Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage. http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form and Deposit Licence.
PURPOSEFUL VISUALIZATION SYSTEMS

XIAOYAN BAI

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Systems

THE UNIVERSITY OF AUCKLAND

2015
ABSTRACT

Human decision making by nature is predominantly a cognitive process conducted in an individual’s mind consciously or subconsciously to select an optimal solution from multiple alternatives for a certain decision goal based on pre-defined evaluation criteria. It is significantly influenced by the individual’s ability to acquire situation awareness and form mental models of the interested decision problem. With the rapid advances in the domain of information visualization, many visualization techniques and systems have been widely adopted to amplify situation awareness, and to inform, shape, change and/or reinforce mental models and behaviors of people. They do so by supporting the development of visualizations that fulfill particular purposes under certain contexts. Typical visualization purposes are to address issues within a diverse range of application domains, support user activities involved in information discovery and interpretation and decision making, and express thoughts/feelings/emotions on life, world, issues, etc. The effectiveness of fulfilling such purposes can be largely affected by both the decisional problem context within which the visualizations are deployed and the situational context of their stakeholders.

A key feature different visualizations share in common is that visualization purposes, contexts and stakeholders may vary over time and domain. Such changing facets of visualizations often result in dynamic changing visualization requirements. Moreover, visualization problems nowadays often incorporate increasingly high complexities that may emerge from enormous data volume, complex data types, the need for integrating multiple visualizations, and/or multiple paradigms/domains involved. These issues significantly exacerbate the difficulties involved in the development of effective visualization solutions. Dealing with such complex problems often requires visualizations to be developed in a manner that they can be flexibly created, instantiated, manipulated, customized, integrated, executed and modified. Though many existing visualization techniques and systems tend to provide reasonable support for particular paradigms, domains and data types, they are weak when it comes to (1) addressing the complexities involved in the problems and (2) supporting the ever-changing purposes, contexts and stakeholders, in a manner that sustains visualization effectiveness.
To address the above problems, issues and requirements, we suggest that visualizations be viewed from the intertwined triumvirate perspectives of purpose, context, and stakeholder. We introduce purposeful visualizations to encapsulate these concepts. We define purposeful visualizations as visualizations that fulfill a particular purpose for one or more stakeholders within a certain context. The concept of purposeful visualizations highlights the status of a visualization with regard to its visual capability of accommodating an intended purpose; it also accentuates the changing nature of visualizations caused by stakeholder, purpose, and context. Every visualization can become purposeful under a certain context but may fail to retain its purposefulness and effectiveness when the context is changed. The objectives of our research are to (1) formally define the concept of purposeful visualizations, (2) propose a model that facilitates understanding, creating and evaluating purposeful visualizations, (3) develop a process to guide the application of purposeful visualizations for addressing various visualization purposes in the context of decision making, (4) design purposeful visualization system frameworks and architectures to support the creation, instantiation, modification, execution, integration, transformation and adaptation of visualizations, (5) validate these concepts, models, processes, frameworks and architectures by implementing a prototypical purposeful visualization system that can be used to create purpose-driven, context-sensitive, stakeholder-relevant visualizations, and finally (6) demonstrate the functionalities and capabilities of the prototype through a sequence of real world scenario-driven illustrations drawn from the utility sector.
ACKNOWLEDGEMENTS

Our research on purposeful visualizations and purposeful visualization systems has been like a wondrous journey with many exciting findings and rewards along the way. It not only allowed me to learn how to conduct academic research, but also helped me to understand how to apply research findings to business environments. This journey would never be this great without my respectable research supervisors and advisors.

I would like to give my deepest appreciation to my supervisor – Dr. David Sundaram who is my great Master Shifu, best friend, and fantastic mentor. He is always patient with me. He never cared about spending long hours on checking my prototypes, papers and thesis. He always encourages me to pursue academic excellence. I would never forget the late nights that we spent together on polishing the first conference paper in my life and the joyful moment when we received the notification of paper acceptance. Without David’s valuable contributions, timely advice, continuous guidance and kind encouragement, I could have never got this far.

I really appreciate the amazing support that my great supervisor – David White has given to me. I am so grateful for the opportunities he has offered me to develop my programming skills. These skills enabled me to complete the whole implementation of a fully-fledged purposeful visualization system on my own. I would particularly like to thank my fantastic advisers - Dr. Claire Donald and Dr. Michael Cooling for their valuable feedback and suggestions.

I would like to express my heartfelt thanks to my beloved parents for their continuous encouragement throughout my life. It is their love that has led me to become a determined hard worker. At the end, I would like to thank my best friend and fiancé – Hao Yang. Thanks indeed for accompanying me in this journey and always making delicious Irish Cream coffee for me!
# OVERVIEW OF CONTENTS

ABSTRACT ................................................................................................................................. i

ACKNOWLEDGEMENTS ........................................................................................................ iii

OVERVIEW OF CONTENTS ..................................................................................................... iv

TABLE OF CONTENTS ............................................................................................................. v

CHAPTER 1  INTRODUCTION ..................................................................................................... 1

CHAPTER 2  LITERATURE REVIEW .......................................................................................... 12

CHAPTER 3  RESEARCH METHODOLOGY .............................................................................. 62

CHAPTER 4  PURPOSEFUL VISUALIZATIONS .......................................................................... 82

CHAPTER 5  PURPOSEFUL VISUALIZATION SYSTEMS ........................................................... 95

CHAPTER 6  PURPOSEFUL VISUALIZATION ARTEFACTS APPLICATION ............................... 110

CHAPTER 7  PURPOSEFUL VISUALIZATION SYSTEM IMPLEMENTATION ............................ 132

CHAPTER 8  PURPOSEFUL VISUALIZATION ARTEFACTS EVALUATION ............................... 168

CHAPTER 9  CONCLUSION ...................................................................................................... 197

APPENDICES .......................................................................................................................... 207

REFERENCES ......................................................................................................................... 255
# Table of Contents

**Abstract** ................................................................................................................................. i

**Acknowledgements** .................................................................................................................. iii

**Overview of Contents** .............................................................................................................. iv

**Table of Contents** ...................................................................................................................... v

## Chapter 1 Introduction ................................................................................................................ 1

1.1 Information Visualization for Decision Making ................................................................. 1
1.2 Challenges with Visual Decision Support ........................................................................... 3
1.3 Purposeful Visualizations & Purposeful Visualization Systems ....................................... 6
1.4 Research Domain .................................................................................................................. 8
1.5 Research Questions and Objectives .................................................................................... 9
1.6 Research Contributions ....................................................................................................... 10
1.7 Structure of the Thesis ....................................................................................................... 10

## Chapter 2 Literature Review .................................................................................................... 12

2.1 Decision Making .................................................................................................................. 13
2.1.1 Decision Problems, Paradigms and Genres ............................................................... 13
2.1.2 Rational Decision Making .......................................................................................... 15
2.1.3 Naturalistic Decision Making ..................................................................................... 17
2.2 Visualization for Decision Making ..................................................................................... 20
2.2.1 Overview of Information Visualization .................................................................... 20
2.2.2 Impacts of Information Visualization on Decision Making .................................... 22
2.2.3 Challenges with Visual Decision Support .................................................................. 24
2.3 Visualization Stakeholders .................................................................................................. 26
2.3.1 Stakeholder Overview ............................................................................................... 26
2.3.2 Human Information Processing ................................................................................... 28
2.3.3 Human Skill Acquisition ............................................................................................. 30
2.3.4 Bottlenecks in Human Decision Making Performance ............................................. 31
2.4 Visualization Purposes ........................................................................................................ 32
2.4.1 Domain-Related Purposes ......................................................................................... 33
2.4.2 Task-Related Purposes ............................................................................................. 34
2.4.3 Challenges with Purpose Alignment .......................................................................... 36
2.5 Visualization Contexts ......................................................................................................... 36
2.5.1 Context Complexities ................................................................................................. 37
2.5.2 Impact on Visualization Effectiveness ....................................................................... 40
2.5.3 Challenges with Addressing Context Complexities ................................................. 40
2.6 Visualization Systems .......................................................................................................... 41
2.6.1 Overview of Visualization Systems ........................................................................... 42
2.6.2 Information Representation Support .......................................................................... 43
2.6.3 Information Presentation Support ................................................................................ 46
<table>
<thead>
<tr>
<th>Paragraph</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.4</td>
<td>Challenges with Supporting Visualization Process</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>2.7</td>
<td>Problems, Issues and Requirements</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>2.7.1</td>
<td>Visualization-Related</td>
<td>52</td>
<td></td>
</tr>
<tr>
<td>2.7.2</td>
<td>Stakeholder-Related</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>2.7.3</td>
<td>Purpose-Related</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>2.7.4</td>
<td>Context-Related</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>2.7.5</td>
<td>System-Related</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>2.7.6</td>
<td>Summary</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CHAPTER 3</strong> RESEARCH METHODOLOGY</td>
<td>62</td>
</tr>
<tr>
<td>3.1</td>
<td>Methodological Requirements</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>A Review of Multi-methodological Research Support</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>3.2.1</td>
<td>Nunamaker et al. (1991)</td>
<td>67</td>
<td></td>
</tr>
<tr>
<td>3.2.2</td>
<td>March and Smith (1995)</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>3.2.3</td>
<td>Hevner et al. (2004)</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>3.2.4</td>
<td>Adams and Courtney (2004)</td>
<td>71</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>An Integrated Multi-Methodological Research Approach</td>
<td>72</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>The Instantiated Research Framework</td>
<td>75</td>
<td></td>
</tr>
<tr>
<td>3.4.1</td>
<td>Observation</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>3.4.2</td>
<td>Theory Building</td>
<td>78</td>
<td></td>
</tr>
<tr>
<td>3.4.3</td>
<td>System Development</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>3.4.4</td>
<td>Experimentation</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CHAPTER 4</strong> PURPOSEFUL VISUALIZATIONS</td>
<td>82</td>
</tr>
<tr>
<td>4.1</td>
<td>A Definition of Purposeful Visualization</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>4.1.1</td>
<td>Stakeholder</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>4.1.2</td>
<td>Purpose</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>4.1.3</td>
<td>Context</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>4.1.4</td>
<td>Visualization</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>4.1.5</td>
<td>Effectiveness of Purpose Fulfilment</td>
<td>87</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Purposeful Visualization Model</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>4.2.1</td>
<td>Influential Factors of Purposefulness</td>
<td>88</td>
<td></td>
</tr>
<tr>
<td>4.2.2</td>
<td>Purposeful Visualization Model</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td>4.2.3</td>
<td>PV Model Implications</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>Purposeful Visual Decision Making Process</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>4.3.1</td>
<td>Rational Process</td>
<td>91</td>
<td></td>
</tr>
<tr>
<td>4.3.2</td>
<td>Naturalistic Process</td>
<td>93</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CHAPTER 5</strong> PURPOSEFUL VISUALIZATION SYSTEMS</td>
<td>95</td>
</tr>
<tr>
<td>5.1</td>
<td>Distinguishing Features of PVS</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>5.1.1</td>
<td>Information Generation</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>5.1.2</td>
<td>Information Representation</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>5.1.3</td>
<td>Information Presentation</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>PVS Frameworks</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>5.2.1</td>
<td>Component Oriented Structural Framework</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 6  PURPOSEFUL VISUALIZATION ARTEFAC TS APPLICATION ........................... 110

6.1  Case Background ........................................................................................................ 111
6.2  Decision Problems ...................................................................................................... 112
6.3  Support for Diverse Visualization Stakeholders, Purposes and Contexts .................. 114

6.3.1  Scenario 1: Operational-Level Network Fault Monitoring .................................. 114
6.3.2  Scenario 2: Managerial-Level KPI Monitoring ...................................................... 115
6.3.3  Scenario 3: Power outage impact investigation ...................................................... 117

6.4  Support for Sustaining Visualization Effectiveness .................................................... 118

6.4.1  Scenario 1: Problem Context Adaptability ............................................................... 118
6.4.2  Scenario 2: Purpose Context Adaptability ............................................................... 120
6.4.3  Scenario 3: Stakeholder Context Adaptability ......................................................... 121
6.4.4  Scenario 4: Time Context Adaptability .................................................................... 122

6.5  Support for Visual Decision Making through Flexible Presentations ......................... 123

6.5.1  VDM – Intelligence .................................................................................................... 124
6.5.2  VDM – Design .......................................................................................................... 125
6.5.3  VDM – Choice and Implementation ......................................................................... 128

CHAPTER 7  PURPOSEFUL VISUALIZATION SYSTEM IMPLEMENTATION ..................... 132

7.1  Overview of the Purposeful Visualization System Prototype ..................................... 132
7.2  Prototype Development ............................................................................................... 135

7.3  Information Generation ............................................................................................... 137

7.3.1  Data Source Management ....................................................................................... 137
7.3.2  IG Model Management ........................................................................................... 142

7.4  Information Representation .......................................................................................... 145

7.4.1  Template IR Models .................................................................................................. 146
7.4.2  IR Model Management ............................................................................................. 147
7.4.3  Transform Representation ......................................................................................... 150
7.4.4  Integrate Representations ........................................................................................ 152

7.5  Information Presentation .............................................................................................. 156

7.5.1  Template IP Models ................................................................................................. 156
7.5.2  IP Model Management ............................................................................................ 161
7.5.3  Transform Presentation ............................................................................................ 165
7.5.4  Integrate Presentations ............................................................................................ 166

CHAPTER 8  PURPOSEFUL VISUALIZATION ARTEFACTS EVALUATION .................... 168

8.1  Purposeful Visualization – Definition & Model ............................................................. 169

8.1.1  Beautiful Evidence Seminar ...................................................................................... 170
8.1.2  Seminar Data Analysis ............................................................................................. 171
8.1.3  Discussion ................................................................................................................. 173
8.1.4  Evaluation by Experts ............................................................................................... 178
8.2 Purposeful Visual Decision Making Process ............................................................... 178
  8.2.1 Evaluation by Prototyping ....................................................................................... 179
  8.2.2 Evaluation by Case Study ...................................................................................... 180
  8.2.3 Evaluation by Illustrative Scenarios ....................................................................... 180
  8.2.4 Evaluation by Experts ............................................................................................ 181
8.3 Purposeful Visualization Systems – Definition & Distinguishing Features ............. 182
  8.3.1 Evaluation by Prototyping ....................................................................................... 183
  8.3.2 Evaluation by Case Study ...................................................................................... 184
  8.3.3 Evaluation by Illustrative Scenarios ....................................................................... 185
  8.3.4 Evaluation by Experts ............................................................................................ 186
8.4 Purposeful Visualization Systems – Frameworks & Architectures ......................... 187
  8.4.1 Evaluation by Architecture Analysis ...................................................................... 188
  8.4.2 Evaluation by Prototyping ...................................................................................... 188
  8.4.3 Evaluation by Experts ............................................................................................ 189
8.5 The Purposeful Visualization System Prototype ......................................................... 190
  8.5.1 Evaluation by Static Analysis .................................................................................. 190
  8.5.2 Evaluation by Functional Testing .......................................................................... 191
  8.5.3 Evaluation by Structural Testing ............................................................................ 192
  8.5.4 Evaluation by Computer Simulations ..................................................................... 192
  8.5.5 Evaluation by Case Study ....................................................................................... 192
  8.5.6 Evaluation by Illustrative Scenarios ....................................................................... 193
  8.5.7 Evaluation by Informed Argument ....................................................................... 193
  8.5.8 Evaluation by Experts ............................................................................................ 194

CHAPTER 9 CONCLUSION ................................................................................................. 197
  9.1 Summary of Research ............................................................................................... 197
  9.2 Contributions ............................................................................................................ 200
  9.3 Limitations and Future Research .............................................................................. 206

APPENDICES ..................................................................................................................... 207

Appendix A: Interface Layer .......................................................................................... 207
Appendix B: Application Layer – Visual Scenario Manager ........................................... 213
Appendix C: Application Layer – Information Generation ............................................. 216
  C.1 Data Source Manager ............................................................................................... 216
  C.2 Information Generation Manager ............................................................................. 218
Appendix D: Application Layer – Information Representation ...................................... 220
Appendix E: Application Layer – Information Presentation ......................................... 226
Appendix F: Resource Layer ........................................................................................... 231
  F.1 Data Pool .................................................................................................................. 232
  F.2 Model Pool ............................................................................................................... 233
  F.3 Solver Pool ............................................................................................................... 248
  F.4 Scenario Pool .......................................................................................................... 253

REFERENCES .................................................................................................................... 255
CHAPTER 1 INTRODUCTION

1.1 Information Visualization for Decision Making

Decision making is an integral part of every human activity in life. It is predominantly a cognitive process conducted in an individual’s mind consciously or subconsciously to select an optimal solution from multiple alternatives for addressing a certain problem. It is significantly affected by the individual’s ability to acquire situation awareness and form mental models of the interested decision problem. As human beings are highly attuned to visual information (e.g. images, charts and media), this naturally raises requirements for effective representations and presentations of the information, which is relevant to (1) the decision problem, (2) alternative solutions and trade-offs, and (3) decision contexts and purposes. With the rapid advances achieved in the field of information visualization, many visualization techniques and applications have been developed to support the requirements for decision making.

Information visualization can be broadly defined as any processes of transforming information into mental visual images (Spence, 2007). A variety of computer or non-computer based techniques have been developed to support such processes, which, in turn, produces many different visualization genres, such as cave paintings, oil paintings, hand drawings, sculptures, statistical visualizations, geographical visualizations, hybrid visualizations, etc. Our research has a particular focus on computer-aided information visualization. Reviews on representative visualization techniques and applications have been conducted by many researchers such as Heer, Bostock and Ogievetsky (2010), Spence (2007), Chen (2005), Chi (2000), and Card, Mackinlay and Shneiderman (1999).

Visualization techniques and systems are deployed in many domains/disciplines where people need to explore, get insights from, and/or interpret data. Card et al. (1999) outline seven typical application domains of visualization, namely, statistical and categorical data management, digital library management, personal services support, complex documents management, history management, classifications management, and networks management. Though they can be used as separate visualization systems, visualization techniques and
applications are often integrated into decision support systems to provide a visualization support environment. For example, bar, line, bubble or stack charts can be used to compare values. Scatter, matrix or network diagrams can show relationships among data points. Tag cloud and treemap can help with comparing proportions. Maps can support presenting spatial data. These visualization techniques are popular in systems that support decision making and/or information visualization, e.g. Microsoft business intelligence (BI) products (SQL Server Data Tools, Excel, and SharePoint), Google Analytics, SAP BI products (Data Warehouse, BusinessObjects, Crystal Reports and Xcelsius), SAS Visual Analytics and Enterprise BI, IBM Cognos, Tableau, etc.

Visualization techniques and systems may assist people with fulfilling a variety of purposes, for example, supporting user tasks and activities involved in information navigation, retrieval, query, discovery and interpretation, and expressing personal thoughts/feelings/emotions on life, world, love, etc. The ultimate goal of information visualization is to amplify and attenuate the cognitive strengths and weaknesses of their users respectively. When visualizations are appropriately deployed for their intended purpose, quite often their users can obtain illuminating, useful, relevant and actionable information from the visualizations with less effort and time. Card et al. (1999) identify several important types of support from effective visualizations: useful data transformations, reduced search workload, enhanced user abilities of information detection, improved monitoring and perceptual inference abilities of users, flexible interactions with data, and increased memory and processing resources available to users. Misleading or wrong visualizations can be costly and waste resources and time.

