: none"> • Visual objects are displayed that encode multiple data attributes. • There is limited screen space. • Expert users may know how to interpret the scene.
<i>Solution</i>	<p>Use legends as a memory aid to assist the user in the interpretation of the display.</p> <p>Legends and annotations help to reduce some of the cognitive load on the user. Datatips is a good annotation technique that only requires attentive viewing when the user requires it and only temporarily increases clutter. Legends tend to be on screen all the time and display fixed information. However, if screen space is at a premium it should be possible to hide the legend. Legends and annotations also help the user learn how to interpret the display and are also useful when interaction with the visualisation is not possible e.g. a printed screen capture.</p> <p>Referring to legends may incur eye-shift, which can result in the user losing the items of interest. However with increased exposure to the visualisation the need to refer to the legend should be reduced.</p> <p>Labels in close proximity to objects also helps to increase identification accuracy but at the potential cost of more attentive viewing and an increase in visual clutter.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • SeeIt, SP500 Minimalist Representation (Brath 1999)  <ul style="list-style-type: none"> • Urban View (Salisbury 2001)

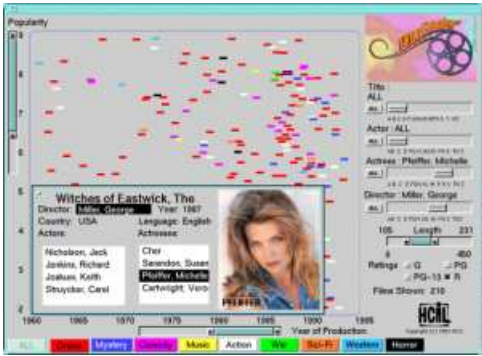
	 <ul style="list-style-type: none"> Any standard chart-based presentation e.g. bar chart, scatter plot.
<i>Related Patterns</i>	

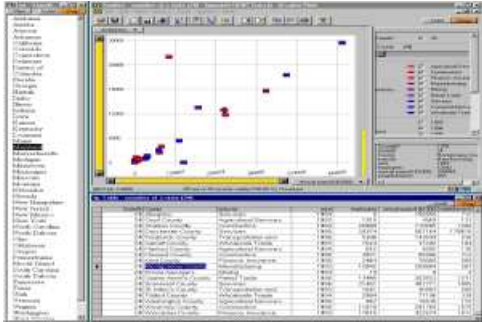
<i>Title</i>	Visual Separation (a.k.a Occlusion)
<i>Context</i>	A large number of overlapping items are present in the display. It may not be necessary to view all these items at once; evidence that more than one item exist in close proximity may be enough.
<i>Problem</i>	How to improve the readability of the display.
<i>Forces</i>	<ul style="list-style-type: none"> Many overlapping visual objects. Not all of the objects need to be viewed at once. The user may want to identify individual items. Limited screen space.
<i>Solution</i>	<p>Increase the visual separation between items or use transparency.</p> <p>Occlusion is the hiding of one object behind another, which can make it difficult for the user to read information or discriminate between different visual objects.</p> <p>The greater the distance between individual objects the less likely it is that objects will overlap. The exception to this is in 3D space objects in the foreground will almost inevitably occlude objects in the background regardless of the distance between them.</p> <p>A number of techniques <i>can be</i> used to overcome occlusion such as filtering the display thus reducing the number of objects (Filter), allowing to use to navigate around the scene in order to find a better viewpoint (Navigation), alternatively allowing the user to move the objects rather than move themselves e.g. Chuah’s et al. (1995) SDM techniques, or using transparency.</p>
<i>Examples</i>	<ul style="list-style-type: none"> Selective Dynamic Manipulation (Chuah et al. 1995)

		
<i>Related Patterns</i>	Filter, Navigation	

<i>Title</i>	Overview and Detail
<i>Context</i>	<p>The dataset is large, too large for all the details to fit in a single view, and there is a need to view details about subsets of data items. The data can be viewed at one or more levels of abstraction e.g. directories and files within a directory, aggregated document content and detailed document content, etc. Alternatively the dataset may be large and continuous but only a subset can be viewed at any one time e.g. map data.</p>
<i>Problem</i>	<p>How to display the entire contents of a large dataset at once, allow users to explore the dataset, and at the same time show details about subsets of items.</p>
<i>Forces</i>	<ul style="list-style-type: none"> • The entire contents of the dataset need to be displayed. • The user needs to know which part of the dataset is being viewed, i.e. where they are in the dataset, at all times. • Details about subsets of data items within the dataset are required. • Lack of screen space. • Large amount of data. • Easy navigation between different parts of the dataset.
<i>Solution</i>	<p>Show an overview of the entire dataset together with some visual indication as to which part of the dataset is currently being viewed. Show details about subsets of items in a separate view.</p> <p>The overview can be a scaled version of the main view, i.e. a spatial zoom, or some other representation, i.e. a semantic zoom. Since the overview tends to display a higher number of data items than any more detailed view it is necessary to use simple glyphs that minimise clutter, maximise use of screen space and portrait the data attributes most relevant to the task.</p>

	<p>The overview is usually spatially adjacent to the main view in order to reduce eye-shift.</p> <p>An overview provides several functions, it reduces the visual space the user has to search, it shows the user their current location and therefore context within the data set, and it acts as a navigation aid by helping them to decide which area of the data space to explore next.</p> <p>Typically the visual location indicator can also be used as a selection and navigation mechanism (Navigation Box).</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> Windows Explorer™  SeeSoft (Eick et al. 1992)  FilmFinder (Ahlberg and Shneiderman 1994) LifeLines (Plaisant, et al. 1996)
<p><i>Related Patterns</i></p>	<p>Navigation Box, Teleportation, Smooth Transitions</p>

<i>Title</i>	Filter
<i>Context</i>	The view displays data items that may be irrelevant to the users current task.
<i>Problem</i>	How to reduce the number of visual objects displayed or assist the user in finding and focusing on specific items of interest.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to find specific items. • May need to maintain context of those items found. • Large number of items in view, many of which may be irrelevant to the current task.
<i>Solution</i>	<p>Filter the display.</p> <p>Filtering facilities can help to reduce the number of data items displayed to those of specific interest to the user and their current task. This helps to reduce the visual complexity of the display. Typically widgets such as sliders, buttons, menus, etc., are attached to different attributes of the data. Manipulating these controls to specify the desired attribute values should cause a rapid update of the main display. This relates to Dynamic Queries. The choice of which attributes to allow the user to filter by may be determined from the task analysis stage.</p> <p>One potential problem with filtering is that because items are removed from the display, the context of the remaining items may be lost. One solution is to change a visual feature of the surviving items e.g. colour, rather than removing those that don't pass the filter. In effect making the relevant items more salient. This means that the surviving items context is preserved but the visual complexity of the display stays high. In addition altering a visual feature in such a way may alter or lose the meaning of the associated data value.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • FilmFinder (Ahlberg and Shneiderman 1994)  <ul style="list-style-type: none"> • LifeLines (Plaisant, et al. 1996)
<i>Related Patterns</i>	Context Maintained Filter, Reduction Filter, Dynamic Queries

<i>Title</i>	Details on Demand
<i>Context</i>	Data items presented with detailed information hidden.
<i>Problem</i>	How to show the user details about a specific item or items of interest.
<i>Forces</i>	<ul style="list-style-type: none"> • Lack of screen space. • Need to see specific data attribute values, i.e. text labels. • Permanent display of details about items.
<i>Solution</i>	<p>Use a separate window to display item details.</p> <p>A visualisation should allow the user to view data item attribute values. Datatips can be used to show a temporary details on demand window. The advantages of the datatips technique are that no mouse clicks are required, just pointing, and since the window is temporary it only interferes with the display for a short time. Both Brath (1999) and Eick (1995) recommend this approach.</p> <p>An alternative is to display the details in a completely separate window. This more permanent window is useful if the details need to be frequently referred to but at the cost of increasing screen clutter. So far, a mechanism for converting the popup window into a separate permanent window has not been described but this may be a useful feature.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Snap Together Visualisation (North and Shneiderman 1999a, 1999b, 2000)  <ul style="list-style-type: none"> • Windows Explorer™ • FilmFinder (Ahlberg and Shneiderman 1994)
<i>Related Patterns</i>	Datatips, Overview and Detail

<i>Title</i>	Interaction
<i>Context</i>	The user needs to be able to find relationships between data items, discover trends, perform comparisons, identify individual item details, etc. In short, the user needs to be able to explore and manipulate the data.