Effective visualizations may amplify human perception and lead to better visibility and more insights of data (Adnan, Noor & Aripin, 2003). To ensure that visualizations are effective and fit for purpose, three fundamental steps in the process of information visualization should be well supported (Figure 1.1). These steps are information generation (generating the data required to support/solve user problems), information representation (visually encoding the data in such a way that users’ perception and cognitive activities can be augmented), and information presentation (presenting the encoded data with appropriate interaction mechanism in order to enable the users to amplify cognition with less effort and time). How well the requirements of each information visualization step are addressed is crucial to
effectively informing, shaping, changing and reinforcing mental models and behaviors of people.

![Information visualization process](image)

Despite the advances achieved in visualization technologies, supporting the entire visualization process to develop purposeful effective visualizations is still rather challenging to accomplish. We outline the challenges associated with this in the following section.

### 1.2 Challenges with Visual Decision Support

Visual decision support in essence aims to leverage the power of visualization technologies to support various decision making activities by helping people to acquire, understand and interpret information with less effort and time (Bai, White & Sundaram, 2009b). We explore the challenges with visual decision support from five perspectives: visualization, stakeholder, purpose, context, and system. A summary of the major problems and issues relating to each perspective is provided in Figure 1.2.

**Visualization-Related Challenges**

Visualization problems nowadays become rather challenging to address due to the complexities caused by (1) high data complexities (Santos & Brodlie, 2004), (2) enormous data volumes (Santos & Brodlie, 2004; Jern, 1999; Saffer et al., 2004), (3) dynamic visualization paradigms associated with gradual or abrupt changes/trends/patterns emerging over time (Chen, 2005; Bai, White & Sundaram, 2011c), (4) visualization misuse, and (5) requirements for visualization quality and aesthetics. The complexities involved in visualization problems often bring many challenges to data navigation, exploration and interpretation (Jern, 1999). Traditional visualizations (such as 2D or 3D statistical charts), even presented in interactive formats, are still ineffective and inadequate to address them (Saffer et al., 2004).
There are two prominent features that different visualizations share in common, regardless of the diversity of visualizations. First, they are all created with a particular purpose for one or more stakeholders within a certain context. Second, their purpose, context and stakeholders may change over time. Effective visual decision support requires dealing with the challenges posed by the changing nature of visualizations (Bai, White & Sundaram, 2011c) and understanding visualization fit (Chen, 2005). More specifically, visualization effectiveness reflects how well a visualization can help its stakeholders to address their decisional problem(s) and fulfill their intended purpose(s). It is widely recognized that visualization effectiveness...
effectiveness is capricious, subjective, complicated and context-dependent in nature. Visualizations that are effective for one stakeholder may not be effective for other stakeholders. Visualizations that are effective for their stakeholders at one point in time may not be able to maintain their effectiveness at other times. Visualizations that are effective for one decision problem may not be applicable for other problems. These uncertainties imply that the effectiveness of a visualization is only valid within a certain context and may change when its context varies. In other words, there is no universal result for the same visualization in terms of its effectiveness.

Furthermore, visual decision support is confronted with the challenges of the intrinsic limitations of human cognitive capacity and information processing ability. The decision making ability of an individual is heavily affected by his cognitive capacity, which, in turn, is dependent upon the ability of processing information in his memory systems. However, due to the limited capacity and duration (Atkinson and Shiffrin, 1968), short-term memory has become the biggest bottleneck in human information processing (Adnan, Noor & Aripin, 2003). This requires visualizations to fit for the cognitive styles and preferences of the individual and present the underlying data into patterns preferred or already familiarized by the individual (Carroll & Zeller, 2012). Moreover, visualization design and development should take into account the need of adequate system and domain knowledge for the individual to manipulate and interpret visualizations.

**System-Related Challenges**

To address or relieve the above challenges with visualization, stakeholder, purpose and context, decision support systems require purposeful effective visual representations and presentations of the relevant data. Accordingly, they often need to include visualization subsystems or be used together with separate visualization systems that support users to flexibly create, instantiate, manipulate, customize, transform, integrate and dispose visualizations. Such system requirements are not only indispensable for developing purposeful effective visualizations but also important to deal with the challenges of maintaining visualization effectiveness under dynamic and changing contexts.
However, while many existing visualization techniques and systems tend to provide reasonable support for particular paradigms, domains, and data types, they are weak when it comes to supporting such requirements pertinent to flexibility and coping with the high complexities involved in visualization problems. They often fail to address the complexities involved in multi-paradigm, multi-domain problems that deal with complex spatio-temporal multi-dimensional data. This has led to visualizations that are context insensitive, data dense, and sparse in intelligence.

Additionally many existing visualization techniques and systems fail to support the diversity and changes of visualization stakeholders, purposes and contexts. They may focus more on how to visually encode and present certain data types, but less on adapting to the changing decision problem context where visualizations are applied, the situational contexts of visualization stakeholders and the corresponding changes in visualization requirements. The lack of concern on the variety of visualization stakeholders, purposes and contexts may result in impaired effectiveness, limited usage and even the misuse of visualizations.

Furthermore, visualization system designs and implementations should explicitly consider the impact from the underlying visualization contexts. An effective visualization should not only solve a problem for one or more decision makers and fulfill their intended purpose at a certain point in time, but also allow them to adapt the visualizations to accommodate context changes whenever needed.

To address the above challenges, we propose purposeful visualizations and implement purposeful visualization systems. They are briefly introduced in the following section.

1.3 Purposeful Visualizations & Purposeful Visualization Systems

Purposeful visualizations (PV) are visualizations that fulfill a particular purpose for one or more stakeholders within a certain context. The concept of PV is not introduced as a new category of visualization, but it intends to highlight the importance of context and purpose when we design, implement, apply and evaluate visualizations. This concept aims to reveal the capability of a visualization in terms of how well it fits for the intended purposes and requirements. It also emphasizes the changing nature of visualizations, which is caused by
stakeholders, purposes and contexts. Every visualization may become purposeful and effective under a certain context, but its purposefulness may vary when the related stakeholders, purposes and contexts are changed.

Typical stakeholders are visualization designers who develop, manage and evaluate PV, viewers who apply PV to aid diverse visualization related tasks, and presenters who present PV to inform/shape/change/reinforce mental models and behaviors of the viewers. To achieve the intended purposes, PV should provide illuminating, useful, relevant and actionable information for stakeholders.

To facilitate the development and application of PV, we introduce a new concept of Purposeful Visualization Systems (PVS). PVS are flexible, interactive and adaptive platforms that enable people to fulfill their intended visualization purposes and requirements by creating and managing effective visual compositions with high visual intelligence density. The concept of PVS is inspired by the classical designs of decision support systems (Brennan & Elam, 1986; Ramirez, Ching & Louis, 1990; Sundaram, Chen & Srinivasan, 2001), information visualization reference models (Haber & McNabb, 1990; Card et al., 1999; Card & Mackinlay, 1997; Chi & Riedl, 1998; Spence, 2007; Bai, White & Sundaram, 2010a), the ideas of scenario planning (Schoemaker, 1993; Chermack, 2004; Chermack, 2005; Avin, 2007; Keough & Shanahan, 2008), and the concepts of model management lifecycle. PVS deploy the four fundamental system building blocks (i.e. data, models, solvers and scenarios) to realize information generation, representation and presentation. Such design enables PVS to accommodate the system requirements for flexibly developing purposeful effective visualizations and sustaining visualization effectiveness when stakeholders, purposes and contexts change over time.

A purposeful visualization system distinguishes itself from other visualization support environments by offering the following key features:

- Enable users to choose how they would like to extract, visually encode and present their data
- Support flexibly creating, modifying and reusing visualizations
- Support flexibly transforming visualizations from one type to another
- Enable users to flexibly integrate various visualizations created from different sources
- Allow users to adapt visualizations to stakeholder/purpose/context changes and sustain visualization effectiveness
- Interweave individual visualizations to create vivid stories

This introduction of PV and PVS lays a basis for later clarifying our research domain in section 1.4 and formulating our research questions and objectives in section 1.5. The concepts of PV and PVS will be further expanded in detail in chapters 4 and 5 respectively.

### 1.4 Research Domain

Our research focuses on the domain of purposeful visualization of information, which involves research subjects about information, visualization and purposefulness (Figure 1.3). The subject of information relates to the research discipline of information systems. Visualization is concerned with the activities and processes of creating and interpreting certain visual images/forms of data. It is a typical inter-disciplinary research area (Chen, 2010), which integrates information systems, computer science, psychology, etc. The subject of purposefulness highlights the capability of a visualization technique/system in terms of fulfilling the intended purposes and requirements under a certain context. It is intimately associated with disciplines such as management and decision making.

![Figure 1.3 Research domain](image-url)
1.5 Research Questions and Objectives

Our research aims to (1) empower people with the ability to flexibly design and develop purposeful effective visualizations, (2) enhance the support for the changing visualization requirements caused by the diversity and dynamics of visualization stakeholders, purposes and contexts, and (3) enrich visualizations by providing illuminating, useful, relevant and actionable information for decision making. To achieve these goals, our research particularly explores the following questions:

- What are purposeful visualizations?
- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?
- How can we apply purposeful visualizations to support various tasks in decision making?

Accordingly, the essential objectives of our research are defined as follows:

- Formally define the concept of purposeful visualizations
- Propose a model that facilitates understanding, creating and evaluating purposeful visualizations
- Develop a process to guide the application of purposeful visualizations for addressing various visualization purposes in the context of decision making
- Design purposeful visualization system frameworks and architectures to support the creation, instantiation, manipulation, integration, execution, evaluation and modification of visualizations
- Validate these concepts, models, processes, frameworks and architectures by implementing a prototypical purposeful visualization system that can be used to create purpose-driven, context-sensitive, stakeholder-relevant visualizations
- Demonstrate the functionalities and capabilities of the prototype through a sequence of real world scenario-driven illustrations drawn from the utility sector

Fulfilling the above research objectives allows us to deliver a series of theoretical and practical contributions to the fields of information visualization and decision support systems. Our research contributions are briefly outlined in the subsequent section.
1.6 Research Contributions

Our research on purposeful visualizations and purposeful visualization systems benefits not only the field of information visualization but also potentially any research domains that require purposeful effective visualizations. Such domains may involve decision support systems, knowledge management systems, business intelligence systems, geospatial information systems, health care systems, simulation/optimization systems, context-aware systems, etc. In particular, the exploration, concepts and model of PV contribute to reveal the complexities involved in visualization problems and contexts. They can help researchers and practitioners to better understand and deal with the changing visualization stakeholders, purposes and contexts and the corresponding visualization requirement changes. This, in turn, may aid in (1) the evaluation of visualization effectiveness and (2) selecting/designing visualizations that are effective to support various tasks in information discovery, decision making and information interpretation.

Furthermore, the concepts, frameworks, architectures and implementations of PVS can be used to guide the design and development of flexible interactive visualization environments. Such visualization environments can be standalone visualization systems, or visualization subsystems to be embedded in certain domain-specific systems that require purposeful effective visualizations. The research artefacts of PVS also demonstrate a generic approach to flexible visualization creation, modification, customization, integration, transformation and adaptation.

1.7 Structure of the Thesis

After this introductory chapter, this thesis is organized as follows. In chapter 2, we review and synthesize the extant literature pertaining to the domain of purposeful visualizations of information, and identify the focal problems, issues and requirements involved in this domain. Keeping our research requirements in mind, we propose and discuss an integrated multi-methodological research framework that was adopted to accomplish the research objectives and validate our research artefacts in chapter 3. We then proceed to introduce the formal definition, model and process to aid the understanding, evaluation and application of PV in
chapter 4. Next, we propose models, frameworks and architectures of PVS in chapter 5, which guide the implementation of a visualization environment to facilitate development and application of purposeful visualizations. To validate the proposed theory of PV and PVS, we implemented a fully-fledged prototypical purposeful visualization system. The application of the prototype is demonstrated through a sequence of scenario-driven visualizations in chapter 6, followed by detailed illustrations of the prototype implementation and its key system features in chapter 7. After this, we explicate how the research artefacts of PV and PVS were validated to ensure the rigor of our research in chapter 8. Finally, chapter 9 concludes the whole thesis with outlining the contributions, limitations and future research directions of our research.
CHAPTER 2   LITERATURE REVIEW

The primary purpose of information visualization is to support decision making. It aims to leverage the power of visualizations to better define decision problems, develop alternative solutions with enhanced visual intelligence, and enlighten decision makers on the trade-offs among the alternatives. By reviewing and synthesizing the extant literature on visual decision support, we identified six fundamental pillars of this research area: **decisions**, **stakeholders**, **purposes**, **contexts**, **visualizations** and **systems**. We hereby organize the literature review around these pillars (Figure 2.1). This review not only highlights the gap our research attempts to bridge or narrow, but also constitutes an essential foundation for our proposed research artefacts in later chapters.

![Framework of literature review](image)

**Figure 2.1   Framework of literature review**

This chapter is structured as follows. We examine the principal decision making genres and their representative decision models in section 2.1, followed by an overall review of the essentials of deploying information visualization to support decision making in section 2.2. We proceed to synthesize the literature from the intertwined triumvirate perspectives of stakeholder, purpose, and context in sections 2.3-2.5 respectively. While identifying the challenges associated with each review area, we explore and evaluate the state of the art of the support offered by existing representative visualization techniques and systems in section
2.6. By analyzing the strengths and weaknesses of the available system support towards addressing the challenges, we articulate the research gap through outlining the focal problems, issues and requirements in section 2.7.

2.1 Decision Making

Decision making by nature is a cognitive process conducted in a decision maker’s mind consciously or subconsciously to select a preferred solution from multiple alternatives for a certain decision goal based on a set of pre-defined evaluation criteria or strategies (Wang & Ruhe, 2007). It is significantly influenced by a decision maker’s ability to acquire situation awareness and form mental models of the interested decision problem (Niu, Lu & Zhang, 2009). Information visualization is introduced into the field of decision support to amplify situation awareness and reinforce mental models. With appropriate visualizations, a decision maker can acquire, understand and interpret information with less effort and time. With a better awareness of the decision situation and richer cognitive maps of the decision problem, the decision maker tends to possess more chances to take a good decision. In this section, we provide an overview of decision making in subsection 2.1.1 and proceed to examine the principal decision paradigms in subsections 2.1.2-2.1.3.

2.1.1 Decision Problems, Paradigms and Genres

Decision making is concerned with activities or processes that mitigate the gap between the existing and desired problem situations. When there are multiple alternative solutions to resolve this gap, a decision problem arises (Grüning, Kühn, Clark & O’Dea, 2005). Decision problems nowadays range (1) from simply making a choice among multiple known solutions to designing a new solution specifically to solve the underlying problem, (2) from reactively resolving a threat to proactively seeking an opportunity, and (3) from making a single independent decision to resolve a well-structured problem to taking a series of parallel/sequential decisions to address an ill-structured problem.

We explore different types of decision problems below, from the simplex to the complex. Decision problems can be as simple as which movie to go to or which dress to wear or where
to holiday. However, when we consider the business world the number of factors and steps involved in a single decision grows exponentially as can be envisaged in the following decisions facing electricity distribution businesses. Selecting a spot for a new distribution substation from the available vacant areas is a choice-type decision problem. In comparison, building the substation is a design type problem. It can be decomposed into a series of sub decision problems, which should be solved sequentially or in parallel. For instance, what substation layout plan is appropriate to use with considering the future network load and development of the local area, the safety of operational staff and the public? What types and models of electrical equipment should be installed, e.g. switchgears, transformers, low voltage boards, capacitor banks and so on? What earthing design should be applied to the equipment? What model of meters should be selected to measure and monitor high voltage supply? What ventilation system design is required?

To aid in addressing various decision problems, extensive research and studies have been conducted to formulate useful decision paradigms and processes. Some decision making models are generic and domain independent such as Simon’s (1960) model, while others set their focus on particular application domains by considering the domain related decision making challenges and characteristics. For example, Gao, Paynter and Sundaram’s (2004) model concentrates on addressing decision problems with geospatial features. Moreover, they span from normative models aiming to address well-structured problems, such the extended decision making process (Mintzberg, Raisinghani & Theoret, 1976), to descriptive models supporting ill-structured problems, for instance, the garbage can model (Cohen, March & Olsen, 1972) and the linkage-driven decision making models (Langley, Mintzberg, Pitcher & Posada, 1995).

In general, decision making models spread across a spectrum on which models at one end are highly rational, normative and analytical and models at the other end are extremely irrational, descriptive, anarchic and intuitive. We thereby classify decision making models into two generic genres: rational decision making, and naturalistic decision making. Both genres are examined in subsections 2.1.2 – 2.1.3 respectively.
2.1.2 Rational Decision Making

Rational decision making attempts to model decision making activities and tasks into a highly structured format. It follows a logical process to understand a decision problem, define evaluation criteria, create alternative solutions, and assess and select a solution for implementation. For well-structured problems, rational decision making processes can often achieve best performance and yield best results (Schoemaker & Russo, 1993). Under the context of visual decision making, they may contribute to the understanding of processes and activities that need the support from visualizations. Representative decision making models of such type are Simon’s (1960) decision making model, and Mintzberg et al.’s (1976) extended decision making process.

Simon’s (1960) model is one of the oldest decision making models and has been widely acknowledged, which serves as fundamentals for decision making modelling by other researchers, such as Mintzberg et al. (1976), Rowe and Boulgarides (1994), and Marakas (2003). Simon’s (1960) model involves four essential stages: intelligence, design, choice and implementation. At the intelligence stage, a decision maker identifies a problem and potential opportunities to develop alternative solutions to solve it, and then sets criteria to evaluate alternatives. Based on the understanding of the problem, the decision maker develops a set of alternative solutions for evaluation at the design stage. Next, at the choice stage, alternatives are assessed against the pre-defined evaluation criteria and the most appropriate solution best fulfilling problem requirements is chosen for implementation. Finally, at the implementation stage, the selected solution is implemented for operation, which may lead to new problems and requirements and then trigger new iterations of the decision making process.

Building on top of Simon’s work (1960), Mintzberg et al. (1976) enriches Simon’s model with a set of general tasks identified by examining 25 decision making processes adopted by many organization. Example of such tasks are problem recognition and diagnosis, alternatives search and design, and judgment/analysis/bargaining evaluations. More specifically, a decision maker recognizes a problem under his decision environment or potential opportunities to improve the current situation. Then, he diagnoses the problem to define it properly, and searches available solutions or develop new alternatives if no existing solution
can be found. The decision maker evaluates the alternatives through judgment, analysis and bargaining to select the optimal solution, and obtains the approval from senior management for his decision.

Similar to Mintzberg et al.’s (1976) extended process, Rowe and Boulgarides’ (1994) five-stage decision making process is also built upon Simon’s (1960) model. Unlike Simon’s (1960) work, this process emphasizes the influences from decision makers on a decision making process by explicitly including a step of understanding a decision maker’s decision style and personal values and external pressures imposed on him. Moreover, Rowe and Boulgarides (1994) highlight the role of decision makers in terms of how they participate in every decision making step. Their decision making process starts with a decision maker receiving certain stimuli from the underlying decision environment, e.g. positive/negative externalities of a decision problem, or differences between the current and desired situations. It then progresses to the step of understanding characteristics and decision styles of the decision maker. For instance, how does the decision maker formulate evaluation criteria and weight up the importance of each criterion? What external pressures does the decision maker confront? How would such pressures affect the selection of alternative solutions? The subsequent problem definition, alternative selection, and solution implementation steps involved in the process are similar to the stages in Simon’s (1960) model.

**Critique**

Though rational decision making models can provide reasonably good support for well-structured problems, they are weak when it comes to solving semi-structured or ill-structured decision problems. An important reason is that these models lack the concern for personal characteristics, preferences, cognitive styles and capabilities of decision makers, and hence neglect their influences on the entire process of decision making. This is why the effectiveness of adopting purely rational side approaches to support decision making is challenged by many researchers such as Langley et al. (1995), Kuo (1998), Burke and Miller (1999), Nutt (1999), and Sinclair and Ashkanasy (2005). Though Rowe and Boulgarides’ (1994) model attempts to unravel certain decision complexities introduced by decision makers, it is insufficient to disclose the impact on decision making generated by intuition, emotion, insight, inspiration, imagination and cognitive capacities of decision makers. Nutt (1999) purports that decision
making following merely analytical style approaches can lead to high failure rates. In general, the rational side approaches tend to model decision making by simplifying the complexities and ignoring the influences on decision originated from human heuristics. This behooves us to examine the naturalistic decision making approaches.

2.1.3 Naturalistic Decision Making

Decision making is unstructured and complicated by nature (Vessey, 1991; Vessey & Galletta, 1991; Langley et al., 1995; Epstein et al., 1996; Sauter, 1999; Marakas, 2003). Naturalistic decision making studies the way of how people take decisions in real world situations with various constraints and uncertainties (Klein & Klinger, 1991; Bryant 2002). Such decision environments are normally featured by ill-defined goals and tasks, dynamic and changing conditions, significant consequences for mistakes, time pressure, etc. (Klein & Klinger, 1991; Bryant 2002). Naturalistic decision making explores uncertainties under such decision environments, and acknowledges the impact of intuition, insight, inspiration, emotion, memory, imagination, personal characteristics and preferences of decision makers.

More specifically, Langley et al. (1995) argue that decision making can be driven by inspiration as well as sequential/lateral/precursive linkages. Decision making performance is also influenced by factors like intuition, emotion, imagination and memories (Sauter, 1999; Marakas, 2003). For example, intuition is the analyses as a part of a decision maker’s intrinsic habit and capacity for fast recognition and response (Larkin & Simon, 1987). It enables the decision maker gain immediate insights into a problem situation without consciously following analytical decision making procedures (Sauter, 1999). Developing strong intuitive and judgmental skills is important for decision makers to better cope with uncertainties in the underlying decision environment and seize the opportunity to develop an appropriate solution to the decision problem (Langley et al., 1995; Basi, 1998). In addition, personal memories and organizational culture are two fundamental forms for retaining organizational memories (Walsh & Ungson, 1991), which help decision makers to learn from experience and take better decisions.
Different from rational decision making that relies on the development and evaluation of alternative solutions, naturalistic decision making focuses on recognizing and assessing potential courses of action based on pre-defined acceptance criteria (Lipshitz et al., 2001; Bryant, 2002). With naturalistic decision making, the selected solution to a problem does not necessarily be an optimized option but can merely meet the acceptance criteria so as to better alleviate the pressure on time (Klein & Calderwood, 1991).

The naturalistic side decision making approaches are irrational, descriptive, and heuristic. Langley et al.’s (1995) inspiration-driven and linkage-driven decision making models are good examples of this type. The inspiration-driven model highlights the importance and influences of inspirations insights obtained by a decision maker from investigating a decision problem. According to Langley et al. (1995), such inspirations and insights serve as effective stimuli to enable the decision maker to immediately gain more understanding of the problem situation and potential solutions, which, in turn, may lead to significant progress in decision making and problem solving. Under the context of visual decision making, this model implicates that visualizations are designed in a way to facilitate the achievement of inspirations and insights. A visualization should be able to highlight useful patterns and trends with the underlying data and enable necessary user interactions. Meanwhile, its design should also take into account cognitive styles and preferences of decision makers.

Compared to the inspiration-driven model, the linkage-driven model stresses the mutual influences among issue streams, decision making activities and decisions. In a complex decision environment, one decision problem may compromise several issue streams, each of which involves a series of decision making activities and decisions. Such decisions are not isolated but interrelated with or interdependent on each other, and can happen in sequential or parallel fashions, evolve over time, and eventually converge to or are interwoven into a final decision (Liew & Sundaram, 2009). Langley et al. (1995) identify three general types of linkage, that is, lateral linkage between streams, sequential linkage between decisions, and precursive linkages. Under the context of visual decision making, the linkage-driven model implies the need of diverse visualization techniques to support various issue streams, activities, decisions and possibly multiple decision makers with different visualization requirements. Visualization solutions should support decision makers interact with and share
visualizations if required. Furthermore, it deserves to be noted that visualizations selected to support one decision may serve as input to generating visualizations for other decisions.

Another representative model of naturalistic decision making is Klein’s (1989) recognition-primed decision model. This model starts with a decision maker assessing features of a decision problem and seeking familiar patterns or situations based on prior knowledge and experience. The recognition of a familiar scenario relies on the recognition of similar goals to achieve, cues to monitor, problem requirements to fulfil, and feasible action plans (Klein & Klinger, 1991). If a familiar scenario is recognized, known actions to the scenario will be carried out to solve the problem. If no familiar scenario can be identified, the decision maker diagnoses the problem by actively seeking more information and resolving ambiguous situations. This diagnostic process aims to create a story to explain the problem situation and define the expectancies of the unfolding of the problem (Klein, 1997). Then, the decision maker generates plausible courses of action and consciously evaluates their adequacies via mental simulation. It is recommended that the options of the action plan be produced and assessed serially (Klein & Klinger, 1991). Once a course of action can resolve the problem, the decision maker will carry out it without bothering if it is the optimal solution or not.

To better resolve real world decision problems, a special type of naturalistic decision making attempts to seamlessly integrate the strengths of both rational and naturalistic approaches. It aims to reconcile and support both the need of highlighting the processes and activities and that of disclosing the complexities and uncertainties in decision making (Eisenhardt & Zbaracki, 1992; Sinclair & Ashkanasy, 2005). For example, Sinclair and Ashkanasy (2005) propose an integrated decision making model that incorporates both analytical and intuitive approaches. The analytical approach aids the rational side information processing and cognition while the intuitive approach assists with the experiential side. Both approaches are harmoniously interconnected with each other. Sinclair and Ashkanasy (2005) also identify four groups of influential factors that may significantly affect decision making: problem characteristics, decision characteristics, personal disposition, and decision context.

**Critique**

Unlike the rational side decision making models that outline the essential tasks and processes involved decision making, the naturalistic side decision making approaches unveil the
complexities and uncertainties involved in decision making. Nonetheless, they often provide very little practical guidance on decision making and problem solving. As both rational and extreme anarchic approaches are flawed to some extent, decision making models and systems should support both decision making types to provide better support for real world decisions. As reflected by anarchic and naturalistic decision making paradigms, the performance of human decision making is determined by the perceptual and cognitive abilities and biases of people. Information visualization is specifically for amplifying their situation awareness and reinforcing their mental models, and thereby supporting decision making effectively. An overall review of visual decision support is provided in the following section.

2.2 Visualization for Decision Making

Visual decision support concerns the use of visualization technologies and tools to support decision making activities. Visualization can be broadly defined as any process that transforms information into appropriate visual forms in order to aid information discovery, decision making and interpretation and eventually amplify human cognition (Card et al., 1999; Spence, 2007; Griffin, 2009; Al-Kassab et al., 2013). In this section, we provide a high-level overview of information visualization in subsection 2.2.1, followed by explicating how information visualization supports decision making in subsection 2.2.2. Then, we highlight the challenges with achieving effective visual decision support in subsection 2.2.3.

2.2.1 Overview of Information Visualization

Information visualization, in essence, is a process of generating, representing and presenting information. The information generation stage is to ensure that the underlying source data are transformed and organized into a series of data sets with only data relevant to decision making and problem solving. At the information representation stage, the extracted data are encoded into appropriate visual structures. At the information presentation stage, the visual structures are further transformed into views that people can display, manipulate and interact with. The detailed reviews on techniques and systems that support information representation and presentation will be provided in section 2.6. No matter what visualizations
are eventually produced by this process, there are three critical facets shared in common: stakeholder, purpose and context. That is, visualizations are all created with a particular purpose for one or more stakeholders within a certain context. These facets will be explored and synthesized in later sections 2.3 – 2.5.

The above structured and logical process is well embodied in many visualization reference models. With a specific focus on information transformation, Card et al. (1999) propose an information visualization pipeline that highlights key transformation steps to turn information into a visual form, i.e. data transformation, visual mappings, and view transformation. In particular, raw data, which are relevant to the underlying decision problem and may come from a variety of sources, are initially converted into a format (i.e. data table) for the convenience of visual encoding. Then the data table is mapped to appropriate visual structures useful to support problem solving and decision making. The visual structure represents the source data through a series of spatial substrates, marks and their associated graphical properties. The same data table may be represented through multiple visual structures. One way to determine which visual structure is appropriate to choose is to assess its capability of carrying more distinctions, accuracy and support for interpreting the visual content. The selected visualization structure can then be displayed in various views through transformations such as location probes, viewpoint controls and distortions (Card et al., 1999). It is worth to note that this pipeline model fails to reveal how multiple data sources may contribute to the formation of an integrated visualization, which is commonly required in real world visualization processes.

Unlike the pipeline model, the table notation framework, introduced by Card and Mackinlay (1997), interprets visualization from fundamental building blocks involved in visualization development. This framework decomposes a visualization into a set of components like marks, controlled processing graphical features, retinal encodings and positions. It is particularly useful to identify the similarities and differences between visualizations at a component level. Moreover, it points out a way to design and develop new visualizations through different combinations of visualization components. Similarly, a visualization may be modified or improved by changing or replacing its involved components.
Building on Card and Mackinlay (1997)'s work, Chi and Riedl (1998) propose a data state model to highlight the transformation process of how user data are converted into certain visual formats. This model outlines the transformation operations performed within and between the data stages. Compared to the previous visualization reference models, the data state model provides a better view of the intermediate data transitions occurred within the visualization process. The model incorporates three types of between-stage transformations: data, visualization, and visual mapping transformations (Table 2.1). These between-stage transformations can completely modify the data structure of the raw data. In comparison, transformation operations (i.e. value stage operator, analytical stage operator, visualization stage operator, and view stage operator) performed within each stage do not change the data structure. The within-stage operations, such as filtering and sub-setting, are utilized to facilitate the between-stage transformations. Similar to the table notation framework, the data state model points out a way to create, customize and modify visualization techniques.

<table>
<thead>
<tr>
<th>Transformation Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Transformation</td>
<td>To convert source data into a certain metadata format (i.e. a form of analytical abstraction).</td>
</tr>
<tr>
<td>Visualization Transformation</td>
<td>To convert the data from an analytical abstraction format to a certain visual format which enables the data to be visually displayed on the screen.</td>
</tr>
<tr>
<td>Visual Mapping Transformation</td>
<td>To generate views requested by users.</td>
</tr>
</tbody>
</table>

Table 2.1 Between-stage transformation operations (Chi, 2000)

Information visualization plays an important role in decision making by turning large volumes of data into valuable actionable information, reducing the involved uncertainties and risks, and helping people to take informed decisions. The impacts of information visualization on decision making are further discussed in the next subsection.

### 2.2.2 Impacts of Information Visualization on Decision Making

The way of how information visualization supports decision making, by and large, is through providing visual intelligence. Visual intelligence refers to the amount of useful decision support information that a decision maker gets by creating, manipulating, layering and
viewing visualization (Bai, White & Sundaram, 2009a). Its ultimate goal is to enhance human cognitive performance and reduce the time to enlighten the decision maker on the underlying data (Jern, 1999). Bai, White and Sundaram (2009b) introduce the concept of Visual Intelligence Density specifically to evaluate the effectiveness of visualization techniques/tools with regard to their support for decision making activities.

The existence of visual decision support has been duly acknowledged by many researchers. Wünsche (2004), Ware (2004) and Spence (2007) state that one important support of visualization is to aid decision makers to make sense of data and foster insights, which is a necessity particularly for dealing with decision problems involving large volumes of data. This point is further reinforced by Zhu and Chen (2008) who articulate the power of information visualization from the perspective of supporting decision making tasks. To assist in selecting the optimal solution from a set of alternatives, Nutt (1998) proposes three different approaches particularly for personal decision making: analytical, subjective and judgmental approaches. Among them, the analytical approach of alternative selection is ideally applied to the analytical decision paradigm that involves well-structured decision problems and tasks, while the other two approaches are more appropriate to be deployed for the anarchic decision paradigm dealing with semi or ill structured problems. Building on top of this study, Zhu and Chen (2008) synthesize and justify the support of information visualization for the analytical, subjective and judgmental approaches with a particular focus on personal decision making. The support from information visualization ranges from presenting the trade-offs among alternatives, to helping with what-if analysis, to detecting patterns/trends, to acquiring domain/system specific knowledge, and to enhancing situation awareness.

Furthermore, Al-Kassab et al. (2013) and Lurie and Mason (2007) illustrate visual decision support through applying visualization techniques to address real-world decision problems. Card et al. (1999) identify some prominent contributions of information visualization towards decision support when applied appropriately. These benefits are the reduced searching workload, enhanced information detection of trends/patterns of data, improved perceptual inference and monitoring abilities, increased memory and processing abilities etc. Despite the advances achieved in the field of information visualization, realizing these benefits is a rather
challenging task to achieve. We discuss the major challenges associated with visual decision support in the following subsection.

### 2.2.3 Challenges with Visual Decision Support

Providing effective visual decision support is quite challenging due to five critical problems, that is, (1) high data complexity, (2) large data volumes, (3) dynamic visualization paradigms, (4) visualization misuse, and (5) visualization quality and aesthetics (Bai, White & Sundaram, 2009b).

**High Data Complexity**

People nowadays often face with the increasingly high data complexity involved in various visualization application domains. Such complexities are mainly caused by the complex and combined data types involved the source data. More specifically, real-world decision problems often involve multiple data types. Examples of such source data could be biographies, annual reports, films, videos and patents (Card et al., 1999). Difficulties in conquering this challenge reside in the high-dimensional data sets in which the number of variables or dimensions or both is more than three. Due to the lack of powerful visualization techniques to address such problem (Hibbard, 1999, Santos & Brodlie, 2004), visualizing high-dimensional data has become a major obstacle for information discovery tasks within the domain of information visualization (Santos & Brodlie, 2004). There is an increasing need for visualizing complex data types. For example, the pharmaceutical industry has a high demand for understanding, communicating and utilizing complex data types representing chemical structures and genome sequences (Saffer et al., 2004).

**Large Data Volume**

There is a lack of support for coping with tremendous data volumes. It is widely acknowledged that people nowadays are experiencing tremendous growth in both data and information volumes (Jern, 1999; Hibbard, 2004; Saffer et al., 2004). The issue with large data volumes can incur many problems pertinent to data navigation and exploration (Jern, 1999). However, traditional visualizations (such as 2D or 3D statistical charts), even presented in interactive formats, are ineffective and insufficient for dealing with extremely large data volumes (Saffer
et al., 2004). Jern (1999) expresses the view that retrieving and exploring useful information from large volumes of data has become a focal issue in the arena of information visualization.

**Dynamic Visualization Paradigms**

Besides the above data related problems, there is a lack of support for dealing with the complexities raised by paradigm dynamics. Paradigm dynamics mainly concerns the requirement of visualizing changes, trends and patterns emerging over time (Chen, 2005). Chen (2005) opines that there is an increasing need for detecting and visualizing changes either gradual or abrupt over time. For example, people may be interested in visualizing abrupt changes (such as climate changes and stock market crash) and gradually emerging trends or patterns (such as long-term climate changing patterns and long-term stock price changing trend)(Chen, 2006). Identifying such changes/patterns/trends allows them to investigate and interpret the underlying causes of the changes happened in the past and even predict future trends. Nevertheless, existing visualization techniques (such as cone tree and treemap) are not sufficient for addressing such issue with visualizing changes over time (Chen, 2005). This is because the existing visualization techniques focus more on supporting structured decision problems while the problems involving dynamical paradigms are illogical and dynamic in nature (Chen, 2005).

**Visualization Misuse**

Due to the complexities involved in various application domains, addressing a certain decision problem often requires deploying multiple visualization techniques. However, it can be difficult for people to select appropriate visualization techniques and even misuse them due to the lack of knowledge regarding the alternative visualizations.

**Visualization Quality and Aesthetics**

Visualization quality is often difficult to assess because the evaluation is dependent on the subjective judgments of people (Keim & Kriegel, 1996). It refers to the expressiveness and effectiveness of the visual contents presented by a visualization. Such difficulty brings challenges for users to build the trust over visualizations. For example, users may feel confused with how accurately a particular visualization technique transforms the underlying
source data to certain visual formats as such transformation details may be hidden (Chen, 2005). Moreover, visualization aesthetics is intimately linked with visualization quality. Although visualizations emphasize more on presenting the essence of source data (not just nice-looking pictures) (Hibbard, 2004), beautiful visualizations can be more attractive and enhance the performance of users (Chen, 2005).

The above challenges bring many difficulties to develop purposeful effective visualizations for decision making under varying contexts. They however guided us to discover the important aspects of information visualization to better understand/address/relieve them keeping in mind different stakeholders. We are hence motivated to examine information visualization from the perspectives of stakeholder, purpose, and context.

2.3 Visualization Stakeholders

In this section, we focus on the review of stakeholders who have certain interests in a visualization. Specifically, we provide an overview of typical types of stakeholders in subsection 2.3.1. We then examine two fundamental aspects that significantly affect human cognitive abilities, that is, human information processing (in subsection 2.3.2) and skill development (in subsection 2.3.3). Based on the understanding of these influential aspects, we proceed to highlight the stakeholder-related bottlenecks that prevent people from accomplishing informed decision making in subsection 2.3.4.

2.3.1 Stakeholder Overview

Under the context of visual decision making, stakeholders of a decision problem may play four different roles in the process of decision making, that is, visualization designer, viewer, decision maker and presenter. The same stakeholder may possess multiple roles simultaneously. For example, a stakeholder may design and develop a visualization and then present it to other stakeholders who have interest in the visualization and/or need it to support decision making. These stakeholder roles are interrelated with each other. Designers are the producers of visualizations. They design, develop and modify visualizations to fulfill their purposes and requirements, which may not be consistent with the requirements of
stakeholders in other roles. Viewers are end consumers of a visualization, and they may deploy the visualization to address their decision problem(s). Presenters, who work in between designers and viewers, interpret a visualization from their own perspective and attempt to impose their understanding onto viewers and eventually affect how the visualization is utilized by the viewers (especially first-time viewers).