<i>Problem</i>	How to explore the dataset.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to explore and manipulate dataset. • Static display cannot support the user in solving the task. • Limited screen space.
<i>Solution</i>	<p>Use interaction to explore and manipulate the data.</p> <p>Interaction can overcome the limitations of a static display and a fixed amount of screen space thus increasing the amount of data that can be explored. In addition it also helps strengthen the users mental model of the data.</p> <p>Interaction can be used to facilitate almost any goal, but the most common uses of interaction are Selection which is the precursor to Filtering, Details on Demand, etc., and Navigation. Both of these primary interaction tasks can be performed indirectly using GUI controls (e.g. Dynamic Queries) or directly (e.g. Direct Manipulation).</p> <p>The importance of interaction requires that it be easy and intuitive to use.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Almost any visualisation
<i>Related Patterns</i>	Selection, Navigation, Dynamic Queries, Direct Manipulation, Filtering, Details on Demand

<i>Title</i>	Selection
<i>Context</i>	The user needs to select one or more data items to facilitate data exploration or manipulation.
<i>Problem</i>	In what way should the user be allowed to select the data items?
<i>Forces</i>	<ul style="list-style-type: none"> • The user may need to select individual items. • The user may need to select multiple items. • Selection is often a precursor to other tasks.
<i>Solution</i>	<p>Provide a mechanism that allows the user to select items based on user-supplied criteria.</p> <p>Before users can filter items, drill down to details about items, or manipulate them in some way, they must have a mechanism that allows them to select the items of interest. This selection process can either be done directly in the view or indirectly via GUI controls.</p> <p>This is a vague pattern that does not really assist the visualisation</p>

Appendix C: VISUALISATION PATTERNS

	designer in choosing a specific selection mechanism. However, it does provide a useful high-level pattern that sets the context for more specific patterns.
<i>Examples</i>	<ul style="list-style-type: none"> • Almost any visualisation
<i>Related Patterns</i>	Single Direct Selection, Multiple Direct Selection, GUI design patterns

<i>Title</i>	Navigation
<i>Context</i>	The view of the data is such that the user needs to move from one subset or aspect of the data to another. Therefore this transition between subsets can either be spatial or based on detail levels.
<i>Problem</i>	In what way should the user be allowed to navigate between different views of the data?
<i>Forces</i>	<ul style="list-style-type: none"> • The user may need to be able to move spatially from one view of the data to another. • The user may need to be able to alter the level of detail of the data they are interested in. • The transition between views should be as simple as possible. • The user does not want to become lost in the data.
<i>Solution</i>	<p>Provide a mechanism that allows the user to navigate between views of the data.</p> <p>There are two main types of navigation, spatial navigation and detail navigation. The user may need to move spatially from one area of the dataset to another e.g. when moving from one area of a map to another, or the user may wish to view the data at a different level of detail e.g. from a high level overview of all the data items to a low level detailed view of a small set of specific data items. Regardless less of the navigation type the transition between views should be easy to invoke, smooth, done in such a way that the user does not become lost, and probably reversible.</p> <p>This is a vague pattern that does <i>not really</i> assist the visualisation designer in choosing a specific navigation mechanism. However, it does provide a useful high-level pattern that sets the context for more specific patterns.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Almost any visualisation
<i>Related Patterns</i>	Level of Detail, Spatial Navigation

<i>Title</i>	Level of Detail
<i>Context</i>	The view of the data is such that the data items can be viewed at various levels of detail. This can include viewing items from a varying distances, high level overviews, low level detail views, etc.
<i>Problem</i>	How can the user alter the level of detail?
<i>Forces</i>	<ul style="list-style-type: none"> • The user may need to be able to move spatially from one view of the data to another. • The user may need to be able to alter the level of detail of the data they are interested in. • The transition between views should be as simple as possible. • The user does not want to become lost in the data.
<i>Solution</i>	<p>Provide a mechanism that allows the user to alter the level of detail of items of interest.</p> <p>In a system in which details about items are displayed but the users spatial location (e.g. distance) from items of interest means that those details cannot be accurately viewed, the user must alter their spatial location in order to increase the level of detail. For example, in a 3D representation the user may need to move towards objects in the scene to increase their level of detail. This could be classed as a spatial zoom.</p> <p>Alternatively the representation may aggregate items or show all items at a significantly reduced representation. In this case the user may be interested in viewing the individual components of a group or indeed the specific data attribute values of one or more data items. This could be classed as a <i>semantic zoom</i>. In this case the more detailed view may replace the existing view or be presented in a separate view.</p> <p>In both the spatial and semantic zooms the level of detail may decrease as well as increase.</p> <p>This is a vague pattern that does not really assist the visualisation designer in choosing a specific level of detail navigation mechanism. However, it does provide a useful high-level pattern that sets the context for more specific patterns.</p>
<i>Examples</i>	
<i>Related Patterns</i>	Details on Demand, Datatips, Overview and Detail, Spatial Navigation

<i>Title</i>	Spatial Navigation
<i>Context</i>	The view of the data is such that the user needs to move from one area of the data to another.
<i>Problem</i>	In what way should the user be allowed to navigate between different areas of the data?
<i>Forces</i>	<ul style="list-style-type: none"> • The user needs to be able to move spatially from one area of the data to another. • The transition between areas should be as simple as possible. • The user does not want to become lost in the data. • The user may want to see the entire dataset so that they can choose a location to navigate to.
<i>Solution</i>	<p>Provide a mechanism that allows the user to navigate between different areas of the data.</p> <p>Spatial navigation is often required when only part of the entire dataset can be viewed at any one time. A common example of this is map-based visualisations. Typically maps cover a large area of terrain only part of which can be viewed at once. The user must be able to navigate from one part of the map to another without losing their global position. This may require the use of a scaled version of the map on which is indicated the users current location.</p> <p>This is a vague pattern that does not really assist the visualisation designer in choosing a specific spatial navigation mechanism. However, it does provide a useful high-level pattern that sets the context for more specific patterns.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Narcissus (Hendley et al. 1995)
<i>Related Patterns</i>	Level of Detail, Overview and Detail, NAFS Model, Teleportation, Navigation Box, Smooth Transitions, Click n Drag, GUI design patterns

<i>Title</i>	Dynamic Queries
<i>Context</i>	The parameters of the task are well defined but only the user can supply the parameter value.
<i>Problem</i>	How to explore the dataset.
<i>Forces</i>	<ul style="list-style-type: none"> • Task parameters well defined. • Only the user can supply the task parameter values. • Rapid input of task parameters required. • Task types include selection, filtering, searching, and navigating.
<i>Solution</i>	Use dynamic query style interaction. Use standard GUI controls as a

	<p>mechanism for rapidly inputting and restricting task argument values.</p> <p>Dynamic queries usually involve some intermediate control such as a scrollbar, slider, button, etc. These controls can be used to provide the values for parameters defined as part of the task. Some controls also have the advantage that they can be associated with a specific range of values thus preventing erroneous input. For example a slider can have its minimum and maximum values set to the minimum and maximum values found in the dataset.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • FilmFinder (Ahlberg and Shneiderman 1994)
<i>Related Patterns</i>	

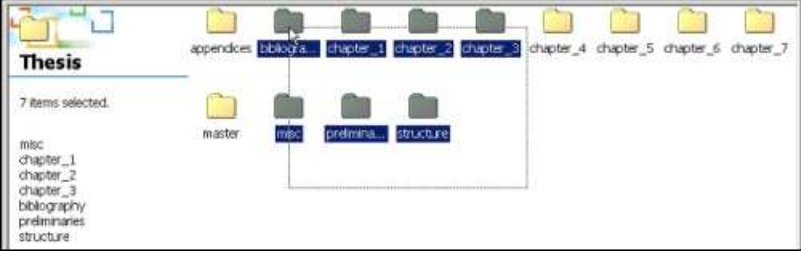
<i>Title</i>	Direct Manipulation
<i>Context</i>	The parameter values for a particular task are ill defined. Direct interaction with visual objects in the display is possible.
<i>Problem</i>	How to explore the dataset.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to explore and manipulate dataset. • Task types include selection, filtering, and navigating.
<i>Solution</i>	<p>Use direct manipulation style interaction. Allow the user to select and manipulate objects directly in the scene.</p> <p>Dynamic queries are an excellent mechanism for providing rapid, easily reversible actions on the scene. However, sometimes it is difficult or less intuitive to map an action onto a control. Instead the action should be applied directly to the object or objects in the scene. Selecting objects by using a bounding box is a good example. The classification that the user has determined by looking at the objects may be difficult to describe in terms of the values of those object's attributes and therefore difficult to set the associated controls correctly. However, by using a bounding box or by clicking on individual items, the user can easily select the desired set.</p> <p>During our daily routines we often pick up objects, move them, rotate them, and so on. Using direct manipulation in a display more closely matches our everyday experiences.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Selective Dynamic Manipulation (Chuah et al. 1995) • Brath (1999)
<i>Related Patterns</i>	

<i>Title</i>	Single Direct Selection
<i>Context</i>	The display consists of a number of visual objects. Action can be taken directly in the display.
<i>Problem</i>	How to select a single object directly.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to select a single object in the display. • Actions can be taken directly on the display. • User requires precise selection of individual data items.
<i>Solution</i>	<p>Allow the user to click directly on objects within the display using for example a mouse or some other input device.</p> <p>Allowing the user to select objects directly rather than via controls has the advantages of direct manipulation.</p> <p>Typically after the select action a graphical transform will be applied to the selected object making it more salient. This provides visual feedback confirming that an action has been taken and also allows the user to see which objects within the dataset have been selected.</p> <p>An alternative applying a graphical transform is to move or copy the selected objects into a separate view. This does have the advantage that the selected items can be grouped by proximity but may require more screen space and the selected items context may be lost.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Selective Dynamic Manipulation (Chuah et al. 1995)
<i>Related Patterns</i>	