Viewers can be further classified into two groups: target viewers and non-target viewers. The former refer to end consumers whose purposes are the key driver for visualization design and development. That is, visualization creation should concern visualization requirements, personal characteristics and preferences, cognitive styles, prior knowledge of target viewers. In contrast, non-target viewers may be interested in the visualization but their purposes and requirements are least concerned by visualization designers. For both viewer groups, general principles and good practices in visualization design are applicable for designers. For example, Miller (1956) suggests that human cognitive capacity is limited because the short term memory capacity of an average individual is delimited to seven plus or minus two elements. The short term memory is able to handle more elements when they are coded into patterns already familiarized by the individual (Carroll & Zeller, 2012). This implies that visualizations with overloaded information or high data ink may limit human visual information processing.

Among viewers, there is a special group of individuals, namely decision makers, who have a particular interest of applying visualizations to support various decision making tasks. Nonetheless, not all viewers are decision makers as they may leverage visualizations for other purposes such as interpreting and communicating information. Decision makers may work alone as individuals or in groups such as workgroups, teams, or virtual teams (Shim et al., 2002). Within the context of organizational decision making, they can be upper level executives, middle level managers, and lower level supervisors (Basi, 1998). In terms of the roles played in a decision making process, a decision maker may act as a creator, an actor, and a carrier of decisional tasks (Langley et al., 1995). A decision maker follows his subconsciousness to get immediate rich insight into a decision problem and can then be inspired to go beyond his rationality boundary and become creative. Such sub-consciousness originates from acute intuition and good judgment, which are analyses already developed into the decision maker’s habit and capacity for rapid response through the recognition of
familiar patterns or situations (Larkin & Simon, 1987). Moreover, the decision maker carries knowledge and experience and applies them to support decision making and inspire others.

No matter what roles visualization stakeholders may play, they share common processes/characteristics of processing information, which are discussed in subsection 2.3.2.

### 2.3.2 Human Information Processing

Human information processing refers to cognitive processes and activities that one performs to acquire, encode, transform, integrate, retain, retrieve and apply information (Wickens et al., 2004). It involves rational-analytical and experiential-intuitive cognitive styles and both of them are equally important to decision making (Epstein et al., 1996). The two thinking approaches work together in a harmonious fashion, and are closely coupled with human memory systems to foster the cognitive capacity of the human brain.

Much research has been conducted in the field of psychology to study how information is processed in human memory systems. For instance, Atkinson and Shiffrin (1968) propose a multi-store model based on a series of cooperative memory stores. This model has become one of the most influential models of human information processing nowadays and serves the basis for much research in this area, e.g. the search of associative memory model (Raaijmakers & Shiffrin, 1981). Due to the critique of the multi-store model oversimplifying the processes and complexities involved in human memory systems, Baddeley and Hitch (1974) develop a model with a particular focus on the depth of information processing. Unlike the multi-store model, the levels of processing model does not differentiate short term memory and long term memory. It argues that how well information is remembered in human memory depends on the way of how it is encoded, and how easily the information can be recalled is determined by how deep it is processed.

Due to the importance and influence of the multi-store model, we review this model in detail. The multi-store model consists of three essential types of memory stores, that is, sensory, short-term, and long-term memories. Different memory stores have different capacities and durations, and information retained by each memory system could be lost or replaced by other information. As demonstrated in Figure 2.2, human information processing is triggered
by a stimulus a person receives from the outside environment via sense organs (Wickens et al., 2004). The stimulus is generated during the interaction with the environment by five typical senses of sight, touch, smell, hearing and taste. The incoming information is then automatically stored in sensory memory, which has rather limited capacity and duration and cannot be prolonged by rehearsal (Wood, Wood & Boyd, 2010). Sensory memory is the shortest type of memory and its duration is less than 200 milliseconds (Bubb & Wohlfarter, 2012). The sensory information is encoded and analyzed unconsciously by the person, and is then passed into short-term memory if it successfully attracts the person’s attention. Unlike the unconscious information analysis in sensory memory, the information is consciously processed and retained in short-term memory. The capacity and duration of short-term memory, though limited, can be extended by information organization strategies and rehearsal (Atkinson & Shiffrin, 1968). After continuous rehearsal, the information is then passed into long-term memory of which the capacity is unlimited and the duration is up to many years. In long term memory, the information is semantically encoded, organized and stored permanently. It can also be retrieved and moved back into short term memory for various applications.

Figure 2.2 A Model of Human Information Processing (Wood et al., 2010)

Among all the external stimuli, visual perception is particularly important for the person to acquire an immediate evaluation of the information delivered by a visualization (Kuo, 1998). Ware (2004) concludes that human visual information processing involves parallel property extraction, pattern perception, and sequential goal-directed processing. Such activities are all performed by leveraging the three memory stores.

The information processing performance of people, to a large extent, determines their abilities of developing new skills, which affects their cognitive styles, preferences and
requirements in the process of decision making. We explore how humans acquire new skills in the following section.

2.3.3 Human Skill Acquisition

A stakeholder may go through six principal stages of learning or skill development through which he progress to achieve higher levels of proficiency and/or expertise (Dreyfus & Dreyfus, 1986). These fundamental learning development stages are novice, advanced beginner, competent, proficient, expertise and master. Each stage is associated with six mental functions, i.e. similarity recognition, aspect recognition, decision paradigm, perspective, commitment, and monitoring. The learning development stages and mental functions together form the building blocks of the skill acquisition model proposed by Dreyfus and Dreyfus (1986).

As going through the learning development stages from novice to master, people gradually develop their abilities of resolving new problems through recognizing the similarities between the new problem situation and previous problem situations that they have experienced. This, in turn, enables the viewers to gain stronger problem solving and decision making capabilities and better performance.

According to Dreyfus and Dreyfus (1986), people at beginner levels are only capable of perceiving and understanding simple clues in a problem context and recognizing very limited similar features to their experienced problems. They have to depend on the available relevant rules and directions for guiding their activities, and on deliberately monitoring their own performance and getting feedback. The lack of guidance on performing certain tasks or the lack of experience for resolving relevant problems may cause them to present low performance levels.

In contrast, people with higher levels of expertise often have stronger capabilities to understand and resolve problems though basing their judgments against relevant experience and knowledge, which often leads to a better performance (Dreyfus & Dreyfus, 1986). They are more likely to cope with complex problems and see through complicated situations,
decide task requirements for resolving the problems, and perform the tasks with less monitoring efforts and more commitment to problem solving activities.

Understanding the features of information processing and skill development reveal the stakeholder-related issues affecting decision making performance. These issues are explicated in the following subsection.

2.3.4 Bottlenecks in Human Decision Making Performance

The challenges relating to visualization stakeholders are mainly associated with two areas, that is, the intrinsic limitations of human information processing, and the inadequacy of prior knowledge.

Intrinsic Limitations with Human Information Processing

The intrinsic limitations of human sensory and short-term memories in terms of memory capacity and duration have become a bottleneck in visual decision making. Miller (1956) purports that human cognitive capacity is limited because the short-term memory capacity of an average individual is delimited to seven plus or minus two elements. This has caused short-term memory to become the biggest bottleneck in human information processing (Adnan, Noor & Aripin, 2003). Nonetheless, short-term memory is able to handle more elements when they are coded into patterns already familiarized by the individual (Carroll & Zeller, 2012). It reveals the importance of taking into account the characteristics of human memories in visualization designs. How well the way to encode and present the information can fit in the cognitive characteristics and memory constraints of a stakeholder determines how effectively the information can be memorized by the stakeholder (Wood et al., 2010). According to the cognitive fit theory (Vessey, 1991; Vessey & Galletta, 1991), this fit affects the stakeholder’s performance in terms of decision making and problem solving.

The Lack of Prior knowledge

Visualization stakeholders may not have sufficient prior knowledge to manipulate and interpret visualizations (Chen, 2005). A visualization stakeholder essentially needs two types of knowledge to be able to comprehend the message conveyed by a visualization, that is,
knowledge of operating the visualization system to fully exploit its benefits, and domain-specific knowledge required for interpreting the generated visual contents (Chen, 2005). For example, in the pharmaceutical industry, the proximity maps generated by OmniViz Galaxy system can be applied to depict the way of how proteins mutually relate to and interact with one another (Saffer et al., 2004). To create a proximity map, users need the knowledge of how to manipulate the visualization system, and the knowledge pertaining to proteins, relationships between proteins and some background information of the pharmaceutical industry.

A decision problem may involve multiple stakeholders with different or even conflicting interests/preferences/requirements in regard to visualizations, which accordingly reflect their purposes with leveraging visualizations. However, the diversity and complexities involved in stakeholder purposes often cause these purpose difficult to fulfil. We proceed to review visualization purposes in detail in the subsequent section.

2.4 Visualization Purposes

The purposes of a visualization reflect what the visualization attempts to accomplish, and may be articulated from different angles. From the task perspective of information seeking, visualization purposes are to support general user activities such as overview, zoom, filter and details-on-demand. From the functional perspective of visualization, they are to assist people with information discovery, information interpretation, and decision making (Card et al., 1999). From the perspective of human cognition, they are to aid the formation of mental images of information, foster new insights and eventually amplify human cognition (Card et al., 1999; WanAdilah et al., 2003; Griffin, 2009; Al-Kassab et al., 2013).

In this section, we examine visualization purposes from two essential perspectives, that is, application domains and visualization tasks. The task-based purposes are further classified into two groups: macro-level and micro-level purposes. A summary of typical visualization purposes are presented in Figure 2.3. Based on the review of visualization purposes in subsection 2.4.1 and 2.4.2, we outline the challenges with purpose fulfillment in subsection 2.4.3.
2.4.1 Domain-Related Purposes

Visualizations have been widely adopted in many domains and disciplines to support presenting and communicating information and fostering insights (WanAdilah et al., 2003). Card et al. (1999) identify seven typical application domains of information visualization, that is, statistical and categorical data management, digital library management, personal services support, complex documents management, history management, classifications management, and networks management. A summary of typical purposes in these application domains is presented in Table 2.2. Real-world visualization problems often span multiple application domains, instead of merely residing in a single domain. For example, in an electric utility company the senior management often requires a good visibility of the frequently problematic network areas that have incurred exceptionally higher maintenance costs. Dealing with such problem requires appropriate management of electricity network data, statistical data of maintenance work orders and costs, and complex semi-structured documents like network fault inspection reports.

Fulfilling each domain-related visualization purpose requires accomplishing a series of related tasks such as information navigation, retrieval and discovery. Visualizations have been acknowledged as powerful support for these tasks (Büring & Reiterer, 2005). The task-based purposes are discussed in the section below.
### Application Domain

<table>
<thead>
<tr>
<th>Application Domain</th>
<th>Domain-Specific Visualization Purpose</th>
</tr>
</thead>
</table>
| Statistical & Categorical Data Management | - To retrieve information that matches a certain set of filtering criteria  

                        | - To discover patterns/trends/relationships within the underlying data set  

                        | - To interpret a certain subject matter  

| History Management                     | - To understand and interpret an existing phenomenon  

                        | - To predict future trends  

| Personal Services Support              | - To aid in exploring and searching the information of interest  

                        | - To provide personal decision making support  

| Complex Documents Management           | - To present complex documents such as contracts, books, bibliography and annual reports  

                        | - To explore and retrieve information  

                        | - To discovery relationships/patterns embedded among the documents  

| Digital Library Management             | - To navigate and explore large collections  

                        | - To search interested items  

                        | - To discover relationships/patterns among the items  

                        | - To present the overall structure of item collections  

| Information Hierarchy Management       | - To manage/relieve the extensive information load  

                        | - To support information navigation and retrieval  

                        | - To present the structure of a set of hierarchical objects through relative positions, sizes and object containment/adjacency  

| Network Management                     | - To present the features/patterns of the relationships among the objects contained in a network such as complex social structures among people  

---

Table 2.2  Domain-related visualization purposes, summarized from Card et al. (1999)

---

### 2.4.2 Task-Related Purposes

Under each application domain, visualizations are deployed to fulfill a variety of task-based purposes. At a macro level, visualizations can be exploited to support tasks like information presentation, navigation, retrieval, query, discovery and/or interpretation. Typical macro-level task-based purposes are (1) to discovery relationships/patterns from large volumes of data points, (2) to facilitate data comparison, (3) to track/display trends over time, (4) to illustrate structure or composition, (5) to analyze words/texts, and (6) to explore geographical data (IBM Many Eyes, 2010). For instance, stacked charts can be applied to highlight the comparisons and variations among multiple variables while node-link diagrams may aid in
presenting the structure/hierarchy of a set of objects (Spence, 2007; Heer, Bostock & Ogievetsky, 2010).

Furthermore, a macro-level visualization task can be further decomposed into a set of micro-level tasks each of which indicates a possible visualization purpose. Shneiderman (1996) identifies seven general visualization tasks of this kind, i.e. overview, zoom, filter, details-on-demand, relate, history, and extract (Table 2.3). The purposes of why visualizations are deployed to support such tasks are summarized in Table. It implies that a macro-level task-based purpose can be accomplished by fulfilling a set of micro-level purposes. For example, Shneiderman (1996) develops a model to guide the design and implementation of visual information exploration by combining such micro-level tasks together. In addition, both macro-level and micro-level task-based visualization purposes are general and domain independent.

<table>
<thead>
<tr>
<th>Micro-Level Task</th>
<th>Task-Oriented Visualization Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>To present an overview of the entire data set.</td>
</tr>
<tr>
<td>Zoom</td>
<td>To facilitate zooming in/out based on the interested items.</td>
</tr>
<tr>
<td>Filter</td>
<td>To support filtering out unrelated items and highlighting the interested ones only.</td>
</tr>
<tr>
<td>Details-on-demand</td>
<td>To show details of the selected item(s) when required.</td>
</tr>
<tr>
<td>Relate</td>
<td>To help with identifying the relationships among items of interest.</td>
</tr>
<tr>
<td>History</td>
<td>To depict a history of operations for easy undoing, replay, refinement and alike.</td>
</tr>
<tr>
<td>Extract</td>
<td>To display the relevant subset extracted from a certain collection and show query criteria.</td>
</tr>
</tbody>
</table>

Table 2.3 Micro-level task-related visualization purposes, adapted from Shneiderman (1996)

Fulfilling a domain-level purpose often requires stakeholders to achieve a set of relevant task-level purposes. Compared to the overall domain-level purpose, a task-level purpose is relatively easier to accomplish. This indicates the importance of aligning the domain-level purpose to the right set of tasks. However, such purpose alignment could be rather challenging to perform. The challenges involved will be elaborated in the following subsection.
2.4.3 Challenges with Purpose Alignment

The challenges with fulfilling the domain and task related visualizations purposes originate primarily from the process of aligning visualization purposes at different levels for a decision problem. In particular, the visualization purposes of a stakeholder in dealing with a certain decision problem should be analyzed by a more holistic manner. This means to identify and define the domain level purpose and decompose it into a series of task level purposes; fulfilling the task level purposes should enable the stakeholder to achieve the domain level purpose. Such purpose alignment process requires identifying visualizations that offer maximum support for addressing the underlying decision problem and fulfilling the purposes. Moreover, the decision problem can involve multiple stakeholders with varied and even conflicting visualization purposes. The purposes of the same stakeholder may even change over time, which limits the effectiveness of a visualization within a particular time period and context. This may further exacerbate the challenges with visualization purpose alignment.

Decision problems, stakeholders, and purposes constitute integral parts of visualization contexts. They largely expose the complexities involved in the visualization contexts, which will be reviewed in detail in next section.

2.5 Visualization Contexts

Context is a broad term and can be articulated in many different ways depending on the underlying research domains. From the perspective of human-computer interaction, context is considered as any situational information that characterizes a user, system application and user-system interaction (Dey, 2001). In the area of mobile computing, context is treated as knowledge determining the status of users or IT devices such as surroundings, situation, task and location (Schmidt et al., 1999). In the domain of context-aware systems, Vrbaski et al. (2012) describe context as any contextual information associated with the computing, user and physical environments.

The context of a visualization refers to the information of any environmental entities that can influence visualization design, implementation, application and evaluation (Bai, White & Sundaram, 2013c). It embraces relevant problem situations, time, space, social context, and
technological context; it also encompasses the visualization profiles of stakeholders such as their cognitive styles, personal characteristics and preferences, pre-knowledge, age, gender and even their mood when taking decisions. Such contextual factors largely affect the effectiveness of the visualization for its stakeholders to fulfill a certain purpose. In this section, we review visualization contexts in subsection 2.5.1 and dwell on the context related challenges imposed on information visualization in subsection 2.5.2.

2.5.1 Context Complexities

Visualization contexts are mainly concerned with the problem context within which visualizations are deployed and the situational context of visualization stakeholders. These contexts are complex by nature. To tackle the complexities involved in context, many researchers have attempted to categorize contextual information. For example, Schilit, Adams and Want (1994) identify three general contextual groups, i.e. computing context, user context and physical context. This categorization scheme is further extended by Chen and Kotz (2000) by adding in two new groups: time context and context history. Building on top of these general context classifications schemes as well as domain related context categorizations in mobile computing and adaptive geographical information systems (Nivala & Sarjakoski, 2003; Petit, Ray & Claramunt, 2006), Wu and Chen (2009) categorize contextual factors into four groups: user context, activity context (i.e. task, tool and data), physical context (including location, orientation, physical surroundings, time, and movement state), and system context (i.e. system style and capability). Hahn et al. (1992) experimentally prove that the digestion of information is greatly influenced by the available time, which, in turn, affects the quality of decisions made against the information.