<i>Title</i>	Multiple Direct Selection
<i>Context</i>	The display consists of a number of visual objects. Action can be taken directly in the display.
<i>Problem</i>	How to select a multiple objects directly at the same time.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to select multiple objects object in the display at the same time. • Actions can be taken directly on the display.
<i>Solution</i>	<p>Provide a mechanism that allows the user to select multiple items by selecting them directly in the display.</p> <p>This is a vague pattern that does not really assist the visualisation designer in choosing a specific selection mechanism. However, it does provide a useful high-level pattern that sets the context for more specific patterns.</p>
<i>Examples</i>	

<i>Related Patterns</i>	Bounding Box, SDS + Keyboard
<i>Title</i>	Bounding Box
<i>Context</i>	The display consists of a number of visual objects. Action can be taken directly in the display.
<i>Problem</i>	How to select a multiple objects directly at the same time.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to select multiple objects object in the display at the same time. • Actions can be taken directly on the display. • Items may be located in close spatial proximity to each other. • The user is less concerned about fine grain item selection.
<i>Solution</i>	<p>Allow the user to click and drag a box around the objects of interest using for example a mouse or some other input device. Show the extent of the box in the display as the interaction occurs.</p> <p>This is often referred to as the <i>bounding box</i> technique. The user clicks at a specific location and drags a box around the objects of interest. The box is displayed on screen as a way of providing visual feedback.</p> <p>This technique can be useful for making gross selections but is less useful when the user needs to select individual objects. If this is the Single Direct Selection pattern may be more appropriate.</p> <p>Allowing the user to select objects directly rather than via controls has the advantages of Direct Manipulation.</p> <p>Typically after the select action a graphical transform will be applied to the selected objects making them more salient. This provides visual feedback confirming that an action has been taken and also allows the user to see which objects within the dataset have been selected.</p> <p>An alternative applying a graphical transform is to move or copy the selected objects into a separate view. This does have the advantage that the selected items can be grouped by proximity but may require more screen space and the selected items context may be lost.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Windows Explorer™

Appendix C: VISUALISATION PATTERNS

	
<i>Related Patterns</i>	Bounding Box + Keyboard

<i>Title</i>	SDS + Keyboard
<i>Context</i>	The display consists of a number of visual objects. Action can be taken directly in the display.
<i>Problem</i>	How to select multiple objects directly.
<i>Forces</i>	<ul style="list-style-type: none"> • Need to select multiple objects in the display. • Single items can be selected at a time with/without loss of previously selected items. • Actions can be taken directly on the display. • User requires precise selection of individual data items.
<i>Solution</i>	<p>Allow the user to click directly on objects within the display using for example a mouse or some other input device and use the keyboard to control the effect of additional selections.</p> <p>Coupling Single Direct Selection with the keyboard allows the user to control how new selections effect the existing selected set. For example the Cntrl key is often used to add/remove individual items.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Windows Explorer™
<i>Related Patterns</i>	

<i>Title</i>	Bounding Box + Keyboard
<i>Context</i>	The display consists of a number of visual objects. Action can be taken directly in the display.
<i>Problem</i>	How to select multiple objects directly.
<i>Forces</i>	<ul style="list-style-type: none"> • Multiple items can be selected at a time with/without loss of previously selected items. • Need to select multiple objects object in the display at the same time. • Actions can be taken directly on the display. • Items may be located in close spatial proximity to each other. • The user is less concerned about fine grain item selection.
<i>Solution</i>	Use the Bounding Box technique together with the keyboard to control the effect of additional selections.

	Coupling Bounding Box with the keyboard allows the user to control how new selections effect the existing selected set. For example the Cntrl-key is often used to add/remove individual items, and the Shift-key can be used to add/remove ranges of items.
<i>Examples</i>	<ul style="list-style-type: none"> • Windows Explorer™
<i>Related Patterns</i>	

<i>Title</i>	Context Maintained Filter
<i>Context</i>	The display consists of a number of visual objects. Visual features of those objects can be altered without loss of information.
<i>Problem</i>	How to search for objects but maintain their context within the dataset.
<i>Forces</i>	<ul style="list-style-type: none"> • Context of matching items is important. • At least one visual feature of the visual object can be changed without loss of context or information.
<i>Solution</i>	<p>Apply a graphical transform. Alter a visual feature of the matching objects or augment the objects.</p> <p>By altering the appearance of those objects that match the filter criteria they become more salient within the display. However, the visual feature that is altered must not affect the objects context. For example, if the position of the object is relevant then another feature, such as colour, should be changed. If none of the visual features can be altered without affecting the objects context then it may be necessary to augment the visual object. For example a box can be placed around all matching objects.</p>
<i>Examples</i>	
<i>Related Patterns</i>	

<i>Title</i>	Reduction Filter
<i>Context</i>	The display consists of a number of visual objects.
<i>Problem</i>	How to filter out unwanted items from the display.
<i>Forces</i>	<ul style="list-style-type: none"> • Non-matching items are irrelevant. • The context of remaining items is not of interest to the user. • User focus on items of interest. • Possible increase in visual separation.
<i>Solution</i>	Apply a special form of graphical transform. Remove the object from the display.

Appendix C: VISUALISATION PATTERNS

	Unwanted items can simply be removed from the display. This often results in a less cluttered display and allows the user to focus on the items of interest.
<i>Examples</i>	
<i>Related Patterns</i>	

<i>Title</i>	2D Navigation Model
<i>Context</i>	The user is allowed to move through a 2D space using a standard desktop input device such as a mouse.
<i>Problem</i>	How to navigate in a 2D space.
<i>Forces</i>	<ul style="list-style-type: none"> • User can move through a 2D space. • The user should know where they are and where they can go. • The navigation model should be simple to use.
<i>Solution</i>	<p>Provide a simple 2D navigational model.</p> <p>Navigation can be carried out directly for example clicking in the scene at the location the user wishes to travel. Navigation may also be indirect for example via an overview or using standard GUI controls such as buttons, scrollbars, etc.</p> <p>The navigation model used should always keep the scene on screen. Allowing the user to view an empty space serves no purpose and may lead to the user becoming lost in the data space.</p> <p>It can also be useful for there to be some mechanism that allows the user to return to a previous location or to some pre-determined fixed location. In this way the user can always return to a 'safe' view of the scene.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Brath (1999)
<i>Related Patterns</i>	Click n Drag, Navigation Box, Teleportation, Smooth Transitions, GUI design patterns

<i>Title</i>	Click n Drag
<i>Context</i>	The user is allowed to move through a 2D space using a standard desktop input device such as a mouse.
<i>Problem</i>	How to navigate in a 2D space.
<i>Forces</i>	<ul style="list-style-type: none"> • User can move through a 2D space. • The navigation model should be simple to use. • The user can click directly in the view. • The users current location is displayed as an icon in the view.

<p><i>Solution</i></p>	<p>Allow the user to click on the icon that represents their current location and drag it to another, possibly off screen, area of the view.</p> <p>This is similar to Navigation Box but instead of moving an indicator in an overview, the user can move directly an indicator in the main view. Moving this indicator from one location to another updates the main view in the same way as Navigation Box. In contrast to Navigation Box the current location icon may actually be moved off screen as can be seen in the figure below. In this case without an overview the user cannot see what area they are moving into.</p> <div data-bbox="692 669 1222 965" style="text-align: center;"> <p>(a) (b)</p> </div> <p>This technique is also similar to the indirect mechanism of using scrollbars to view different sections of the data e.g. scrolling through a document in an editor.</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> • Computer Games?
<p><i>Related Patterns</i></p>	

<p><i>Title</i></p>	<p>3D Navigation Model</p>
<p><i>Context</i></p>	<p>The user is allowed to move through a 3D space using a standard desktop input device such as a mouse.</p>
<p><i>Problem</i></p>	<p>How to navigate in a 3D space.</p>
<p><i>Forces</i></p>	<ul style="list-style-type: none"> • User can move through a 3D space. • Need to try to prevent user from becoming disoriented. • Scene should always be on screen. • The navigation model should be simple to use.
<p><i>Solution</i></p>	<p>Provide a simple 3D navigational model.</p> <p>Navigation can be carried out directly, for example clicking in the scene at the location the user wishes to travel. Navigation may also be indirect for example via an overview or using standard GUI controls such as buttons, scrollbars, etc.</p> <p>The navigation model used should always keep the scene on screen.</p>

	<p>Allowing the user to view an empty space serves no purpose and may lead to the user becoming lost in the data space.</p> <p>In most cases the model should prevent the scene from ‘rolling’, so the horizon should always remain horizontal. This helps to maintain the users orientation and is consistent with their everyday experiences. It also prevents information encoded in a ‘positive’ direction from being misinterpreted.</p> <p>It can also be useful for there to be some mechanism that allows the user to return to a previous location or to some pre-determined fixed location. In this way the user can always return to a ‘safe’ view of the scene.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Brath (1999)
<i>Related Patterns</i>	Navigation Box, Teleportation, Smooth Transitions, GUI design patterns