Building on top of the existing context classifications, we organize visualization contextual factors via four principal dimensions: problem, stakeholder, purpose, and time (Figure 2.4). Each dimension consists of a series of contextual factors.
More specifically, the problem dimension is concerned with the contextual information pertaining to the problem situation to be supported and potential solutions. The stakeholder dimension involves any stakeholder-related aspects that affect the design, development, cognition, interpretation and/or evaluation of a visualization by different kinds of stakeholders. The purpose dimension incorporates the contextual information about what a visualization stakeholder is trying to achieve through applying the visualization in a particular domain to address/accomplish a certain problem/task respectively. The time dimension concerns the time data (such as year, season, month, week, day, and time) associated with decisional problems, stakeholders and purposes. Table 2.4 provides a brief summary of our visualization context classification scheme.
<table>
<thead>
<tr>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem</strong></td>
</tr>
<tr>
<td>- Problem Situation</td>
</tr>
<tr>
<td>- Visualization Tasks <em>(overview, zoom, filter, details-on-demand, relate, history, and extract)</em></td>
</tr>
<tr>
<td>- Information Generation <em>(data transformation requirements)</em></td>
</tr>
<tr>
<td>- Information Representation <em>(data type, data quality, data volume, and relevant representation techniques)</em></td>
</tr>
<tr>
<td>- Information Presentation <em>(semantic layer, animation, interaction, output device (size, resolution), input device (touch panel, keyboard, mouse, etc.), network connectivity, and communication costs/bandwidth)</em></td>
</tr>
<tr>
<td>- Location <em>(latitude, longitude, altitude, city, suburb, country, etc.)</em></td>
</tr>
<tr>
<td>- Physical Surroundings <em>(lighting, temperature, surrounding landscape, weather conditions, noise levels, etc.)</em></td>
</tr>
<tr>
<td>- Movement State <em>(static or mobile)</em></td>
</tr>
<tr>
<td><strong>Stakeholder</strong></td>
</tr>
<tr>
<td>- Stakeholder Profile <em>(cognitive styles, personal characteristics and preferences, educational background, culture and social background (faith, nationality, etc.), personality (introversive/extroversive), physical condition (disability, left/right hands, etc.), age, gender, mood, etc.)</em></td>
</tr>
<tr>
<td>- Stakeholder Ability <em>(prior knowledge (e.g. knowledge in the problem domain, past experience with manipulating the visualization, past experience with using the visualization system), skill acquisition ability (i.e. novice, advanced beginner, competent, proficient, expert, and master levels), etc.)</em></td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
</tr>
<tr>
<td>- The Domain Perspective <em>(e.g. statistical and categorical data management, digital library management, personal services support, complex documents management, history management, classifications management, and networks management)</em></td>
</tr>
<tr>
<td>- The Stakeholder Perspective <em>(e.g. to support financial analysis, to support education and E-learning, to support military debriefing, etc.)</em></td>
</tr>
<tr>
<td>- The Task Perspective <em>(e.g. to discovery relationships/patterns from a large volume of data points, facilitate data comparison, track/display trends over time, illustrate structure or composition, analyze words/texts, and explore geographical data)</em></td>
</tr>
<tr>
<td><strong>Time</strong></td>
</tr>
<tr>
<td>- Problem-related time data <em>(e.g. time-series data involved in a decisional problem, when the effectiveness of a visualization solution is confirmed, time pressure against problem solving, etc.)</em></td>
</tr>
<tr>
<td>- Stakeholder-related time data <em>(e.g. when a new stakeholder appears)</em></td>
</tr>
<tr>
<td>- Purpose-related time data <em>(e.g. when a purpose becomes relevant)</em></td>
</tr>
</tbody>
</table>

Table 2.4 Contextual factors affecting visualization effectiveness, from the author’s creation (Bai, White & Sundaram, 2013a)
Whether a visualization is effective and fit for purpose, by and large, is determined by the context where it is applied. We dwell on the context related impacts on visualization effectiveness in the following subsection.

2.5.2 Impact on Visualization Effectiveness

The effectiveness of a visualization depends upon the context where it is applied. Visualization context determines what visual designs are most effective for aiding visualization stakeholders to address their interested decisional problem(s) and to fulfill the intended purpose(s). The design elements (e.g. border, background, combination of elements) involved in a visual design may significantly influence a stakeholder’s understanding and interpretation of the underlying data (Ziemkiewicz & Kosara, 2010). Moreover, cognition, by definition, refers to mental activities and processes by which an individual perceives and understand information from an external environment in order to match his internalized view of the same environment to reality (Sharda, Delen & Turban, 2014). Different cognitive styles may lead to different preferences for information and even different decisions (Ho & Rodgers, 1993). When a visualization is appropriately deployed for the right context, quite often its stakeholders can obtain illuminating, useful, relevant and actionable information from the visualization with less effort and time.

Addressing the complexities involved in visualization contexts is rather difficult to accomplish; the challenges involved will be discussed in the next subsection.

2.5.3 Challenges with Addressing Context Complexities

The challenges with addressing context complexities are caused by two fundamental features of visualization contexts: diversity and dynamics. Resolving the impacts from visualization contexts requires dealing with contextual factors pertaining to both decisional problem context and the situational context of visualization stakeholders. Different contextual factors can lead to varied or sometimes even conflicting visualization requirements to support them.

Furthermore, the contextual factors involved in a visualization problem may change over time. Specifically, the decisional problem context within which a visualization is deployed may
change as the underlying problem situations, time, space, social context and/or technological context may vary. The same visualization may be applied to address different user problems under a variety of visualization contexts, but the visualization contexts of the same problem can change over time. For example, geographical maps can be applied to various application domains (such as statistical data analysis, personal information support, and historical data management) to address diverse decision problems. As users of these maps become familiar with the underlying visualization tool and their decision problems, their visualization profiles (e.g. cognitive characteristics and personal preferences) may change accordingly.

So far, we have elaborated on the challenges with visualization problems, stakeholders, purposes and contexts. Addressing these challenges requires systems with powerful visualization capabilities. We explore information visualization systems and examine their support for information representation and presentation processes in the subsequent section.

### 2.6 Visualization Systems

Visualization techniques and systems may facilitate decision makers to better identify and communicate problems, develop alternative solutions and take decisions. Human beings are highly attuned to visual information, such as images, charts and media. To amplify decision makers’ perception and cognition of the underlying data, decision support systems (DSS) should include appropriate visualization subsystems or be used together with separate visualization systems. This enables DSS to leverage a variety of visualization techniques to visually encode and present the data and allow decision makers to interact with the data.

The visualization capabilities of a decision support system may significantly affect how effectively the presented data can be perceived, understood and interpreted by decision makers, which, in turn, decides the quality and performance of decision making. Visualizations can enable decision makers to take better decisions and to fulfill their intended purposes and even bridge the gap between owning the data and effectively leveraging the data, especially if designed and applied appropriately. Misleading or wrong visualizations can be costly, waste resources and time, and lead to increased risks. Information visualization is therefore an important building block to all decision support systems.
In this section, we present an overview of information visualization systems in subsection 2.6.1, and then concentrate on exploring and examining their support for information representation and presentation processes in subsections 2.6.2 and 2.6.3 respectively. Based on the review of the system support, we dwell on the challenges with developing effective visualization support in subsection 2.6.4.

2.6.1 Overview of Visualization Systems

With the advances achieved in the field of information visualization, a large number of visualization techniques and systems have been implemented and widely adopted to support various decision making processes and tasks. They can be broadly divided into two categories, that is, specific information visualization system (IVS) and IVS generators.

A specific IVS is a visualization application that is developed for implementing a limited set of visualization techniques and offers very limited flexibilities especially for creating new visualizations and integrating multiple visualization techniques. Representative specific IVS have been reviewed by many researchers such as Chi (2000), Chen (2006), Turetken and Sharda (2007), and Roque, Slaughter and Tkatsenko (2010). Among them, the system reviews of Chi (2000) and Chen (2006) are independent of visualization application domains. Compared to them, the reviews, conducted by Turetken and Sharda (2007) and Roque et al. (2010), tend to be domain-specific. That is, they have clear focuses on visualization applications for presenting web spaces and health care data in respective. Due to the design intentions pertaining to domain/data, specific IVS often involve very few options of visualization techniques and are restricted to be used in limited domains.

In comparison, IVS generators serve as system platforms or environments where specific IVS can be generated for addressing certain type of structured or unstructured problems. They often offer multiple visualization techniques and enable the development and application of instances of these techniques. Examples of IVS generator are AVS (Upson et al., 1989), IBM OpenDX system (Lucas et al., 1992), Khoros (Konstantinides & Rasure, 1994), spreadsheet for information visualization system (Chi et al., 1997), and TiViPE (Lourens, 2004).
To examine how well IVS address or relieve the challenges with visualization problems, stakeholders, purposes and contexts, we proceed to explore the support of specific IVS and IVS generators from two perspectives: information representation (encoding data into visual structures) and information presentation (presenting visual structures for user interaction).

2.6.2 Information Representation Support

Information representation refers to the process of visually encoding the underlying data into appropriate visual forms. Many visual representation techniques and systems have been developed to support such process. They can be classified according to the underlying data types that they are dealing with. Shneiderman (1996) identify seven fundamental data types within the field of information visualization, that is, one-dimensional, two-dimensional, three-dimensional, multi-dimensional, temporal, tree and network data. Typical techniques and systems for representing each data type have been reviewed by many researchers, for example, Card, Mackinlay and Shneiderman (1999), Chen (2005), Zhu and Chen (2005), Spence (2007), and Heer, Bostock and Ogievetsky (2010). An example of the review of representative visualizations is demonstrated in Figure 2.5.
Low-Dimensional Data

Among the seven data types identified by Shneiderman (1996), one-dimensional data type is the simplest form of data. One-dimensional data may be either a single value or a one-dimensional data collection (Spence, 2007), for example, a single temperature or a collection of temperature values that are measured at different times of a day. One-dimensional data can be visualized in many ways such as icon, color, size, linear scale or one-dimensional axis.

Typical two-dimensional data can be used to describe any object characterized by two attributes. For example, locations may be represented by their geographical latitudes and longitudes, while a point within a two-dimensional coordinate system can be referred via its x and y coordinates. A large number of visualization techniques may be deployed to present two-dimensional data, for example, scatter plots (Spence, 2007), two-dimensional geographical maps (Shneiderman, 1996), and TileBars (Hearst, 1995).

The three-dimensional data type may be deployed to portray any real world object that can be defined by three attributes (Spence, 2007). For instance, a location may be indicated by a combination of latitude, longitude and altitude. Three-dimensional maps and contour maps are often used to illustrate three-dimensional geographical data. Many advanced computer-based graphical design tools have been developed to visualize three-dimensional objects by projecting the data to a two-dimensional landscape (Shneiderman, 1996).

High-Dimensional Data

Multi-dimensional data often represent complex associations with each being indicated by more than three attributes (Shneiderman, 1996). A common way to visualize multi-dimensional data is to decrease the dimensions and project the data within two-dimensional landscape (Santos & Brodlie, 2004). Some typical visualization techniques that can visually depict high-dimensional geometries within two-dimensional space are three-dimensional landscape (Chen & Paul, 2001), parallel coordinates (Inselberg, & Dimsdale, 1990; Inselberg, 1997), star plots (Coekin, 1969) and scatter plot matrix. For example, parallel coordinates and scatter plot matrix can be used to explore the relationships (e.g. apparent trade-off or correlation) between multiple variables/measures involved in a set of high-dimensional data (Heer, Bostock & Ogievetsky, 2010).
**Temporal Data**

For representing temporal data, one of the most wildly adopted visualization techniques is time line, which is often used for project management (Shneiderman, 1996). Another example could be LifeLines, which are proposed by Plaisant, Milash, Rose, Widoff and Shneiderman (1996) and used specifically for visualizing personal history such as patient records.

**Network Data**

A simplistic form of network data is in tree structures, which can be adopted to manage and represent complex relationships among the items within vast volumes of collections. Representative visualization techniques for dealing with tree data are tree maps (Shneiderman, 1992), disk trees (Chi et al., 1998), cone trees (Robertson, Mackinlay & Card, 1991) and 3D hyperbolic cone trees (Munzner & Burchard, 1995). These techniques are quite effective for coping with the complexities and tremendous data volumes involved in the source collections.

Similar to tree structures, network data also aim to define the intricate associations among the nodes within a collection. Nonetheless, the relationships among the nodes are more complicated than those in hierarchies. Force-directed diagrams, arc diagrams and matrix views are often used to illustrate the relationships among the network nodes. In particular, force-directed diagrams may be employed to portray the overall structure of a network and illustrate the relationships among the involved nodes (Bollen et al., 2009). Arc diagrams and matrix views may be deployed to reflect the features/patterns of the relationships among the objects involved in a network (Heer, Bostock & Ogievetsky, 2010). Some techniques used for visualizing hierarchical data may also be applied to present network data, such as 3D hyperbolic cone trees (Munzner & Burchard, 1995).

Once data are properly represented in visual structures, they need to rely on information presentation to become easily accessible and manipulable for their stakeholders.
2.6.3 Information Presentation Support

The field of information presentation has attracted interests from many researchers. By examining a large number of visualization techniques and systems, Spence (2007) summarizes a series of typical presentation techniques, for example, scrolling information, separating overall and detailed views of information, distortion to highlight the interested information, suppression to balance details and relevant contexts, zoom and pan, semantic zoom, and so on. A common objective of these techniques is to conquer the limitations of available visual space and time. Chuah and Roth (1996), with a specific interest in the user interaction perspective of presentation, propose a taxonomy to outline typical interaction tasks in regard to graphical, set and data operations. Additionally, to support visual information seeking, Shneiderman (1996) introduces a presentation mechanism - “overview first, zoom and filter, then details-on-demand”, which provides design guidelines particularly useful for dealing with information exploration and retrieval. This mechanism has been widely adopted in the implementations of standalone visualization systems and visualization subsystems embedded in other systems like business intelligence and reporting systems, knowledge management systems, and ubiquitous systems. From the perspective of storytelling, Segal and Heer (2010) examine visualizations from a variety of sources (such as digital journalism, online blogs, videos and the extant research in the field of data visualization summarized), and synthesize them into seven visual narrative forms: magazine style, annotated chart, partitioned poster, flow chat, comic strip, slide show, and animation.

By exploring the support for information presentation, we identify five general presentation models. An information presentation model defines the way of how the represented visual structures of data are organized and transformed into views that users can display, manipulate, interact with and leverage. Typical visual structures are spatial substrate, marks (including points, lines, areas and volumes), connections, enclosure, retinal properties and temporal encoding (Card et al., 1999). It is through the views that the useful and effective information embedded within the data is eventually conveyed to the users and aid their decision making and problem solving. Based on our literature review on implementations of information presentation, we identified five popular presentation models, that is, landscape, semantic layering, nested spaces, sequential scenes, and integrated presentations. Among
them, the landscape model is the only single layer presentation paradigm while all others are multi-layer multi-dimension paradigms. Figure 2.6 compares these models in terms of the overall complexities involved and development efforts expected. It deserves to be noted that these models fulfill many common presentation requirements, but may not support every individual requirement for presentation.

Figure 2.6 Essential presentation models, from author’s creation (Bai, White & Sundaram, 2015)

**Landscape**

A landscape (also namely single layer landscape) is essentially a two-dimensional or three-dimensional display canvas on which users can present various visual representations of their interested data and navigate through the representations by scrolling, zooming, panning and/or rotating. Applications of this model fall into two types: bounded and unbounded landscapes. The key difference between them is the number of visual representations the landscape is able to incorporate. As its name suggests, a bounded landscape has a clear boundary of the available display canvas and hence allows presenting a limited number of visual representations. Typical data analytics and reporting systems with bounded landscapes are Microsoft Excel, SQL Server Reporting Services, and SAP Crystal Reports. Bounded landscape is also popularly deployed in most mobile-based applications. In comparison, an unbounded landscape is a never-ending visual space, which allows an unlimited number of visual representations.
**Semantic Layering**

A semantic layering model comprises a series of landscapes, each of which is used to reflect different levels of details and/or features of the underlying data. With semantic layering, users may navigate through the landscapes to show details on demand by zooming in and out. Typical applications of this model are Microsoft Bing Maps, Google Maps, bifocal display (Schaffer et al., 1996), hyperbolic display (Keahey & Robertson, 1997) and pliable surfaces (Carpendale, Cowperthwaite & Fracchia, 1995). Depending on whether the landscapes are of the same type (bounded or unbounded), this model may be further instantiated into two forms: symmetric and asymmetric semantic layering. If the model employs both bounded and unbounded landscapes, it is then an asymmetric semantic layering model. Otherwise, it is in the symmetric form.

**Nested Spaces**

A nested spaces model defines a virtual space that contains multiple nested and/or parallel landscapes with each one (either bounded or unbounded) presenting an independent visual world. Users may navigate through the landscapes by zooming, panning and/or rotating. It is often applied to present high-dimensional spaces. Example applications of this model are n-dimensional worlds (Feiner & Beshers, 1990) and Microsoft’s deep zoom image viewer (Deep Zoom, 2015).

**Sequential Scenes**

The sequential scenes model organizes and presents visual representations via a list of landscapes, which can be displayed in chronological order or by subject or in any other logical sequence. When it deploys a chronological axis, it can use any time scale, such as linear scale or logarithmic scale, depending on user requirements for grouping the visual representations. It can also use a single unit of time or mix up multiple time units. An example visualization presented based on the logarithmic scale is the Sparks’ Histomap (Sparks, 1931). Typical applications deploying this model with a linear scale are Tableau (Tableau, 2015) and Gapminder (Gapminder, 2015). Moreover, the landscapes may also be displayed in an animated fashion. For example, the animated sequential scenes model is used in ArcGIS (ArcGIS, 2015) to create an animation of a series of map frames.
Integrated Presentation

An integrated presentation model combines a set of preselected presentation models by defining the communication and cooperation among them. Compared to previous models, it is normally employed to support more sophisticated presentation requirements. Typical applications of this model are often seen in systems offering effortless layout transformation, for example, CA Xtraction (Xtraction, 2015) and Microsoft SharePoint (Microsoft, 2015b).

Depending upon complexities and requirements of a visualization problem, users may adopt one or more of the above models to aid in information presentation. The effectiveness of each selected model is determined by how well it fits to presentation requirements of the problem and the cognitive characteristics and preferences of the users.

Based on the review of IVS and their support for information representation and presentation, we discuss the system related challenges with effectively supporting visualization processes.

2.6.4 Challenges with Supporting Visualization Process

Many existing visualization systems tend to provide reasonably good support for the information representation process. However, they are still weak when it comes to developing flexible visual presentations for addressing the changing visualization requirements raised by dynamic visualization purposes, stakeholders and contexts. A common issue with implementations of information presentation in many visualization systems is that information presentation is tightly coupled with, even embedded within information representation. Such weakness is closely associated with, but not limited to, the following common problems:

- Lack of presentation options
- Lack of support for presentation transformation
- Lack of support for presentation integration at both model and instance levels
- Lack of support for reusable presentation models
- Lack of support for turning visual contents into a vivid story to communicate with and influence stakeholders.
The flexibility required for the information presentation process can be reflected from the system support for (1) creating presentation models, (2) combining visual presentations generated from different presentation models, (3) layering visual contents, (4) semantic zoom and navigation, (5) transforming visual presentations from one type to another, and (6) presentation model reuse. Keeping these requirements in mind, we evaluated the information presentation support of a series of visualization systems (Table 2.5). The assessment results clearly indicate the weak support for flexible information presentation.

Based on the review of visualization problems, stakeholders, purposes, contexts and systems, we summarize our research problems, issues and requirements pertaining to these perspectives in the subsequent section.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Excel</td>
<td>weak support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server Reporting Services</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP BusinessObjects</td>
<td>weak support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Crystal Reports</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Xcelsius</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP BW</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSS Clementine</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS Enterprise Miner</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tableau</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xtraction</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIMMS</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lavastorm</td>
<td>strong support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5  A review of the system support for information presentation
2.7 Problems, Issues and Requirements

Based on the literature review, we classify the identified problems, issues and requirements (PIR) into five categories: visualization, stakeholder, purpose, context, and system (Table). The following subsections 2.7.1-2.7.5 elucidate these PIR categories in detail, which lays a foundation for presenting a summary of these categories in subsection 2.7.6.

<table>
<thead>
<tr>
<th>PIR Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization</td>
<td>Any problems, issues and requirements associated with the data and visualization paradigms involved in decisional problems.</td>
</tr>
<tr>
<td>Stakeholder</td>
<td>Any user-side problems, issues and requirements that may impede stakeholders from leveraging the power of visualization tools and accomplishing their purposes.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Any problems, issues and requirements raised by the changing nature of visualization purposes.</td>
</tr>
<tr>
<td>Context</td>
<td>Any problems, issues and requirements raised by the changing dynamic decisional and situational visualization contexts.</td>
</tr>
<tr>
<td>System</td>
<td>Any problems, issues and requirements specific to visualization systems and their support for other PIR categories.</td>
</tr>
</tbody>
</table>

Table 2.6 Categories of problems, issues and requirements

2.7.1 Visualization-Related

Problems & Issues

As we discussed in subsection 2.2.3, the visualization-related problems are essential around five areas: (1) high data complexity, (2) large data volumes, (3) dynamic visualization paradigms, (4) visualization misuse, and (5) visualization quality and aesthetics (Bai, White & Sundaram, 2009b). These problems are associated with two important issues: decisional problem complexity, and visualization fit. The former is about managing and resolving the complexities involved in a certain decision problem. Such complexities are often incurred due to the challenges with (1) visualizing high-dimensional complex data and (2) dealing with multiple combined data types and large data volumes. They may also be caused by the
dynamic paradigms involved in the decision problem. Failing to address these complexities can prevent stakeholders from obtaining useful and relevant information from the underlying source data.

The latter issue of visualization fit is concerned with applying the right visualizations to solve the right user problems and visualization purposes for the right people under the right contexts. Different visualization techniques tend to reflect different features of source data and no such visualization technique can present a complete view of all possible features (Chi et al., 1997). It is important that the right visualization techniques be selected for the data and decision problems. Addressing this issue is crucial to conquer the challenges relating to visualization misuse. This issue also reveals the importance of a comprehensive understanding of both the strengths and limitations of various visualization alternatives.

**Requirements**

To address the visualization-related problems and issues, visualization techniques need to be designed in a flexible way by which users can “see more information, more quickly, with more comprehension” (Adnan, Noor & Aripin, 2003). This flexibility requirement can be further explained from the following three perspectives. Firstly, visualization techniques should be able to encode the underlying source data in a format maximizing the amount of useful and relevant information while minimizing the space required. In addition, they should present the information in such a manner that users can easily detect, comprehend and interpret the message delivered. Secondly, visualization techniques should appropriately cope with the complexities embedded in the large volumes of high-dimensional data such as transaction data records of credit card payments. They should enable users to “navigate efficiently to retrieve specific data, to explore data surrounding a topic of general interest and effortlessly browse the wider data world” (Jern, 1999). As high-dimensional data are quite popular in our daily lives (Keim, 2002), addressing data complexities has become an urgent need in the field of information visualization. Thirdly, due to the complexities involved in real-world user problems, users often need to apply multiple visualization techniques to a single user problem. Therefore, visualization techniques should offer flexible interfaces to enable the integration, cooperation and communication among them.
2.7.2 Stakeholder-Related

Problems & Issues

As we pointed out in subsection 2.3.4, there are two critical problems with visualization stakeholders: the limited information processing capabilities and the lack of prior knowledge. Both problems are intimately associated with the issue of the changing nature of visualization stakeholders. More specifically, the stakeholders of a visualization may change over time, which, in turn, may lead to changes in visualization requirements. The visualization requirements of the new stakeholders might be different from those of the previous stakeholders. Even if the stakeholders remain unchanged, their visualization requirements may vary over time due to (1) the changing problem/task requirements that the visualization attempts to support and (2) the changing proficiency levels of the stakeholders in terms of using the underlying visualization system. For example, compared to an expert-level visualization stakeholder, a beginner-level stakeholder may not have enough prior knowledge to operate a visualization system and interpret the produced visualizations (Chen, 2005), and hence needs the system to provide more systematic support for how to manipulate visualizations. When the stakeholder becomes more proficient with using the system, he may need more support for customizing visualization to complete sophisticated tasks.

Requirements

To address the stakeholder-related challenges, visualization design should concern the limitations of human memories and cognitive characteristics of stakeholders. For example, if beginner-level stakeholders are unlikely to process large chunks of relevant data at a time, it is important that visualizations offer decomposed views to prevent them from being overwhelmed with information. This requirement is to ensure the stakeholders can detect the intended information carried by visualizations. Moreover, how well visualizations fit the cognitive characteristics of stakeholders may largely affect the effectiveness of the visualizations and the performance of the stakeholders.

Additionally, the design and implementation of visualizations are closely coupled with the mental tasks and attributes associated with different learning development stages. This is because visualization requirements may be significantly affected by the purpose(s) and
cognitive characteristics of stakeholders, which, in turn, can be reflected and justified by the mental functions relating to the stakeholders. More specifically, visualization design and implementation need to concern to what extent the viewers rely on clearly defined decision making rules or task instructions, how well they are aware of the underlying problem situations, how easily they can recognize similarities between the problem under investigation and problems resolved in the past, how accurately they may identify and understand the relevant task requirements from the similarities, and how effectively they can monitor their own performance. This also reveals the importance of the support for visualization system personalization and customization. The ultimate goal is to assist stakeholders with their transition from beginners through to experts.

Furthermore, visualization system implementation should provide sufficient support for stakeholders to operate the system and be aware of the underlying application context. It requires user-friendly and easy-to-use interfaces and context support to minimize the time and effort required for learning the systems. This requirement also helps to avoid visualization misuse and the perceptual and cognitive capabilities of stakeholders (Ware, 2004; Spence, 2007). Additionally, visualization system implementation should consider the required flexibility and scalability to be able to handle the changing nature of visualizations caused by stakeholders.

2.7.3 Purpose-Related Problems & Issues

The purpose-related problems are mainly associated with consistently aligning visualization purposes at different levels for a decision problem, as presented in subsection 2.4.3. Such problems expose two important issues around stakeholder/visualization purposes. The first one is concerned with the changing nature of visualizations caused by purposes. The stakeholder/visualization purposes may vary over time, which, in turn, may cause the changing visualization requirements and restricted visualization effectiveness.

The second issue is associated with visualization fit, which reflects (1) how well task and domain level visualization purposes can be aligned and (2) how well the visualizations
selected for the underlying decision problem can fulfill the identified purposes. The extent of visualization fit significantly affects the visualization effectiveness and stakeholder satisfaction. It even decides (1) the length of the period during which the visualization is effective for the stakeholders and (2) when new visualization requirements appear.

Requirements

To address the purpose-related problems and issues, visualization design and implementation should offer sufficient system flexibilities in order to accommodate the changing nature of visualization purposes. As the purposes of a visualization involve both domain and task perspectives, purpose changes may occur at both perspectives. Furthermore, the change to one purpose perspective often requires corresponding changes to other perspectives. It is important that visualization techniques and systems can handle such purpose changes and their related visualization requirement changes.

2.7.4 Context-Related

Problems & Issues

As discussed in subsection 2.5.3, the major context-related problems are around the diversity and dynamics of visualization contexts. These problems are tightly associated with the issue - the changing nature of visualization contexts. Nevertheless, there is a lack of system support for (1) dealing with diverse and changing visualization contextual factors and (2) developing and adapting visualizations to address the corresponding changing visualization requirements.

Requirements

To resolve or relieve context-related problems, visualization design and implementation need to provide adequate flexibilities to handle the changing nature of visualization contexts. Visualization systems should offer sufficient personalization and customization mechanisms in order to enable stakeholders to modify the visualizations to better suit their requirements. The visualization systems should also provide appropriate adaptability mechanisms to assist the stakeholders with their transition from beginner levels through to master/expert levels.
Specifically, visualization system design and implementation should accommodate requirements for stakeholders at different skill levels. For stakeholders at a novice level, the system should offer them more within-context support such as the minimum system inputs they must provide, hints for system functionalities they are interested, solution templates for approaching typical problems, and so on. In contrast, for stakeholders at higher proficiency levels, the system needs to provide them more control and flexibility so that they can customize and adapt the system according to the features and requirements of the problem under investigation.

Moreover, stakeholders at beginner levels can generally deal with smaller chunks of data at one time and thus normally requires visualization designs containing the support/guidance for basic operations to accomplish a particular task. Compared to them, stakeholders with higher levels of expertise are often able to process relatively large chunks of data. They may not need visualizations to provide basic operation guidance but rather the support for more complicated tasks such as advanced information analysis. To realize these preferred features, visualization system implementations should incorporate mechanisms and functionalities of system personalization and customization.

### 2.7.5 System-Related Problems & Issues

There are two major problems associated with visualization systems. The first one is the lack of system support for developing effective visualizations that fit the underlying decision problem, stakeholders, purposes and contexts. This problem is caused due to the closely coupled visualization representation and presentation in system design and the lack of support for addressing the complexities involved in information visualization. Many visualization systems tend to provide weak support for handling the complexities incurred by large data volumes, complex and combined data types, and paradigm dynamics. This leads to the need of visualization systems supporting flexible visual contents integration and visualization techniques integration /customization/ transformation.
The second problem is the lack of system support for sustaining the effectiveness of visualizations under purpose/stakeholder/context changes. This occurs due to insufficient system support for the changing nature of visualization. The effectiveness of a certain visualization is affected by its capability to match the underlying visualization purposes, the decisional problem context within which it is applied and the situational context of the stakeholders. Such match may vary when visualization purposes, contexts and stakeholders change over time. However, many visualization systems tend to provide weak support for adapting to the changing purposes/contexts/stakeholders and the related changing visualization requirements. In other words, they are weak with regard to visualization transformation and customization. This requires visualization systems to enable (1) the flexible design and implementation of visualization models and (2) the flexible transformation of the models.

The system-related problems reveal a fundamental issue of the flexibility with visualization system design and implementation. Visualization systems need to be developed in a flexible manner to address or relieve the problems and issues associated with the visualization fit, changing nature, and complexities of visualizations. This system flexibility issue can be further decomposed into a set of functional issues, that is, flexible visualization creation, instantiation, manipulation, integration, customization, execution, modification, and enhancement.

Requirements

To address the system-related problems and issues, visualization systems should enable stakeholders to flexibly design and develop purposeful visualizations and sustain the purposefulness and effectiveness under stakeholder/purpose/context changes. In particular, they should allow stakeholders to create new visualizations from scratch and/or by reusing existing visualization components. They should support the integration of the visual contents generated by different visualization techniques to produce a rich view of the underlying source data. They should enable stakeholders to flexibly modify/customize/enhance visualizations. They should support transforming visualizations from one type to another in a flexible and seamless manner. Implementing these system-related requirements ensures stakeholders to visualize the same set of data through different visualization techniques and observe different features/views of the data. They also lay a good foundation for us to
formulate the system requirements for information representation and presentation for purposeful visualization systems in Chapter 5. These requirements are further explained as follows:

**Visualization Creation:** The changes involved in visualization contexts often lead to the variation of visualization requirements and even cause the same visualization to become irrelevant to the new context and requirements. In such circumstances, stakeholders may need to build new visualization solutions to their problem. Accordingly, visualization systems should enable stakeholders to create new visualizations from scratch or based on existing reusable visualization components.

**Visualization Modification/Enhancement:** A visualization, which can address a certain problem, task or purpose under a specific context, may not achieve the same level of effectiveness when the problem, stakeholder and/or purpose contexts change over time and space. Consequently, visualization systems need to allow stakeholders to flexibly modify and enhance visualizations to suit the changing requirements. Specifically, stakeholders should have the access to adjusting and/or replacing data models representing their interested problem and visualization models / techniques involved in the existing solution, for example, selecting the desired visual representations, changing the color or the hue, modifying model input/output parameters, etc.

**Visualization Integration:** To enable stakeholders to adapt visualizations under contextual changes, visualization systems need to provide at least three levels of support for solution integration, that is, low-level source data integration, medium-level visualization model integration (integrating the outputs generated from different visualization techniques), and high-level presentation/visualization integration. The purpose of visualization integration is to present a rich view of the underlying data. Visualization techniques often have their specific focus on handling particular types of data and reflecting particular features of the underlying data (Chi et al., 1997). In other words, no single visualization solution can address all data types, visualization purposes and fit in various contexts. Integrating multiple visualization techniques thus becomes a natural and effective way to enable stakeholders to explore more features of their data.
**Visualization Transformation:** Besides creating, modifying and integrating visualizations, visualization transformation is equally important for maintaining visualization effectiveness. Visualization systems should enable stakeholders to transform visualizations from one type to another in a flexible, seamless and efficient manner. This helps them to visualize the same set of data through different angles/perspectives via different visualization techniques.

### 2.7.6 Summary

Building on top of the review in subsections 2.7.1-2.7.5, we summarize the key problems, issues and requirements that our research attempts to address, and outline their connections in Figure 2.7. Keeping these problems, issues and requirements in mind, we proceed to discuss the strategy adopted for guiding and evaluating our research in the following chapter.
Figure 2.7 A summary of problems, issues and requirements
CHAPTER 3 RESEARCH METHODOLOGY

This chapter aims to elaborate on the methodology adopted for guiding and evaluating our research of purposeful effective visualizations and systems. To fulfill our research methodological requirements for literature review, theory building, system development and artefacts validation, we adopted a multi-methodological approach. Multi-methodological research approaches have long been recommended for guiding information systems (IS) research and deal with its intrinsic complexities (Nunamaker, Chen & Purdin, 1991; Morrison & George, 1995; Mingers, 2001; Adams & Courtney, 2004; Hevner, March, Park & Ram, 2004; Cao, Crews, Lin, Deokar, Burgoon & Nunamaker, 2006). A main reason for this is that each involved research method may produce positive complements to others and benefits of the integrated research approach are greater than the sum of those of its parts (Nunamaker et al., 1991; Burstein & Gregor, 1999; Mingers, 2001; Cao et al., 2006).

There are many research frameworks of this kind that have been proposed in past few decades, for example, March and Smith’s (1995) two-dimensional research framework, Hevner et al.’s (2004) information systems (IS) research framework, and Adams and Courtney’s (2004) DAGS framework. Nonetheless, many representative multi-methodological approaches are still flawed to some extent. This motivates us to design and develop a research framework that integrates the strengths of representative multi-methodological research frameworks and remedies their deficiencies. The eventual goal of introducing the integrated research framework is to provide more guidance for our research.

Following the integrated research framework, our research on purposeful effective visualizations and systems yields a set of outputs: concepts, models, frameworks, architectures, and system implementations. These research artefacts will be explicated in the chapters 4-7. Apart from the support for constructing these artefacts, the integrated research framework also guides us to validate them. Details of the artefacts evaluation will be presented in chapter 8.
This chapter is organized as follows. We first justify the choice of adopting a multi-methodological approach for guiding our research in 3.1. We then proceed to examine the strengths and weaknesses of some representative multi-methodological research approaches in 3.2. To address the weaknesses associated with these approaches, we design and propose an integrated multi-methodological research approach in section 3.3. Finally, we discuss the application of the proposed research approach through leveraging it to guide our research in defining, designing, developing and evaluating purposeful effective visualizations and systems in section 3.4.

### 3.1 Methodological Requirements

Our research on purposeful effective visualizations and systems falls under the general umbrella of information systems. Multi-methodological research approaches are considered most appropriate for us due to the requirements originated from the IS domain and those raised by our research in particular.

IS research is complex and dynamic in nature. It involves a variety of research areas such as the design, development and delivery of general/specific information systems, and the impact of deploying information systems in organization and society (Keen, 1987). It also produces a variety of research artefacts, for example, models, frameworks, processes, tools, languages, system components or applications (Burstein & Gregor, 1999). Moreover, it incorporates different dimensions of real world situations, materials and dynamic social and personal research contexts (Mingers, 2001). Such diverse research interests and artefacts lead to high complexities of IS research (Cao et al., 2006). As a result of this, deploying a single-methodological approach is often inadequate to tackle the complexities involved in IS research and not to mention develop conclusive solutions to a research problem (Galliers & Land, 1987). Though single-methodological approaches tend to provide reasonable support for particular types of research in IS, they are weak when it comes to dealing with the complexities involved in IS research. No single-methodological approach can serve as “the pre-eminent research paradigm” in the field of IS (Nunamaker et al., 1991), which, in turn, raises a natural requirement for integrating and adopting multiple research methods (Mingers, 2001; Cao et al., 2006).
Furthermore, there is a natural match between our research and multi-methodological approaches. To identify the problems, issues and requirements about visual decision support, there is a need for a proper observation method. To address the identified problems, issues and requirements, effective concepts, processes, system frameworks and architectures need to be developed, which reveals the requirement for adopting an appropriate theory building approach. To prove the validity of the proposed theories, a working prototypical system should be implemented, which raises the requirement for deploying an appropriate system development methodology. Moreover, both the proposed theory and prototype need to be validated and refined through being applied to real-world scenarios, which leads to the requirement for a suitable experimentation methodology. Nevertheless, a single method research approach (e.g. theory building, experiment, or system implementation) often specializes in guiding a particular type of research task. This in turn, exposes that it is difficult for one single method research approach to address all our research requirements, and hence the need of an appropriate multi-methodological approach.

<table>
<thead>
<tr>
<th>General Benefits</th>
<th>Examples</th>
</tr>
</thead>
</table>
| Combined results deepen the understanding of a research problem.                | ▪ Identified inconsistency of results from different methods and led to new insights and modes of analysis (Kaplan & Duchon, 1988)  
▪ Obtained different understandings of the same evidence from different research methods (i.e. interpretive analysis and positivist analysis) (Trauth & Jessup, 2000) |
| Triangulated results offer the validation of research outcomes and interpretations.| ▪ Triangulation of data alerted for potential analytical errors and omissions (Kaplan & Duchon, 1988) |

Table 3.1 Benefits of adopting multi-methodological approaches, adapted from Cao et al. (2006)

With a multi-methodological approach, each involved research method may produce positive complements to others and benefits of the integrated research approach are greater than the sum of those of its parts (Nunamaker et al., 1991; Burstein & Gregor, 1999; Mingers, 2001; Cao et al., 2006). For instance, based on the study of a computer-based detection deception system, Cao et al. (2006) prove the existence of such positive interactions and complements, and demonstrate the benefits of a multi-methodological approach through examining how
system evaluation activities may interact with theory testing activities. With an appropriate multi-methodological approach, researchers are more apt to achieve a good understanding of the research problem and validated research outcomes. Examples of such benefits are presented in Table 3.1.

The above general domain and specific research related requirements and the potential benefits behoove us to explore and examine the existing support for developing and applying multi-methodological research approaches.