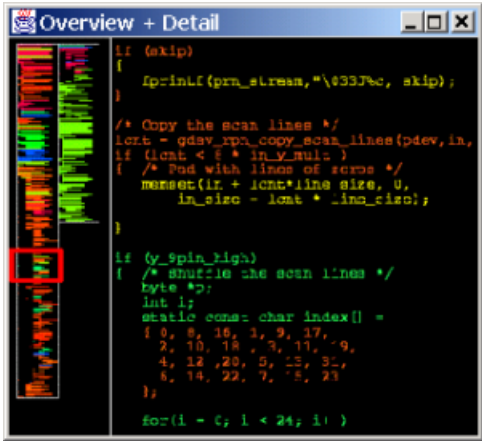
<i>Title</i>	NAFS Model
<i>Context</i>	The user is allowed to move through a 3D space using a standard desktop input device such as a mouse.
<i>Problem</i>	How to navigate in a 3D space.
<i>Forces</i>	<ul style="list-style-type: none"> • User can move through a 3D space. • Need to try to prevent user from becoming disoriented. • Scene should always be on screen. • The navigation model should be simple to use.
<i>Solution</i>	<p>Use the non-standard pilot model. This model should NOT replicate the standard flight simulator model.</p> <p>Typically this model will allow the user to move forward by moving the mouse forward and view items to the left or right by moving the mouse left or right i.e. rotation. In contrast to a standard flight simulator model, this model also allows the user to move backwards and prevents roll i.e. the horizontal should remain horizontal. This helps to maintain the users orientation and is consistent with their everyday experiences. It also prevents information encoded in a ‘positive’ direction from being misinterpreted.</p> <p>Additional actions such as fast forward, fast rotation, etc., may be achieved by combining mouse movements with mouse button presses. However the model must be consistent and be simple to use.</p> <p>The navigation model used should always keep the scene on screen.</p>

Appendix C: VISUALISATION PATTERNS

	Allowing the user to view an empty space serves no purpose and may lead to the user becoming lost in the data space.
<i>Examples</i>	<ul style="list-style-type: none"> • Brath (1999)
<i>Related Patterns</i>	Smooth Transitions

<i>Title</i>	Teleportation
<i>Context</i>	The display consists of an Overview and Detail.
<i>Problem</i>	How to allow the user to navigate the data space via the overview.
<i>Forces</i>	<ul style="list-style-type: none"> • The user needs to be able to jump from one location to another without viewing locations in between. • The user should be aware of the context of the destination within the dataset. • Discontinuous jumps between locations are commonplace.
<i>Solution</i>	<p>Transfer the user from one location to another directly.</p> <p>Allow the user to click directly on any location in the overview. When the user clicks in the overview the detailed view should be rapidly updated to reflect the new location without showing locations in between. For example in the Windows ExplorerTM users typically jump from one directory to another without viewing the directories in between. This technique allows the user to rapidly view different portions of the dataset.</p> <p>Teleportation can be used in combination with Navigation Box in which case the visual location indicator in the overview can be positioned by clicking at the desired location as well as dragging the indicator to the desired location.</p>
<i>Examples</i>	<ul style="list-style-type: none"> • Windows ExplorerTM • SeeSoft (Eick et al. 1992)
<i>Related Patterns</i>	Navigation Box

<i>Title</i>	Navigation Box
<i>Context</i>	The display consists of an Overview and Detail.
<i>Problem</i>	How to allow the user to navigate the data space via the overview.
<i>Forces</i>	<ul style="list-style-type: none"> • A simple mechanism for altering the contents of the detailed view. • Clear indication of item(s) being viewed in detail. This may include both the size and location for continuous data.
<i>Solution</i>	Place a moveable location indicator within the overview.

	<p>There should be a visual location indicator in the overview. Often this is simply a rectangle that represents both the location and area of the dataset being viewed. The user can then click and drag this indicator to the desired location and the contents of the detailed view should be updated accordingly. Similarly if the contents of the detailed view can be altered by some other means e.g. scrollbars, then the location indicator in the overview should be moved to reflect this change. In this way the location indicator and the detailed view are kept in sync.</p> <p>Whilst dragging the indicator the detailed view may be continuously updated (Smooth Transitions). Alternatively the detailed view may only update when the mouse button is released, in which case Teleportation is often an option.</p>
<p><i>Examples</i></p>	<ul style="list-style-type: none"> • SeeSoft (Eick et al. 1992)  <ul style="list-style-type: none"> • The Information Mural (Jerding and Stasko 1997)
<p><i>Related Patterns</i></p>	<p>Smooth Transitions, Teleportation</p>

Appendix D: ANALYSIS FACTOR QUESTIONS

Using the components in the analysis phase as a guide we propose a small set of questions the designer can ask which will help steer them to appropriate patterns and heuristics. These questions form a kind of ‘decision tree’ that can be used in addition to the pattern languages. The questions are not intended to guide the designer to fine grain design details but rather to ‘gross’ structural decisions. Therefore this design method could be thought to represent a simple model the designer can use to reach a pattern or heuristic quickly. The solution may then be used as a starting point from which the designer can get a feel for the overall structure of the visualisation and fill in the details using other methods.

Ideally questions relating to other analysis factors such as task, usability factors, and user modelling should also be developed. For example a task category question could be “Do humans find the task difficult?” If the answer is ‘yes’ then data transforms may be applicable, if ‘no’ then mapping transforms may be more appropriate. However although these questions have been addressed as part of this research they have not been made explicit due to time limitations. In addition questions regarding other analysis factors such as the usability factors have been covered extensively in existing HCI literature.

<i>Data Factor</i>	Source
<i>Question</i>	Is the data based on a physical model? E.g. MRI data, sensor data, etc.
<i>Answer: Yes</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Use the physical model as a basis for the visualisation</i> Rationale: The visualisation is more likely to match the users mental model. It is also easier for the designer to modify an existing satisfactory representation than to invent a new one. Advantages: Reduced learning time and improved performance for user. Established representation for the designer. Heuristics: 1.0, 3.0, 4.0, 12.0
<i>Answer: No</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Consider other data factors</i> Rationale: Other data factors may be relevant that lead to an initial design. • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> Rationale: Since there is no physical model upon which to base the visualisation, more traditional chart-based displays may be appropriate. Alternatively a novel visualisation structure may need to be created instead.

Appendix D: ANALYSIS FACTOR QUESTIONS

<i>Data Factor</i>	Source
<i>Question</i>	Is the data currently represented in an existing organisational device?
<i>Answer: Yes</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Use the existing organisational device as a basis for the visualisation</i> <p>Rationale: As long as existing users deem the existing organisation device satisfactory it is likely to be more readily accepted than presenting them with a completely new system.</p> <p>Heuristics: 4.0</p>
<i>Answer: No</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Consider other data factors</i> • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> <p>Rationale: Since there is no physical model upon which to base the visualisation, more traditional chart-based displays may be appropriate. Alternatively a novel visualisation structure may need to be created instead.</p>

<i>Data Factor</i>	Structure						
<i>Question</i>	<p>Does the data have any obvious structure?</p> <p>Can the data be mapped onto an obvious structure?</p>						
<i>Answer: Yes</i>	<p>Solutions:</p> <table border="1"> <tr> <td><i>Question</i></td> <td>Is the structure relevant to the task or the user?</td> </tr> <tr> <td><i>Answer: Yes</i></td> <td> <p>Solutions:</p> <ul style="list-style-type: none"> • <i>Replicate the structure in the visualisation</i> <p>Rationale: Since the structure is important, the designer has no choice but to use it as part of the visualisation.</p> </td> </tr> <tr> <td><i>Answer: No</i></td> <td> <p>Solution:</p> <ul style="list-style-type: none"> • <i>Consider other data factors</i> • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> <p>Rationale: Since the structure is not relevant, any other design may be valid.</p> </td> </tr> </table> <p>Heuristics: 12.0</p>	<i>Question</i>	Is the structure relevant to the task or the user?	<i>Answer: Yes</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Replicate the structure in the visualisation</i> <p>Rationale: Since the structure is important, the designer has no choice but to use it as part of the visualisation.</p>	<i>Answer: No</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Consider other data factors</i> • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> <p>Rationale: Since the structure is not relevant, any other design may be valid.</p>
<i>Question</i>	Is the structure relevant to the task or the user?						
<i>Answer: Yes</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Replicate the structure in the visualisation</i> <p>Rationale: Since the structure is important, the designer has no choice but to use it as part of the visualisation.</p>						
<i>Answer: No</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Consider other data factors</i> • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> <p>Rationale: Since the structure is not relevant, any other design may be valid.</p>						
<i>Answer: No</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Try to constrain the data to an existing structure</i> • <i>Consider other data factors</i> 						

Appendix D: ANALYSIS FACTOR QUESTIONS

	<ul style="list-style-type: none"> • <i>Use a chart-base representation</i> • <i>Invent a novel visualisation</i> <p>Rationale: Since there is no obvious structure other designs may be valid.</p>
--	---

<i>Data Factor</i>	Range
<i>Question</i>	Is the complete range of data values necessary to the task? Can the chosen visual feature support the range of data values?
<i>Answer: Yes</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Encode the range of values using the visual feature</i>
<i>Answer: No</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Reduce the number of unique data values</i> <p>Rationale: Quantitative data types can be converted to ordinal or nominal data types by categorisation.</p> <p>Advantage: The number of visual features that can more accurately support the converted data type is increased.</p> <p>Disadvantage: There is a loss of information.</p>