### 3.2 A Review of Multi-methodological Research Support

Despite the need of adopting multi-methodological approaches to deal with the intrinsic complexities of IS research, integrating multiple research methods is not easy to accomplish. Mingers (2001) points out four common problems involved in this: philosophical problems (e.g. paradigm incommensurability); cultural problems like the negative effect of particular organizational culture that disturbs the implementation of a certain multi-methodological approach; psychological problems such as difficulties about some researchers feeling uncomfortable with adopting multiple research approaches; and practical problems. Many researchers have attempted to resolve the difficulties with developing and adopting multi-methodological research approaches. For example, Mingers and Brocklesby (1997) introduce a framework for mapping the characteristics of different methodologies by assessing how each methodology facilitates researchers to appreciate, analyze, explore and action to problems and issues with social, personal and material dimensions. Mingers (2001) identify and demonstrate five typical ways to integrate multiple research methodologies, that is, sequential, parallel, dominant, multi-methodology, and multi-level integrations (Table 3.2). IS studies employing multiple research methods are illustrated in Table 3.3.
### Table 3.2 Five ways to integrate multiple research methodologies, summarized from Mingers (2001)

<table>
<thead>
<tr>
<th>Type of Multiple Methods Integration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>Research methods involved are carried out in sequence with results from one feeding into the later one.</td>
</tr>
<tr>
<td>Parallel</td>
<td>Multiple research methods are adopted in parallel with results feeding into one another.</td>
</tr>
<tr>
<td>Dominant</td>
<td>One research method is employed as the main approach with contributions from other methods.</td>
</tr>
<tr>
<td>Multi-methodology</td>
<td>A set of research methods are integrated for supporting different paradigms and tasks.</td>
</tr>
<tr>
<td>Multi-level</td>
<td>Research is conducted simultaneously at different levels within an organization through using different research methods.</td>
</tr>
</tbody>
</table>

### Table 3.3 Typical IS studies adopting multi-methodological research approaches, adapted from Cao et al. (2006)

<table>
<thead>
<tr>
<th>Multiple Methods Integration Type</th>
<th>Typical IS Study</th>
<th>Methods Involved</th>
</tr>
</thead>
</table>
| Sequential                       | Markus (1994) – how and why managers use electronic mails | ▪ Positivist approach: survey  
▪ Interpretive approach: inductive and interpretive analysis on interview and written comments |
| Parallel                         | Trauth and Jessup (2000) – Use group support systems to facilitate employee discussions on gender equity in a university | ▪ Positivist content coding and interpretive open coding on the sample data set from a survey |

To effectively guide IS research, many multi-methodological approaches have been proposed in past few decades, for example, the two-dimensional research framework (March & Smith, 1995), the IS research framework (Hevner et al., 2004), and the DAGS framework (Adams and Courtney, 2004). Many representative multi-methodological approaches are nevertheless still flawed to some extent. We examine the strengths and weaknesses of these multi-methodological approaches in the following subsections. Moreover, this review sheds a light on how our proposed integrated multi-methodological approach is developed.
3.2.1 Nunamaker et al. (1991)

To guide IS research activities, Nunamaker et al. (1991) introduce a multi-methodological research framework that is composed of four mutually complementary research strategies, that is, observation, theory building, systems development and experimentation. Observation aims to achieve a good understanding of the interested research problem and domain as well as provide support for tasks involved in other research strategies, for example, helping with the formation of new hypotheses to be tested for the experimentation strategy. Theory building strategy is to guide the design and development of new concepts, models and frameworks. When a new theory is developed, research can adopt experimentation and system development strategies to examine, validate and refine the theory. Among these research strategies, the systems development strategy plays a pivotal role by interacting with other strategies and linking them together.

To illustrate how this approach can be applied to guide IS research, Nunamaker et al. (1991) propose a five-step system development oriented research process. It starts with identifying and understanding a research problem, and proceeds to develop system architecture that requires researchers to identify and define system development goals, core system components and functionalities and interactions among these components. The following step is to analyze the requirements for implementing the architecture, for example, understanding the underlying implementation domain, selecting appropriate technologies for the implementation, designing alternative solutions and defining evaluation criteria to choose the most appropriate alternative. Keeping the requirements in mind, researchers develop prototypical system(s) to validate and refine the proposed theory and system designs, and then assess the prototype(s) against pre-defined research and system development objectives. Such evaluation results influence their understanding accumulated at previous steps, which may initiate a new iteration of the system development process.

Strengths and Weaknesses

A key strength of Nunamaker et al.’s (1991) framework is that the involved strategies can produce mutually complementary research outcomes, if used appropriately. For instance, a prototype, developed by following the system development strategy, contributes to the
validation of the theory proposed by applying the theory building strategy. Furthermore, Nunamaker et al.’s (1991) approach involves systems development as its central research strategy. Burstein and Gregor (1999) state that a research process including systems development tends to be more complete, comprehensive and dynamic. Systems development method has also been recommended for IS studies by many researchers, such as Nunamaker et al. (1991), Parker et al. (1994), Morrison and George (1995), and Burstein and Gregor (1999). It can even mitigate the gap between the technological and the social sides of IS research (Burstein & Gregor, 1999).

Besides the above strengths, the design of Nunamaker et al.’s (1991) research framework is flawed on three counts. First, it lacks a clear indication that a research process is generally initiated with observation. Researchers and practitioners should first carefully observe real world problems before they progress to other research tasks such as defining a research problem, building theories for addressing the problem, and developing a prototype to validate and refine the theory. Second, their framework fails to incorporate considerations about the underlying research environment such as the characteristics of the involved people/organization(s)/technologies and their impact on the research. These characteristics and impacts may significantly affect the selection of methods for implementing each research strategy. Third, this framework is presented inappropriately in the sense that the four strategies are demonstrated with inconsistent focuses. The observation strategy is accompanied with possible research methods (e.g. case, survey or field studies) and the same thing applies to the experimentation strategy. However, the systems development strategy is presented with only possible research activities (e.g. prototyping or technology transfer), while the theory building strategy just outlines potential research outputs (e.g. conceptual frameworks, methods or models). This brings difficulties to identify and differentiate research methods, activities and outputs for each research strategy. This framework could be better organized through deploying an appropriate and consistent presentation pattern.

Despite the above deficiencies, Nunamaker et al.’s (1991) research framework is still a good starting point for us to develop our proposed approach. To seek ways to address its deficiencies, we review other representative multi-methodological approaches in the following subsections.
3.2.2 March and Smith (1995)

The two-dimensional IT research framework, proposed by March and Smith (1995), has a particular focus on highlighting common research activities and outputs involved in IS research. March and Smith (1995) opine that IS research is actually a combination of design science and natural science. Hence how well the research activities involved in design and natural sciences are performed determines the relevance and effectiveness of IS research. March and Smith (1995) identify four general research activities (i.e. build, evaluation, theorize and justify) and four broad types of research outputs (i.e. constructs, models, methods and instantiations). By integrating these research activities and outputs together, they develop the two-dimensional research framework that consists of sixteen cells with different research objectives, efforts, methods and evaluation strategies (1995).

Strengths and Weaknesses

As the building blocks of March and Smith’s (1995) framework are high-level generalizations of research activities and outputs, this framework itself is a conceptual framework and thus inappropriate to be used to provide practical guidance for conducting IS research. However, compared to Nunamaker et al.’s (1991) framework presenting research strategies with different levels of details, March and Smith’s (1995) framework demonstrates a good way to present potential research activities and outputs involved in IS research. This further motivated us to apply March and Smith’s (1995) framework to examine IS research adopting strategies involved in Nunamaker et al.’s (1991) framework. We realized that employing each research strategy might occupy multiple cells in March and Smith’s (1995) framework, i.e. conducting a set of research activities, and delivering certain research outputs. For example, the potential research activities and outputs in implementing systems development strategy are illustrated in Figure 3.1.
3.2.3 Hevner et al. (2004)

Building on top of March and Smith’s (1995) IT research framework, Hevner et al.’s (2004) propose a conceptual IS research framework to integrate two mutually complementary research paradigms: behavioral science, and design science. This research framework involves three essential building blocks, that is, research environment, IS research, and knowledge base. Among them, the research environment outlines how business requirements for addressing a certain problem should be defined, evaluated and positioned within the underlying problem context; the IS research component demonstrates the way of how behavioral science and design science paradigms may complement each other and fulfill the business requirements through developing/justifying theories and building/evaluating artefacts; such theories and artefacts are designed and developed based on the materials available in the knowledge base. This framework aims to assist researchers with understanding IS research, provide fundamental principles for guiding IS research, and illuminate the way to achieve relevant, effective and rigorous IS research.
**Strengths and Weaknesses**

Different from March and Smith’s (1995) framework, a key strength of Hevner et al.’s (2004) research framework is that it provides a comprehensive view of environmental factors that may affect IS research. Building on the research of Silver, Markus and Beath (1995), Hevner et al. (2004) generalize research environmental factors into three categories: people, organization, and technology. Understanding these environmental factors and their impacts may better help with conducting IS research and producing effective solutions to particular business needs. The strength of Hevner et al.’s (2004) framework exposes an opportunity to address the weakness that Nunamaker et al.’s (1991) framework fails to incorporate the impacts from the underlying research environment. Furthermore, similar to March and Smith’s (1995) framework, Hevner et al.’s (2004) framework tends to provide little practical guidance on how to conduct IS research.

### 3.2.4 Adams and Courtney (2004)

Based on the research of Nunamaker et al. (1991), Adams and Courtney’s (2004) introduce a DAGS framework that involves four research methodologies: design science, action research, grounded theory, and systems development. Among them, design science and grounded theory support theory building while systems development and action research aid in testing and refining the theories. The DAGS framework extends Nunamaker et al.’s (1991) framework by leveraging grounded theory and action research to support theory building and observation strategies respectively. Moreover, consistent with Nunamaker et al.’s (1991) framework, the DAGS framework also treats systems development as the core component to connect other involved methodologies all together. However, the DAGS framework lacks research methods to support the experimentation strategy of Nunamaker et al.’s (1991) framework.

**Strengths and Weaknesses**

The DAGS framework reinforces the importance of including systems development into a research framework to guide IS research. It also attempts to present a way to instantiate Nunamaker et al.’s (1991) framework. Nonetheless, it deserves to be noted that involving
design science as a component in the framework is quite confusing. This is because design science actually includes observation, theory building, systems development and experimentation (Peffers et al., 2008). In other words, design science already covers everything described in Nunamaker et al.’s (1991) framework. Accordingly, it is not proper to claim that the design science approach in the DAGS framework is merely used for theory building. Additionally, theory formation tends to be inappropriate to be included in systems development since it is just one step involved in theory building processes.

Understanding the strengths and weaknesses of the representative multi-methodological research approaches lays a good foundation for us to develop a research framework appropriate for guiding our research of Purposeful Visualizations and Purposeful Visualization Systems.

3.3 An Integrated Multi-Methodological Research Approach

Building on top of the existing representative approaches, we propose an integrated multi-methodological research framework, which remedies the deficiencies associated with Nunamaker et al.’s (1991) framework by integrating the strengths of other representative research frameworks (Figure 3.2). This integrated research approach aims to facilitate the development of practical research frameworks that integrate a set of mutually complementary research methodologies. As illustrated in Figure 3.2, framework comprises three presentation layers, that is, research strategies, research methods, and research outputs.

The research strategies layer outlines the high-level research strategies that may be adopted to conduct IS research. These strategies are observation, theory building, system development and experimentation, which are consistent with the strategies of Nunamaker et al.’s (1991) framework. Different from their framework, this integrated framework clearly indicates that IS research should start with careful observations of real world problems.
Figure 3.2 An integrated multi-methodological research framework, from the author’s creation (Bai, White & Sundaram, 2013b)
The observation strategy is concerned with deploying appropriate research and evaluation methods (such as case studies, survey studies or field studies) to achieve a good understanding of the interested research problem(s) and domain(s). Based on the understanding of the problem(s), researchers may apply the theory building strategy to define and develop new ideas, concepts, conceptual frameworks, models or methods. When a new theory is developed, they may adopt system development and experimentation strategies to examine, validate and refine their theory. With the systems development strategy, researchers design, develop and implement appropriate prototypical systems to validate their proposed theory. Experimentation can be conducted by employing, but not limited to, computer simulations, laboratory or field experiments in order to facilitate the development and refinement of the proposed theory, system designs and implementations.

Unlike Nunamaker et al.’s (1991) framework, this integrated framework presents research strategies in a consistent manner. To realize the strategies, the research methods layer shows possible methods that can be deployed to implement the related strategies, as illustrated in Figure 2. At this layer, researchers need to select appropriate research methods and specify the corresponding research processes/activities. The expected outcomes to be achieved by implementing each research method are presented at the research output layer. The idea of realizing and presenting research strategies in such way is inspired by the DAGS framework.

It is widely acknowledged that the evaluation of research artefacts plays a vital role in IS research (Hevner et al., 2004; Venable, Pries-Heje & Baskerville, 2012; Ostrowski & Helfert, 2012; Peffers et al., 2012). The artefacts must be rigorously assessed to ensure their utility, quality and efficacy (Hevner et al., 2004). Consequently, it is indispensable to integrate appropriate evaluation methods during the selection of research methods. Typical examples of research evaluation methods are case study, field study, static analysis, architecture analysis, simulation, functional/structural testing, informed argument, and scenarios (Hevner et al., 2004). Based on an extensive review of the research in information systems, computer science and engineering disciplines, Peffers et al. (2012) outline eight popular research evaluation methods: informed argument, expert evaluation, technical experiment, subject-based experiment, action research, prototype, case study, and illustrative scenario.
The selection of evaluation methods should carefully consider the nature and validation requirements of the research artefacts involved. That is, certain types of research artefacts may be more appropriate to be validated through specific evaluation methods than others. For instance, Peffers et al. (2012) state that some IS research prefer to validate the research outcomes through case studies while computer science and engineering studies are frequently validated through technical experiments.

It deserves to be pointed out that the evaluation methods adopted for a research strategy can contribute to the evaluation of the research artefacts produced by other strategies. For instance, the scenarios and static/structural analyses, though being used for the observation strategy, they assist in the evaluation of the prototypical systems that we implemented for validating the proposed concepts, models, frameworks and architectures of purposeful effective visualization systems.

Furthermore, the integrated framework incorporates Hevner et al.’s (2004) environmental factors that may affect how researchers and practitioners identify, define, interpret and address research problems and requirements. It also integrates the research output categorization of March and Smith’s (1995) framework and classifies research output into four categories: constructs, models, methods, and instantiations. In addition, the double-sided arrows connecting research outputs with research strategies indicate the mutual complements among the implementations of multiple research strategies.

To apply this integrated research framework to guide IS research, it needs to be instantiated by selecting relevant research strategies, adopting/developing appropriate research and evaluation methods and defining the expected research outputs. We explicate how this research framework is deployed to guide our research on purposeful effective visualizations and systems in the following section.

### 3.4 The Instantiated Research Framework

Bearing in mind the methodological requirements of our research discussed in section 3.1, we instantiated the generic integrated multi-methodological research framework by selecting all four research strategies, as illustrated in Figure 3.3.
Figure 3.3  An instantiated multi-methodological research framework, adapted from the author’s creation (Bai, White & Sundaram, 2013b)
For implementing the research strategies, we deployed a set of research and evaluation methods and specified the expected research outputs. It is imperative to involve appropriate evaluation methods into the research framework design. This is because validating research artefacts to ensure their relevance and rigor is a rather important requirement for all IS research (Hevner et al., 2004; McNaughton, Ray & Lewis, 2010; Gonzalez & Sol, 2012; Peffers et al., 2012). In this section, we discuss how our research on purposeful effective visualizations and systems was conducted and evaluated by research strategy in subsections 3.4.1 – 3.4.4.

3.4.1 Observation

The observation strategy is adopted due to three principal reasons. Firstly, it aims to enable us to achieve a thorough understanding of the domain of visual decision making, the challenges associated with decision problems and the existing support from information visualization techniques and systems. Secondly, it helps us to identify the focal problems, issues and requirements pertaining to (1) designing and developing purposeful effective visualizations and systems and (2) sustaining visualization effectiveness under diverse and changing visualization stakeholders, purposes and contexts. Thirdly, this strategy contributes to the evaluation of our research artefacts to ensure their rigor and validity.

To support implementing this strategy, we adopted five research and evaluation methods: literature review and reflection, informed argument, case study, scenarios, static analysis, architecture analysis, and expert evaluation. Among them, literature review and reflection are deployed to accommodate the first two objectives of adopting the observation strategy. Compared to this, the other methods are applied to evaluate the concepts, models, frameworks, architectures and implementations of purposeful effective visualizations and systems. A summary of how these research evaluation methods are utilized in our research is presented in Table 3.4.

By implementing this observation strategy, we expect to achieve an in-depth understanding of the research domain, identify the research gap, and perform the systematic evaluation of the research artefacts created for addressing the gap. This, in turn, contributes to the
implementation of the theory building strategy, which is expounded in the following subsection.

<table>
<thead>
<tr>
<th>Evaluation Method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Informed Argument</td>
<td>To evaluate the information representation and presentation capabilities of purposeful effective visualization systems by comparing them to other typical visualization systems.</td>
</tr>
<tr>
<td>Case Study</td>
<td>To study the visualization requirements of a case in an electricity distribution business, and demonstrate/evaluate how well the prototypical systems that we implemented for validating the proposed concepts, models, frameworks and architectures of purposeful effective visualization systems support these requirements.</td>
</tr>
<tr>
<td>Illustrative Scenarios</td>
<td>To assess system functionalities and applications of purposeful effective visualization systems through constructing a variety of scenarios.</td>
</tr>
<tr>
<td>Static Analysis</td>
<td>To examine the prototypical systems at unit, technology, and system levels.</td>
</tr>
<tr>
<td>Architecture Analysis</td>
<td>To appraise the fit of the frameworks and architectures of purposeful effective visualization systems into technical IS architectures.</td>
</tr>
<tr>
<td>Expert Evaluation</td>
<td>To validate and refine our research artefacts based on the suggestions and feedback collected from domain and system experts.</td>
</tr>
</tbody>
</table>

Table 3.4 Research evaluation methods selected for the observation strategy

3.4.2 Theory Building

The theory building strategy is selected to assist in developing the concepts, models, processes of purposeful effective visualizations and systems. To realize this strategy, we review the extant literature about visual decision support and ensure that our proposed artefacts incorporate the good practice of modeling and scenario planning, effective visualization design patterns and so on. An important prerequisite of theory building is the careful observation of the research domain and the relevant problems, issues and requirements. The theory building strategy attempts to model and define: what a purposeful effective visualization is; how exactly a visualization can become effective for diverse visualization stakeholders, purposes and contexts; how to apply purposeful effective visualizations to support decision making; how to design and develop such visualizations; and
how to sustain the visualization effectiveness when stakeholders, purposes and contexts change over time.

Furthermore, the evaluation methods, adopted for realizing the observation and experimentation strategies and validating the research artefacts, may help with refining the proposed theory of purposeful effective visualizations and systems. Meanwhile, the research outcomes of this strategy lay a theoretical foundation to underpin the implementation of the system development strategy. We detail the system development strategy in the next subsection.

3.4.3 System Development

The systems development strategy is applied to validate the concepts, models, frameworks and architectures of purposeful effective visualizations and systems through implementing prototypical systems based on the proposed theory. The experience gained through prototype development allows us to refine the proposed theory. To implement this strategy, we adopted evolutionary prototyping to guide the design and development of the prototypical systems. This system development approach is selected due to its widely acknowledged advantages. It is particularly useful to deepen the understating of system requirements and identify missing requirements (Davis, 1992; Lichter, Schneider-Hufschmidt & Zullighoven, 1994). It also minimizes the risks and uncertainties associated with a system development process and attempts to address them at earlier stages (Connell & Shafer, 1995; McConnell, 1996; Carter et al., 2001). Furthermore, it facilitates establishing regular communications between system developers and end users and delivering system prototypes with good user satisfaction (Connell & Shafer, 1995).