<i>Data Factor</i>	Dimensionality
<i>Question</i>	How many data dimensions need to be displayed at once?
<i>Answer: High (>6)</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Use multiple views</i> <p>Rationale: Each view can be designed to support a subset of the data dimensions.</p> <p>Advantages: Easily separation of tasks and related data. Traditional displays can also be made use of.</p> <p>Disadvantages: Loss of data context, although this may be maintainable using some visual feature e.g. connection, colour, etc., or interaction. Increase in eye shift also incurred. Increase in cognitive workload required to integrate multiple views.</p> • <i>Use glyphs or small multiples</i> <p>Rationale: Glyphs or small multiples can be designed to convey a number of data dimensions in a small space.</p> <p>Advantages: No loss of context, no eye shift, no integration difficulties.</p> <p>Disadvantage: Glyphs can be difficult to design. Increase in workload required to interpret glyph encoding.</p>
<i>Answer: Medium (>4)</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>As for large</i>

Appendix D: ANALYSIS FACTOR QUESTIONS

	<p>The solutions that apply to the display of a large number of data dimensions also apply to a medium number.</p> <ul style="list-style-type: none"> • <i>Use a simplified glyph</i> Rationale: A simplified glyph for example combining position, shape and colour, can be used for to represent a medium number of data dimensions. Advantages: More easily interpreted, easier to design, and often require less space than complex glyphs. Simply glyphs can often be combined with traditional chart-based representations.
<i>Answer: Low (<4)</i>	<ul style="list-style-type: none"> • <i>Use a chart-based representation</i> Rationale: Traditional chart-based displays are capable of supporting low numbers of data dimensions.

<i>Data Factor</i>	Quantity
<i>Question</i>	What is the data quantity?
<i>Answer: Large</i>	<p>Solutions:</p> <ul style="list-style-type: none"> • <i>Use three dimensions</i> Rationale: Increase in the amount of space available. Disadvantages: Occlusion, navigation issues, depth perception. • <i>Use filters</i> Rationale: Filters can be used to allow the user to reduce the amount of data that needs to be displayed at any one time. Advantage: Simple method for reducing data quantity. Disadvantage: If visual objects are removed from the display then the context of the remaining items may be lost. • <i>Use Overview and Detail technique</i> Rationale: A reduced representation can show the entire dataset while a separate view shows the details of a subset of the data. Advantages: Context preserved, navigation aid, visual search aid.
<i>Answer: Medium</i> <i>Answer: Small</i>	<p>Solution:</p> <ul style="list-style-type: none"> • <i>Use a chart-based representation</i> Rationale: Low data quantities should be compatible with chart-based representations such as scatter plots, bar charts, etc. Advantage: Users are familiar with these types of display. • <i>Use filters</i> See filter solution in large data quantity answer.

Appendix E: MILITARY TASK SCENARIOS

In order to develop visualisations that can help support command and control teams it is necessary to understand and define the types of tasks that such teams carry out. In particular the types of questions that are asked by planners and more importantly the combat operations teams who are trying to execute plans.

The basic premise is that combat operations are given a set of mission plans that must be executed within some time frame. However problems can occur with plans in one of two ways. A plan may make incorrect assumptions about various aspects of a mission or a plan may start to fail due to some unforeseen event occurring during the execution of a mission. In either case it will be necessary for combat operations to develop new plans ‘on the fly’ and to try to judge the impact of these modifications to the overall mission plan set. It is hoped that by using visualisations, plan problems can be identified more easily, new plans can be developed more rapidly, and their consequences can be more easily assessed.

Initially a set of military scenarios were developed each of which consisted of a number individual mission plans typical of those executed by combat operations. Analysis of the military scenarios led to a set of task objects including *mission*, *resource*, *route*, *objective*, *timing constraints*, *mission package*, and *mission interdependence*, each of which is defined in more detail in section 5.2.2. These common task objects would be relevant to any visualisation designs.

E.1 Mission Plan Problems

Analysis of the mission scenarios also led to a generic set of common problems experienced by combat operations teams. These are as follows:

- *Pre-mission*
 - A particular resource is unavailable at the start of the mission. Either there are no resources of this type left, or there are none that will be in a ready state by the start of the mission.
 - A particular resource is not capable of fulfilling the requirements of the mission in some way. For example it cannot travel the distance indicated without being refuelled.
 - Certain timing constraints cannot be met, for example a resource must be in two locations at the same time.
 - The regions along the route planned have changed ownership, which may make them too risky for the current mission.
 - The target/objective is no longer at the location specified.

- *During-mission*
 - Timing constraints cannot be met. For example there is a strong headwind that may slow an aircraft down, or, an aircraft is damaged by flak and therefore cannot operate at peak efficiency.
 - The target/objective is no longer at the location specified.
 - A particular resource fails on route.

E.2 Task Scenarios

Using the military scenarios, the task objects, and the common problems experienced by command and control teams, a series of task scenarios were developed. Each task scenario describes a typical problem that may arise during the execution of a military plan. The task scenarios were used to determine the types of tasks combat operations personnel perform and in particular the types of questions they ask themselves when trying to solve mission plan problems. For completeness the reasoning behind the questions was also discussed.

E.2.1 Scenario 1 (pre-mission assessment)

Situation: Combat operations personnel are presented with a complete mission plan including a set of resources, the route they must follow, the objective and the timing constraints.

Task: Find any problems/conflicts with this mission plan.

Sub-tasks/Questions:

Resource Checks:

- Availability: Are all the resources used in the mission available?
 - Will there be enough desired resources in a ready state to meet the mission plan requirements?
 - What is the state of each desired resource?
 - How many of each desired resource are in a ready state?
 - How many of those in a ready state will be available by the mission start time?
 - Are the desired resources at the bases indicated in the mission plan?
 - What are the locations of the desired resources?
- Capability: Are all the resources capable of meeting their individual objectives?
 - Is it the right resource for the job?
 - Does resource e.g. bomb type B, have the DMPI size?
 - Is resource R capable of reaching location L following route P? E.g. does it have enough fuel? Is the target within range? Can the resource meet the altitude requirements planned?
- Reliability: How reliable is each resource used in the mission plan?
 - How often do resources of type R fail?

Route Checks:

- Risks: How risky is each region along the specified route?
 - Are there any regions along the route that are a particular cause for concern?
 - Which forces does each region belong to?
 - What is the strength of the enemy forces within each region? (see stage 3)
 - Are the resources capable of functioning in the conditions of each region? For example, is it too stormy for jets to fly, or is the terrain too marshy for tanks? Another example would be if tanks needed to cross a river is there a bridge for them to cross? If not then a new route or a plan to include a portable bridge resource needs to be developed.
- How long is each resource in a friendly/neutral/enemy region?
- What are the weather conditions like along the route?
 - Can the resource be used in these conditions? Is it an acceptable risk?

Objective Checks:

- Is it an enemy target? Is it on the Target Nomination List (TNL)?
- Is it the correct target? How can you tell?
- Does the target have any issues relating to the rules of engagement? Is it a sensitive target?
- How critical is the target (objective) to the enemy?

Time Checks:

- Can each resource get to the required location at the required time?
 - Do resources meet at a pre-determined location within a specified time window?
 - How long is the time window?
 - How long can each resource safely maintain its position? Does this meet the time window?
- Do any resources need to be in two locations at the same time?
- Do any resources need to be in multiple locations in a time frame they cannot meet?

Reasoning:

For each of the task objects the combat operations personnel must ask questions similar to those above. If the answers to any of these questions are not as expected then it is reasonable to assume that there is a problem with the plan.

E.2.2 Scenario 2

Situation: One of the problems with a mission plan is that a SAM site exists along the route planned.

Task: Assess the reliability of this information using the intelligence reports. How reliable is the information that indicates there is a SAM site at the specified location? Rank the reliability as one of the following: *very reliable, fairly reliable, poor reliability, unreliable, don't know*.

Sub-tasks/Questions:

- Find the report(s) that relate to SAM sites in the specified location.
- Find reports of other activities in the area that may provide supporting evidence for a SAM site being located in the area. For example, vehicles heading in towards the site, aircraft destroyed in the region, etc.
- Find reports that may provide contradictory evidence. For example reports stating that there is no unusual activity in the area.
- Find other reports by the same author(s).
- How has the rate of submitted reports changed over time?

Reasoning:

To determine the reliability of a report you must either find evidence to support the report or evidence of contradictory information. To do this you can look for other reports that explicitly state that a SAM site is in the area or you can look for reports that describe enemy activity in the area, which may indicate a SAM site, or other installation. For example if reconnaissance shows a major infrastructure in the region, or if supply vehicles are often seen heading in the appropriate direction, this may indicate the presence of some installation. Similarly if friendly aircraft are regularly lost in the region this may also indicate the presence of something like a SAM site. Contradictory reports may state no activity in the region. The reliability of these contradictory reports may also need to be investigated.

E.2.3 Scenario 3

Situation: There is a choice between an original planned route and two alternative routes that meet the requirements of the mission plan using the same resources and the same time constraints.

Task: For each region within each route, rank the regions by the following criteria:

- *Strength:* the size of the force + the type of force (e.g. SAM, troops, tank, etc)
- *Threat:* strength + location
- *Political risk:* sensitivity = types of object within region e.g. hospital building vs. HQ building

Sub-tasks/Questions:

- How many forces of each type are there?
- What is the size of each force?
- What is the type of each force?
- What is the location of each force?
- In what direction is each force heading? Stationary, towards, away, etc.
- What types of building are in close proximity to the objective? Are the resources involved accurate or reliable enough to hit the target and not sensitive buildings?
- Can a more direct route be taken? What effect does this have on timing, risk, cost, etc?

Reasoning:

The regions a resource follows along a specified route each have an associated risk. This risk depends on who owns the region and the strength of the force within that region. The number of and type of resources in the force determine its strength. For example if force (F1) consists of 50 men and 5 tanks, and force (F2) consists of 0 men and 20 tanks, F2 may be considered stronger than F1. However, the location of that force plays a significant role in determining the *threat* of a force. If a river with no bridges blocks F2, tanks only, it may be considered less of a threat than F1, mostly men, who can swim across the river. The mobility of the force may also need to be considered.

Assessing the risks of various routes in terms of the forces involved in the regions being crossed and other factors such as weather conditions may reduce the risks to friendly forces.

The political risk of a target within a region also needs to be assessed. If the target is near a sensitive object such as a hospital or school a dumb bomb may not be the best choice. If the sensitivity of surrounding objects is unknown, the current political climate may need to be taken into account. The reliability of reports detailing building types in the designated region needs to be considered.

E.2.4 Scenario 4

Situation: Monitoring of a mission as it is being executed.

Task: Is the mission going according to plan?

Sub-tasks/Questions:

- Where is each resource located?
- Is each resource heading in the right direction?
- Are all the resources involved still operational?
- Is each resource on time?
- Has any new intelligence been gathered that may affect the mission?

Reasoning:

This is an example of *monitoring*. During the execution of a mission the user must monitor the location of the resources, make sure they are still operational and judge if they are going to arrive at their designated locations on time. Other factors that must be monitored include the weather conditions, enemy movement, and any new intelligence information gathered.

E.2.5 Scenario 5

Situation: During mission execution a SAM site destroys a resource. There are no equivalent resources available that are not currently assigned to a mission.

Task: Select a set of missions that are capable of providing an equivalent resource.

Sub-tasks/Questions:

Resources:

- In terms of specification are there any equivalent resources?
 - Is exactly the same resource type available?
 - Is a compatible resource available?
- See *resource checks* – scenario 1.

Route:

- What is the route that the resource will have to take to meet the mission objectives?
 - See *route checks* – scenario 1.
- Compare the risks of the routes within each mission.
 - See *route checks* – scenario 1.

Timing:

- Are any of the equivalent resources capable of meeting the original mission timing requirements?
 - Are the selected missions before, after, or during the current mission?
 - Can the resource be used in both missions?
 - Can the resource meet the time requirements of the first mission?
 - Can the resource be made ready for use in its original mission? E.g. Can it be refuelled in time?
- See *timing checks* – scenario 1.

Missions:

- Compare the criticality of the current mission and the missions in the selected set:
 - Compare the number of dependant waypoints within each mission.
 - Compare the number of dependant missions.
 - Compare the priority of each mission objective and the dependant mission objectives.
 - Rank the missions in order of criticality, risk, objective importance (criticality?), etc.
- What is the likelihood the resource will return and can be used in its original mission?
 - Compare the risks of each mission.
 - Is it an acceptable risk to take? If the resource cannot return to its original mission does it matter? What effect does it have?
- What are the effects on the following factors:
 - Cost
 - How is the financial cost affected?
 - Is there any impact on human cost? For example, if this resource is usually less accurate than the intended resource and the intended target is close to a sensitive object e.g. hospital, this may affect the human cost.
 - Can you afford to lose this resource in the original mission? For example there is a greater cost involved in losing a “one off” fighter that uses specialised equipment than in losing a standard fighter.
 - Timing
 - Will missions have to be delayed? If so by how much? Is this acceptable?
 - How narrow is the time window for missions to be completed on time?

- If a further unexpected delay occurs, is there now enough time to complete dependant missions?
- Criticality
 - Have missions become more or less critical?
 - How critical is the resource to each mission?
 - How critical is each objective in each mission?
- Risk
 - Is each mission more or less likely to fail/succeed?
 - Is there a greater risk that the alternative resource used will fail? E.g. Miss the target.

Reasoning:

If a specified resource is unavailable there are several options that can be considered. The mission could be aborted, the mission could be delayed until the specified resource becomes available, an equivalent unassigned resource could be used, or, an equivalent resource could be 'pinched' from an existing mission plan. This stage focuses on determining the consequences of taking a resource from one mission and using it in the current mission.

The first thing to check is that there is an equivalent resource. Ignoring timing and route constraints an equivalent resource is one that can perform the same job as the original resource. Once an equivalent resource has been found then many of the resource checks defined in stage one need to be applied.

At this point a set of missions capable of providing an equivalent resource will have been identified. For each mission in the set the route, timing and mission questions need to be asked and the answers to each compared.

Having found an equivalent resource the route that resource must take to become part of the current mission must be determined. Each route must be compared in terms of risk and the other risk checks defined in stage one.

Timing constraints, which may be closely linked to the potential routes that can be taken, must also be considered. In addition, the relative times and time windows within each mission must be compared. Again the timing checks in stage one should also be applied.

If all of the above can be satisfied it is still necessary to compare the effects on each mission. The criticality of each mission is extremely important. Criticality has been defined in terms of the number of dependant waypoints from the time the resource is required, the number of subsequent missions that are dependant on the current mission and the priority of each mission objective. This is a repetitive process, so if mission C depends on mission B and mission B depends on mission A then the criticality of all three missions and their interdependency must be taken into consideration.

The decision as to which mission to take the resource from is a process of 'weighing up' various factors. How critical is each mission in comparison to the current mission? How many and how important are the missions dependent on the selected mission in comparison to those in the current mission? What are the risks associated with each mission? If the resource fails to return to the mission it was originally intend to be used in, what effect does it have? Each

of these factors needs to be taken into account, ranked, compared and analysed in various ways so that a decision can be reached.

E.2.6 Scenario 6 (post-mission analysis)

Situation: The mission has been executed.

Task: Assess various mission criteria.

Sub-tasks/Questions:

- Was the objective (target) achieved (hit)?
 - How much damage was inflicted on the target?
- What was the reliability of the resources used?
- How many and what types of resources were lost? How does this effect stock level?
- What was the total financial cost of the mission?
- How long did the mission take? How long was the mission supposed to take?
- Were any parts of the mission late? If so, why? What effect did this have?
- How does using resource R1 compare to using resource R2? (especially if the two different resources are used in the same mission)
- During the mission time which forces gained the most area?

Reasoning:

Upon completion the planners need to know how successful the mission was. The primary question is, was the target hit, and if not then how much damage was done. If the target was not hit then the reason why must be investigated. It may have been due to an unreliable resource, inappropriate use of resources, to poor weather conditions causing delays thus allowing the enemy to move position, unreliable intelligence, and so on. Ultimately once the success and impact of the mission has been established, the need for a similar mission against the same target can be determined.

In addition it is essential to know what effect the mission has had on stock levels so that new resources can be manufactured if necessary, or resources can be transferred from one location to another in preparation for future missions.

E.2.7 Scenario 7 (campaign analysis)

Situation: A number of missions have been executed.

Task: Assess various campaign criteria.

Sub-tasks/Questions:

- Which forces occupy the largest area?
- Since the start of the campaign which forces have gained the most area? In which regions?
- Rank the regions by damage inflicted and sustained.
- Determine the least risk routes and the regions that need to be occupied.

- Analyse various resource characteristics over time.
 - How accurate are certain resources?
 - How often do certain resources fail? What is their success rate?
- Compare resource characteristics.
 - Is resource R1 more accurate in general than resource R2?
 - How often is R1 used in comparison to resource R2?
- Compare various ATO characteristics.
 - Compare number tasked, airborne, successful, failed, etc., over time.
 - Analyse trends in number of sorties, number of missions, etc., during campaign.

Reasoning:

It is important to establish how the campaign is progressing overall as this will influence future missions. The relative changes in territory and in particular the current status of critical regions needs to be reviewed. The reasons why regions have been lost need to be investigated and the impact of regions gained and regions lost needs to be determined.

In addition to analysing changes in ownership of territory, finding trends in ATO characteristics, missions, etc., may also be useful. If particular bases have a high success rate it may be important to establish why. This may be due to the resources used, the training personnel at the base have received and so on. The measure of success may also be important, is it based on time, cost, damage inflicted on the enemy, etc? Successful bases may be given priority for critical missions.

Finally, comparisons between resources, their accuracy, reliability, cost, quantities, etc., may prove useful to future missions. A weapon with close to 100% success rate is an obvious choice if it is to be used against a critical target in a politically sensitive area.

It should be noted that the questions relating to campaign analysis might include some of those from post-mission analysis.

List of References

- Ahlberg, C., Shneiderman, B. (1994). Visual information seeking: tight coupling of dynamic query filters with starfield displays. In: *Human Factors and Computing Systems*. Boston, USA, 24-28 April 1994. pp 313-317.
- Alexander, C. *et al.* (1977). *A Pattern Language: Towns, Buildings, Constructions*. New York : Oxford University Press.
- Alty, J.L., *et al.* (2000). A Framework for Engineering Metaphor at the User Interface. *Interacting with Computers*, 13(2), pp 301-322.
- Baldonado, M.Q.W., Woodruff, A., Kuchinsky, A. (2000). Guidelines for Using Multiple Views in Information Visualization. In: *Advanced Visual Interfaces 2000*. Palermo, Italy, 23-26 May 2000. pp 110-119.
- Bermudez *et al.* (2000). Data Representation Architecture. Visualization Design Methods, Theory and Technology Applied to Anesthesiology. In: *ACADIA 2000*. Washington D.C, USA, 19-22 October 2000. pp 91-102.
- Borchers, J.O. (2000a). Interaction Design Patterns: Twelve Theses. In: *CHI'2000: Workshop on Pattern Languages for Interaction Design: Building Momentum*. The Hague, Netherlands, 2-3 April 2000.
- Borchers, J. O. (2000b). A Pattern Approach to Interaction Design. In: *International Conference on Designing Interactive Systems*. New York, USA, 16-19 August 2000. pp 369-378.
- Brath, R. (1997a). 3D Interactive Information Visualization : Guidelines from experience and analysis of applications. In: *The 7th International Conference on Human-Computer Interaction*. San Francisco, California, USA, 24-29 August 1997. pp 865-868.
- Brath, R. (1997b). Concept Demonstration Metrics for Effective Information Visualization. In: *IEEE Symposium on Information Visualization*. Phoenix, Arizona, USA, 20-21 October 1997. pp 108-111.
- Brath, R. (1999) Effective Information Visualization : Guidelines and Metrics for 3D Interactive Representations of Business Data. Masters of Computer Science Thesis, Graduate Department of Computer Science, University of Toronto, Canada.

- Bruce, V., Green, P.R., Georgeson, M.A. (1996). *Visual Perception: Physiology, Psychology and Ecology*. 3rd ed. East Sussex, UK : Psychology Press.
- Callaghan, T.C. (1989). Interference and Domination in Texture Segregation: Hue, Geometric Form, and Line Orientation. *Perception & Psychophysics*, 46(4), pp 299-311.
- Callaghan, T.C. (1990). Interference and Dominance in Texture Segregation. In: Brogan, D., ed. *Visual Search*. New York : Taylor & Francis. pp 81-87.
- Card, S., Mackinlay, J., Shneiderman, B. (1999). *Readings in Information Visualization : Using vision to think*. San Francisco, California : Morgan Kaufmann.
- Carr, D.A. (1999). Guidelines for Designing Information Visualization Applications. In: *Ericsson Conference on Usability Engineering '99*. Stockholm, Sweden, 1-3 December 1999.
- Casner, S.M. (1991). A Task-Analytic Approach to the Automated Design of Graphic Presentations. *ACM Transactions on Graphics*, 10(2), pp 111-151.
- Chi, E., Card, S.K. (1999). Sensemaking of Evolving Web Sites Using Visualization Spreadsheets. In: *IEEE Symposium on Information Visualization*. San Francisco, California, USA, 24-29 October 1999. pp 18-25.
- Chuah *et al.* (1995). SDM: Selective Dynamic Manipulation of Visualizations. In: *The 8th ACM Symposium on User interface and Software Technology*. Pittsburgh, Pennsylvania, USA, 15-17 November 1995. pp 61-70.
- Chuah, M.C. (2000) AVID : Automatic Visualization Interface Designer. PhD Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.
- Cook, S.M. (2002). Visualisation Cognitive Walkthroughs. Report to B. Wilkins at University of Birmingham, UK. Unpublished QinetiQ Report.
- Cox, K.C., Eick, S.G., He, T. (1996). 3D Geographic Network Displays. *SIGMOD Record*, 24(4), pp 50-54.
- Csinger, A. (1992). The Psychology of Visualization. University of British Columbia, Department of Computer Science. Technical Report; TR-92-28.
- Dix, A. *et al.* (1998). *Human-Computer Interaction*. 2nd ed. London : Prentice-Hall Europe.

Eick, S.G. (1995). Engineering Perceptually Effective Visualizations for Abstract Data. In: Nielson, G.M., Hagen, H., Muller, H. eds. *Scientific Visualization Overviews, Methodologies and Techniques*. USA: IEEE Computer Science Press. pp 191-210.

Eick, S. G., Steffen, J. L., Sumner, E. E. (1992). SeeSoft - A Tool for Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering*, 18(11), pp 957-968.

Eysenck, M.W., Keane, M.T. (1995). *Cognitive Psychology*. 3rd ed. East Sussex, UK : Erlbaum (UK) Taylor & Francis.

Feibush, E., Gagvani, N., Williams, D. (1999). Geo-Spatial Visualization for Situational Awareness. In: *IEEE Visualization 1999*. San Francisco, California, USA, 24-29 October 1999. pp 441-443.

Feiner S., Beshers C. (1990). Visualizing n-Dimensional Virtual Worlds with n-Vision. *Computer Graphics*, 24(2), pp 37-38.

Fincher, S. (1999). What is a Pattern Language? In: *Late-Breaking Papers, CHI'99 Human Factors in Computing Systems*. Pittsburgh, Pennsylvania, USA, 15-20 May 1999.

Foley, J.D., Van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics*. Boston, Massachusetts : Addison-Wesley.

Furnas, G. W. (1982). The FISHEYE view: A new look at structured files. Bell Laboratories, Murray Hill, New Jersey. Technical Report; #82-11221-22.

Graham, M., Kennedy, J., Benyon, D. (2000). Towards a methodology for developing visualizations. *International Journal of Human Computer Studies*, 53(5), pp 789-807.

Granlund, A., Lafrenière, D. (1999). PSA: A Pattern-Supported Approach to the User Interface Design Process. In: *Usability Professionals Association*. Scottsdale, Arizona, USA, 29th June – 2nd July 1999.

Granlund, A., Lafrenière, D., Carr, D.A. (2001). A Pattern-Supported Approach to the User Interface Design Process. In: *The 9th International Conference on Human-Computer Interaction*. New Orleans, USA, 5-10 August 2001. pp 282-286.

Griffiths, R., Pemberton, L., Borchers, J.O. (1999). Usability Pattern Language: Creating a Community. In: *Workshop at INTERACT '99*. Edinburgh, Scotland, UK, 30-31 August 1999.

Griffiths R.N., Pemberton, L. (2001) "Don't Write Guidelines Write Patterns!"
[<http://www.it.bton.ac.uk/staff/lp22/guidelinesdraft.html>] viewed on 8th April 2002.

Haber, R.B., McNabb, D.A. (1990). Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In: Nielson, G., Shriver, B., Rosenblum, L.J., eds. *Visualization in Scientific Computing*. California : IEEE Computer Society Press.

Healey, C.G., Booth, K.S., Enns, J.T. (1993). Harnessing Preattentive Processes for Multivariate Data Visualization. In: *Graphics Interface '93*. Toronto, Canada, 19-21 May 1993. pp 107-117.

Healey, C.G., Booth, K.S., Enns, J.T. (1995). Visualizing Real-Time Multivariate Data Using Preattentive Processing. *ACM Transactions on Modeling and Computer Simulation*, 5(3), pp 190-221.

Healey, C.G., Booth, K.S., Enns, J.T. (1996). High-Speed Visual Estimation Using Preattentive Processing. *ACM Transactions on Computer Human Interaction*, 3(2), pp 107-135.

Healey, C.G., Enns, J.T. (1998). Building Perceptual Textures to Visualize Multidimensional Datasets. In: *IEEE Visualization '98*. Research Triangle Park, North Carolina, 18-23 October 1998. pp 111-118.

Healey, C.G., Enns, J.T. (1999). Large Datasets at a Glance: Combining Textures and Colors for Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(2), pp 145-167.

Healey, C. G., Amant, R. St., Elhaddad, M. (1998). ViA: A Perceptual Visualization Assistant. In: *The 28th Applied Imagery Pattern Recognition Workshop*. Washington, DC, USA, 13-15 October 1999.

Hendley, R. J., Drew, N. S., Wood, A. M., Beale, R. (1995). Case Study - Narcissus: Visualising Information. In: *IEEE Information Visualization '95*. Atlanta, Georgia, USA, 29th October - 3rd November 1995. pp 90-96.

Holmquist, L.E., Fagrell, H., Busso, R. (1998). Navigating Cyberspace with CyberGeo Maps. In: *The 21st Information Research Systems Seminar In Scandinavia (IRIS)*. Saeby, Denmark, 8-11 August 1998. pp 391-401.

- Inselberg, A., Dimsdale, B. Parallel Coordinates: A tool for visualizing multidimensional geometry. In: *IEEE Visualisation '90*. Los Alamitos, California, USA, October 1990. pp 360-375.
- Jerding D. F. and Stasko J.T. (1997). The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. Graphics, Visualization and Usability Center, College of Computing, Georgia Institute of Technology, Atlanta. Technical Report; GIT-GVU-97-24.
- Johnson, B., Shneiderman, B. (1991). Treemaps : a space-filling approach to the visualisation of hierarchical information structures. In: *IEEE Visualization '91*. San Diego, California, USA, October 1991. pp 284-291.
- Kandogan, E., Shneiderman, B. (1996). Elastic Windows: Improved Spatial Layout and Rapid Multiple Window Operations. In: *Advanced Visual Interfaces '96*. Gubbio, Italy, 27-29 May 1996. pp 29-38.
- Keim D.A., Kriegel H.P. (1993). Possibilities and Limits in Visualizing Large Amounts of Multidimensional Data. In: *IFIP Workshop*. San Jose, California, USA, 23-24 October 1993.
- Keim D.A., Kriegel H.P. (1994). VisDB: Database Exploration using Multidimensional Visualization. *IEEE Computer Graphics and Applications*, 14(5), pp 40-49.
- Keim, D., Kriegel, H.P., Ankerst, M. (1995). Recursive Pattern: A technique for visualizing very large amounts of data. In: *IEEE Visualization '95*. Atlanta, Georgia, USA, 29th October - 3rd November 1995. pp 279-286.
- Koffka, K. (1935). *Principles of Gestalt Psychology*. New York : Harcourt-Brace.
- Lamping, J., Rao, R. (1996). The Hyperbolic Browser: A Focus + Context Technique for Visualizing Large Hierarchies. *Journal of Visual Languages and Computing*, 7(1), pp 33-55.
- Lange, S. *et al.* (1995). Problem-oriented visualisation of multi-dimensional data sets. In: *International Symposium on Scientific Visualization*. Cagliari, Italy, 27-29 September 1995. pp 1-15.
- Lim, K.Y., Long, J.B. (1995). *The MUSE Method for Usability Engineering*. New York : Cambridge University Press.

MacEachren, A.M. *et al.* (1998). Geographic Visualization: Designing Manipulable Maps for Exploring Temporally Varying Georeferenced Statistics. In: *IEEE Information Visualization Symposium*. Reliegh-Durham, North Carolina, USA, 19-20 October 1998. pp 87-94.

Macklin, C.M. and Dudfield, H. (2001) Campaign Assessment Visualisation Techniques for Command Teams. DERA Farnborough, Hampshire, UK. Note: Draft Report. [Private Correspondence]

Mackinlay, J. (1986). Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2), pp 110-141.

Miller, N. *et al.* (1997). The Need for Metrics in Visual Information Analysis. In: *Workshop on New Paradigms In Information Visualization*. Las Vegas, Nevada, USA, 13-14 November, 1997. pp 24-28.

Miller, N.E. *et al.* (1998). TOPIC ISLANDS – A Wavelet-Based Text Visualization System. In: *IEEE Information Visualization '98*. Research Triangle Park, North Carolina, USA, 18-23 October 1998. pp 189-196.

Morse, E., Lewis, M., Olsen, K.A. (2000). Evaluating visualisations : using a taxonomic guide. *International Journal of Human Computer Studies*, 53(5), pp 637-662.

Munzner, T. (1997). H3: Laying out large directed graphs in 3D Hyperbolic space. In: *IEEE Symposium on Information Visualization*. Pheonix, Arizona, USA, 20-21 October, 1997. pp 2-10.

Nielsen, J. (1992). The Usability Engineering Life Cycle. *IEEE Computer*, 25(3), pp 12-22.

Nielsen, J. (1993). *Usability Engineering*. Boston Massachusetts : Academic Press Professional.

Nielsen, J. (1994). Heuristic Evaluation. In: Nielsen, J., Mack, R.L., eds. *Usability Inspection Methods*. New York : John Wiley & Sons.

Nielsen, J., Molich, R. (1990). Heuristic evaluation of user interfaces. In: *ACM CHI'90 Conference*. Seattle, USA, 1-5 April 1990. pp 249-256.

North, C., Shneiderman, B. (1999a). Snap-Together Visualization: Coordinating Multiple Views to Explore Information. University of Maryland Computer Science Dept. Technical Report; #CS-TR-4020.

- North, C., Shneiderman, B. (1999b). Snap-Together Visualization: Evaluating Coordination Usage and Construction. University of Maryland Computer Science Dept. Technical Report; #CS-TR-4075.
- North, C., Shneiderman, B. (2000). Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. In: *Advanced Visual Interfaces '00*. Palermo, Italy, 23-26 May 2000. pp 128-135.
- Pfleeger, S.L. (1987). *Software Engineering: The production of quality software*. Indianapolis : Macmillan Publishing Co.
- Plaisant, C. *et al.* (1996). LifeLines : Visualizing Personal Histories. In: *Conference on Human Factors in Computing Systems*. Vancouver, British Columbia, Canada, 13-18 April 1996. pp 221-227.
- Rao, R., Card, S.K. (1994). Table lens: Merging graphical and symbolic representations in an interactive focus plus context visualization for tabular information. In: *ACM Conference on Human Factors in Computing Systems*. Boston, Massachusetts, USA, 24-28 April 1994. pp 318-322.
- Redmond-Pyle, D., Moore, A. (1995). *Graphical User Interface Design and Evaluation (GUIDE): A Practical Process*. London : Prentice-Hall.
- Rheingans, P., Landreth, C. (1995). Perceptual Principles for Effective Visualizations. In: Grinstein, G., Levkowitz, H., eds. *Perceptual Issues in Visualization*. Berlin : Springer-Verlag. pp 59-73.
- Robertson, G., Card, S., Mackinlay, J. (1993). Information Visualization using 3D Interactive Animation. *Communications of the ACM*, 36(4), pp 57-71.
- Roberston, P., De Ferrari, L. (1994). Systematic Approaches to Visualization: Is a Reference Model Needed? In: Rosenblum, R.A. *et al.*, eds. *Scientific Visualization: Advances and Challenges*. New York : Academic Press.
- Salisbury, L.D.P. (2001). Automatic Visual Display Design and Creation. PhD Thesis, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, USA.
- Senay, H. and Ignatius, E. (1994). A Knowledge-Based System for Visualization Design. *IEEE Computer Graphics and Applications*, 14(6), pp 36-47.

- Shneiderman, B. (1994). Dynamic Queries for Visual Information Seeking. *IEEE Software*, 11(6), pp 70-77.
- Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualization. In: *IEEE Symposium on Visual Languages*. Boulder, Colorado, USA, 3-6 September 1996. pp 336-343.
- Shneiderman, B. (1997). Direct manipulation for comprehensible, predictable and controllable user interfaces. In: *International conference on Intelligent User Interfaces*. Orlando, Florida, USA, 6-9 January 1997. pp 33-39.
- Shneiderman, B. (1998). *Designing the User Interface : Strategies for Effective Human-Computer Interaction*. 3rd ed. Reading Massachusetts : Addison-Wesley.
- Sutcliffe, A.G., Ennis, M., Hu, J. (2000). Evaluating the effectiveness of visual user interfaces for information retrieval. *International Journal of Human Computer Studies*, 53(5), pp 741-763.
- Tidwell, J. (1999) "Common Ground: A Pattern Language for Human-Computer Interface Design" [http://www.mit.edu/~jtidwell/common_ground.html] viewed on 8th April 2002.
- Trafton, J.G. *et al.* (2000). Turning pictures into numbers: extracting and generating information from complex visualizations. *International Journal of Human Computer Studies*, 53(5), pp 827-850.
- Tufte, E.R. (1983). *The visual display of quantitative information*. Cheshire, Connecticut : Graphics Press.
- Vanderdonckt, J., Zucchinetti, G. (1995). Key Activities for a Development Methodology for Interactive Applications. In: Benyon, D., Palanque, P., eds. *Critical Issues in User Interface Systems Engineering*. Berlin : Springer-Verlag. Chapter 7.
- Ware, C. (1999). *Information Visualization : Perception for Design*. San Francisco, California : Morgan Kaufmann
- Welie, M., Veer, G.C. (2000). A Structure for Usability Based Patterns. In: *CHI'2000: Workshop on Pattern Languages for Interaction Design: Building Momentum*. The Hague, Netherlands, 2-3 April 2000.

- Welie, M., Veer, G.C., Eliens, A. (2000). Patterns as Tools for User Interface Design. In: *International Workshop on Tools for Working with Guidelines*. Biarritz, France, 7-8 October 2000. pp 313-324.
- Wesson, J.L. (1999a). Integrating HCI with the Software Development Process. In: *BITWorld '99*. Cape Town, South Africa, 30th June – 2nd July 1999.
- Wesson, J.L. (1999b). Usability Pattern Language: Creating a Community : Position Paper. In: *Workshop at INTERACT '99*. Edinburgh, Scotland, UK, 30-31 August 1999.
- Williamson, C., Shneiderman, B. (1992). The Dynamic HomeFinder: Evaluating Dynamic Queries in a RealEstate Information Exploration System. University of Maryland Computer Science Dept. Technical Report; #CS-TR-2819.
- Wise, J.A. *et al.* (1995). Visualizing the Non-Visual: Spatial Analysis and Interaction with Information from Text Documents. In: *IEEE Information Visualization '95*. Atlanta, Georgia, USA, 20-21 October 1995. pp 51-58.
- Zhou, M.X. and Feiner, S.K. (1996). Data Characterization for Automatically Visualizing Heterogeneous Information. In: *IEEE Symposium on Information Visualization*. San Francisco, California, USA, 28-29 October 1996. pp 13-20.
- Zhou, M.X. and Feiner, S.K. (1998). Visual Task Characterization for Automated Visual Discourse Synthesis. In: *Conference on Human Factors in Computing Systems*. Los Angeles, California, USA, 18-23 April 1998. pp 392-399.