Following the evolutionary prototyping approach, a system development process starts with implementing confirmed and well-understood system requirements. Then, it continuously refines and adds more functionality to the system prototype based on the user feedback collected and the deeper understanding of the system requirements until user expectations are matched (Davis, 1992; McConnell, 1996). Therefore, the implementation of the system that is constructed to validate our proposed theory of purposeful effective visualizations is an
iterative and incremental process. Moreover, the evolutionary prototyping approach is used in conjunction with the evaluation methods adopted in our research. In this way, it improves the system functionalities or initiates a new development iteration based on the collected evaluation results, and triggers further iterations of research evaluation. For example, by examining the support of the prototype for the requirements from the case study, we may identify the missing functionalities to be implemented in the system, which could require a new iteration of system development. Any new insights into improving the theory building, gained from research artefacts evaluation and prototype development, may again contribute to enhancing the system implementation.

Besides the positive complements from the observation and theory building strategies, the system development strategy needs the support of the experimentation strategy to evaluate system implementations. This requirement is further expanded in the following subsection.

### 3.4.4 Experimentation

Experimentation is an effective way for IS researchers to explore and understand the complexities involved in a research area that is not fully understood and to identify the problems and issues for investigation (Vaishnavi & Kuechler, 2015). Our research adopted this strategy particularly to support the design and development of the prototypical system that enables us to create purposeful effective visualizations and sustain the visualization effectiveness when the underlying stakeholders, purposes and contexts change over time. To realize this strategy, we deployed three research evaluation methods: computer simulations, functional testing, and structural testing. A brief summary of why these methods were used in our research is provided in Table 3.5.

Among these evaluation methods, computer simulations are used to help us to identify appropriate software development technologies to implement prototypical systems. As each software development technology has its own strengths and limitations, finding the right technologies means that the selected technologies provide good support for implementing the system requirements of visualization creation, modification, transformation, integration and context adaptation. Creating computer simulations by applying different sets of software
development technologies allows us to choose the right technologies wisely, which saves the
time for the prototype development.

<table>
<thead>
<tr>
<th>Evaluation Method</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Computer Simulations</strong></td>
<td>To test and execute the prototypical system that allows creating purposeful effective visualizations and sustaining visualization effectiveness under stakeholder/purpose/context changes.</td>
</tr>
<tr>
<td><strong>Functional Testing</strong></td>
<td>To test and execute the functionalities of purposeful effective visualization systems with artificial data.</td>
</tr>
<tr>
<td><strong>Structural Testing</strong></td>
<td>To examine the internal structure of the prototype at unit, integration and system levels to ensure the robustness and stability of the prototype.</td>
</tr>
</tbody>
</table>

Table 3.5       Research evaluation methods selected for the experimentation strategy

Compared to computer simulations, the functional and structural testing methods aim to
examine the main functionalities and internal structure of the system prototype. More
specifically, in functional testing we create a variety of test cases against the prototype
functionalities for developing purposeful effective visualizations and maintaining visualization
effectiveness. For the structure testing, we test against the internal structure of the system
prototype at unit, integration and system levels in order to detect hidden issues within the
system. The test results obtained from functional and structural testing contribute to the
further improvement of the prototype functionalities and the refinement of the proposed
concepts, models, frameworks and architectures of purposeful effective visualizations and
systems. The details of how these evaluation methods were conducted in our research are
presented in chapter 8.

In this chapter, we discussed the methodological requirements of our research on purposeful
effective visualizations and systems. To accommodate these requirements, we explored the
existing representative multi-methodological research approaches, which lays a solid
foundation for us to develop the integrated multi-methodological research approach. We
instantiated this approach specifically for guiding the construction and evaluation of the
research artefacts produced by our research. These artefacts comprises concepts, models,
frameworks, architectures and system implementations of purposeful effective visualizations
and systems. We present these artefacts in detail in chapters 4 – 7 and discuss how they are
validated in chapter 8.
CHAPTER 4  PURPOSEFUL VISUALIZATIONS

In this chapter, we define and illustrate the concept of purposeful effective visualizations and depict the process of how they can be applied to accommodate various visualization purposes under diverse contexts. This is to address the following research questions raised in the previous chapters:

- What are purposeful visualizations?
- How can we apply purposeful visualizations to support various tasks in decision making?

More specifically, we present a formal definition of purposeful visualizations in section 4.1 as well as a model to help with understanding, analyzing and evaluating them in section 4.2. To demonstrate how such visualizations can be created and applied to support visual decision making, we propose a visual decision making process in section 4.3.

4.1 A Definition of Purposeful Visualization

*Purposeful visualizations* are visualizations that fulfill a particular *purpose* for one or more *stakeholders* within a certain *context*. The concept of PV highlights the status of a visualization in terms of its visual capability of achieving an intended purpose; it also accentuates the changing nature of visualizations caused by stakeholders, purposes and contexts. Every visualization can become a PV within a certain time frame and context, but its purposefulness may change when the surrounding visualization context is changed. This concept is further explained in the subsequent sub-sections.

4.1.1 Stakeholder

Typical stakeholders of PV are (1) visualization designers who develop, manage and evaluate PV, (2) viewers who apply PV to aid diverse visualization related tasks, and (3) presenters who act as an intermediary role of presenting PVs to inform/shape/change/reinforce mental models and behaviors of the viewers. Viewers may utilize PV to address their visualization related problems, issues, requirements, objectives, goals and/or tasks. The visualization requirements, personal characteristics/preferences, cognitive features, and prior knowledge
of the viewers may significantly affect the design decisions made by the designers. How well
the designers can understand and support these characteristics and requirements of viewers
directly affects the effectiveness of their visualizations. For example, it is difficult for the
designers to develop effective visual representations of spatial temporal data without a clear
understanding of how the viewers perceive space and comprehend time. The presenters’
interpretation and demonstration of PV may affect the viewers’ understanding of the
visualizations, especially those first-time viewers. Quite often, an individual may
simultaneously act in two or even all of these roles. Any visualization can become a purposeful
visualization for a certain group of stakeholders under a certain context.

4.1.2 Purpose

Purposes reflect what PV attempt to achieve. PV may be adopted to address any visualization
related problems, issues, requirements, objectives, goals and/or tasks. A general goal of PV is
to inform/shape/change/reinforce mental models and behaviors of their audience. PV may
be created to solve issues within a variety of application domains (such as
statistical/categorical data management, and digital library/document management),
support user activities involved in information navigation, retrieval, query, discovery, and/or
interpretation, and express personal thoughts/feelings/emotions on life, world, love, etc.
When visualizations are appropriately deployed for their intended purpose, quite often their
stakeholders can obtain illuminating, useful, relevant and actionable information from them
with less effort and time.

In general, the purposes of a certain visualization can be defined from four essential
perspectives, that is, visualization stakeholder, task, application domain, and visual
primitive/composite perspectives (Figure 4.1). The stakeholder perspective specifies the
visualization purposes from the angle of what objectives its stakeholders attempt to achieve
via the visualization within their specific context. The task perspective depicts the
visualization purposes from the angle of what user tasks the visualization aims to support.
The domain perspective describes the visualization purposes from the angle of what in
general the visualization is trying to fulfill within its typical application fields/context. The
visual primitive/composite perspective delineates the specific purposes of the fundamental
components involved in the visualization. The purposes from different perspectives mutually support one another and contribute to a more thorough and comprehensive understanding of visualization purposes. Defining the visualization purposes according to these four perspectives is quite helpful for creating an integrated visualization purpose model.

![Visualization purpose perspectives](Figure 4.1)

### 4.1.3 Context

Under a particular context, any visualization can become a purposeful visualization for a certain group of stakeholders. Understanding the decisional problem context within which the visualization is deployed is vital. Some examples of the decisional problem context could be relevant problem situations, time, space, social context, technological context (including hardware and software). Equally important is the situational context of visualization stakeholders, such as their cognitive styles, personal characteristics and preferences, prior knowledge, age, gender and even their mood when making decisions.

The identification and analysis of visualization purposes and contexts allow designers/viewers to define a set of requirements in terms of how their data should be visually represented and presented. Some of these requirements tend to be objective as they are often context independent. In other words, they are raised to address a certain visualization
problem or fulfill a particular visualization purpose, but are less dependent on the decisional problem context and the situational context of the visualization stakeholders. For example, to address the goal of forecasting future product sales, a possible objective requirement could be visualizing the trend of the product sales in previous years no matter who are the viewers. In contrast, there are also some subjective visualization requirements that are caused by the underlying context. For instance, to address the same visualization problem, some viewers may prefer a particular type of visualizations and some designers may prefer to present the information via certain layouts.

### 4.1.4 Visualization

From the perspective of cognition and interpretation, a visualization can be decomposed into a set of semantic layers, each of which presents different facets and information of the underlying data (Figure 4.2).

![Figure 4.2](image)

**Figure 4.2** A multi-layered view of visualization

This concept can be illustrated through an example portrayal of Napoleon’s March in Russia. In June 1812, Napoleon invaded Russia with a massive force of 422,000 troops. When the army retreated in December 1812, there were only about 10,000 men alive. The problem of visualizing this campaign is in the nexus of the historical, statistical and categorical domains where data are complex in the sense that spatial, temporal and multi-dimensional types of data are required to be visualized. In 1869, Charles Joseph Minard published a map to portray this campaign (Tufte, 2001), which plots six variables on a simple two-dimensional map. These
variables are locations along the route (latitude and longitude coordinates), the number of survivors at each location, army movement directions, temperatures, and dates. Building on top of this map, we reproduced different semantic layers involved in this visualization. As illustrated in Figure 4.3, the bottom layer simply displays a road map of the areas where Napoleon’s army went through towards Moscow. In comparison, the top three layers have a specific focus of illustrating the information about the army march with displaying different levels of details.

Figure 4.3 A multi-layered view of Napoleon’s March, from the author’s creation (Bai, White & Sundaram, 2011c)

From the perspective of visualization designers, the presented facets and information reflect the message that the designers intend to convey. From the perspective of viewers, these facets and information can be applied to represent their cognition and interpretation of the visualization. When designing and developing a visualization, designers may have a set of information that they would like to display via semantic layers. A semantic layer may convey exactly the messages that the designers intend to present. Nonetheless, it may also display
some misleading or even wrong information (i.e. unwanted information) which is displayed before the designers realize them. This often results in unpleasing, ugly, ineffective and/or bad visualizations.

As demonstrated in Figure 4.2, a designer acts as the producer of both intended and unwanted information, and the information is conveyed to all viewers via the semantic layers 1 – n. Although displaying the unwanted information is not expected by the designer, viewers may still observe the layers containing the misleading information. This implies that designers and viewers may have different visualization objectives and goals as well as different evaluation criteria when considering/performing visualization-related tasks (e.g. visualization design, development and assessment). In addition, it is worth noting that different viewers may perceive, recognize and understand different semantic layers involved in the same visualization and even different information within the same layer due to the differences among their cognitive styles, characteristics, prior knowledge of the problem domain, etc.

4.1.5 Effectiveness of Purpose Fulfilment

Visualization purpose fulfillment is concerned with how well the objective and subjective requirements raised by a certain decision problem can be addressed by the visualizations of interest. These visualization requirements help to reveal how designers or viewers will assess the visualizations, such as the evaluation criteria in use and their weights, how often they measure the visualization and the ratings they may give to the measures.

To achieve its intended purposes, a purposeful visualization needs to provide illuminating, useful, relevant and actionable information for viewers to address their visual visualization problems/issues/requirements. The viewers can obtain the information by creating, manipulating and interacting with the visualization and meanwhile observing, perceiving, interpreting and understanding the message conveyed. Their judgment on the effectiveness of the visualization is primarily affected by two factors. The first factor is the match between the semantic layers of the visualization and the visualization purposes of a viewer while the second is the effectiveness of a visualization presenter articulating and demonstrating this match. This will be further explicated in the following section.
4.2 Purposeful Visualization Model

To facilitate representing, understanding and communicating purposeful visualizations, we model what presents a conceptual-level structure of PV and point out a way to create/analyze/evaluate PVs. According to our definition of PV, the most important requirement for visualizations becoming purposeful is that they need to provide illuminating, useful, relevant and actionable information for facilitating viewers to achieve their intended purposes effectively.

4.2.1 Influential Factors of Purposefulness

The performance of the viewers observing, perceiving, understanding and interpreting visualizations within a certain amount of time is substantially affected by two factors. The first factor is the match between the semantic layers of a visualization and the visualization purposes of a viewer. In other words, whether a visualization means purposeful and effective for the viewer is decided by how well fulfilling his visualization purposes (defined from the stakeholder, task, application domain, and visual primitive/composite perspectives) can be supported by the useful facets and information of the underlying data conveyed through the semantic layers. The four purpose perspectives actually define both objective and subjective visualization requirements of the viewer. Fulfilling the visualization purposes is in fact satisfying these visualization requirements. The better the visualization purpose fulfillment can be achieved, the more effective and purposeful the visualization is for the viewer. The more easily the viewer can manipulate, perceive, understand and interpret the information delivered by the semantic layers, the better visualization fit and user satisfaction can be achieved.

The second factor is the effectiveness of the visualization presenter articulating and demonstrating the above match of visualization fit. To help a viewer to realize the match between the semantic layers of a visualization and his visualization purposes, the presenter plays an intermediary role for justifying the existence of this match and convinces the viewer of accepting what the visualization designer/presenter would like him/her to behave and/or
believe (e.g. taking a certain decision). The more effectively the presenter can present the visualization and the match, the better visualization fit can be achieved.

To demonstrate how a visualization can fulfill its intended purpose, we propose a purposeful visualization model in the following section.

### 4.2.2 Purposeful Visualization Model

The way to ensure the achievement of purposeful visualizations is to optimize the match between a visualization’s semantic layers and the visualization purposes of a viewer and ensure that both the visualization and the match are appropriately presented by an effective presenter (Figure 4.4).

![Purposeful Visualization Model](image)

As demonstrated in Figure 4.4, the design and development of a visualization may be largely affected by the designer’s understanding of the various perspectives of visualization purposes.
Meanwhile, the visualization purposes of the designer could be significantly influenced by the decisional problem context and situational context of the designer. When a visualization is created by the designer, its semantic layers may well satisfy, partially fulfill or sometimes even not match the objective and subjective requirements that are defined by the designer’s visualization purposes. To ensure the viewer can obtain the useful information conveyed by the semantic layers, the presenter learns about the meaning and implications of the visualization and understands the visualization purposes of the viewer, and then interprets his understanding of the visualization to the viewer. The viewer’s judgment on the visualization is again influenced by his own visualization purposes and context. The judgment on the same visualization given by different viewers may vary depending on how well their visualization purposes can be fulfilled by the semantic layers, and the judgment may change over time/domains due to the variations in the match.

4.2.3 PV Model Implications

Understanding the concepts involved in this PV model is important for both viewers and designers. When a viewer explores a variety of visualizations to fulfill his visualization purposes, the process of aligning the semantic layers with the subjective/objective visualization requirements (defined by the visualization purposes) may inspire the viewer to refine the existing visualization purposes to better express his needs. It may also lead the viewer to generate new visualization purposes and requirements. Meanwhile, this alignment process may help designers to identify and better understand the difference between what they think the visualization purposes of a viewer are and what the viewer really expects. Thus, the designers may be able to modify the visualization to better fit for the viewer’s visualization purposes. To apply purposeful visualizations to aid visual decision making, we propose a visual decision making process, which is delineated in the subsequent section.

4.3 Purposeful Visual Decision Making Process

The purposeful visual decision making (PVDM) process describes how purposeful contextual interactive visualizations can be developed and employed to address visual decision making problems. It is developed by integrating the ideas of information visualization frameworks
and reference models, decision making processes, purposeful visualization requirements, scenario planning, modeling and model management. Building on the work of Sinclair and Ashkanasy (2005), the PVDM process comprises both rational and naturalistic decision making processes which cooperate with each other in a harmonious manner (Figure 4.5). The rational process is formed in a structural format, while the naturalistic process is conducted automatically by decision makers with no consciousness. Combining the two different types of decision making processes may lead to better opportunities of integrating the strengths of two approaches and reflecting/guiding the real world decision making practice.

Figure 4.5 Purposeful visual decision making process, adapted from author’s creation (Bai, White & Sundaram, 2009b)

4.3.1 Rational Process

The rational decision making process consists of eight key stages, as illustrated in Figure 4.5. In general, the first two stages assist decision makers with understanding and defining the decision problem to be addressed, visualization purposes and how the effectiveness of purpose fulfillment will be assessed. These stages correspond to the intelligence phase of
Simon’s (1960) model. Then, the subsequent three stages focus on designing and developing alternative purposeful visualization solutions, which are consistent with the design phase of Simon’s (1960) model. Next, the evaluation and decision making stages help decision makers to assess how effectively the visualization alternatives can fulfill the purposes and then choose the optimal solution. These two stages are in line with the choice phase of the Simon’s (1960) model. Finally, at the solution implementation stage, the selected visualization solution is implemented to address the decision problem. The whole PVDM process may iterate if further problems or opportunities are detected at the implementation stage.

More specifically, the problem identification stage aims to identify and define the decisional problem, which is actually the gap between the existing problem situation (AS-IS) and the desired state decision makers attempt to achieve (TO-BE). A clear definition of the decisional problem ensures the decision makers to formulate a unified understanding of the problem situation, which, in turn, facilitates the communication of the problem. For example, the problem definition may include the descriptions of the boundary and time duration of the problem situation (Schoemaker, 1998; Chermack, 2005; Keough & Shanahan, 2008).

Then, based on the problem definition, the decision environment understanding stage requires identifying characteristics of visualization contexts, purposes and decision makers. Examples of such characteristics could be factors determining the appropriateness of adopting a particular analytical decision making process for the decision makers (Schoemaker, 1998; Chermack, 2004), and the culture and leadership status involved in organizational decision making context (Keough & Shanahan, 2008). Apart from the decision environment identification, equally important is to understand the potential impact of the contextual characteristics as they may affect later visualization solution design, development and evaluation.

Next, to develop appropriate alternative visualization solutions, the underlying raw data are initially cleaned and transformed to remove data irrelevant to support visual decision making at the information generation stage. This stage can be ignored if the raw data already possess good quality. At the information representation stage, the data are then visually encoded and transformed into certain visual structures so that the later data visualization can become easier, compared to visualizing the data without being properly represented. The information
presentation stage allows the visual structures to be transformed into views through which the illuminating, useful, relevant and actionable information can be delivered. A common feature shared by these visualization development stages is that they all involve tasks concerning scenario building, modeling and model management.

After the alternative visualization solutions are developed, decision makers measure the effectiveness of the alternatives against pre-defined assessment criteria at the evaluation stage. The evaluation results then guide the decision makers to select the solution that better addresses the decision problem at the decision making stage. Finally, at the solution implementation stage, the decision makers implement the selected solution to address their problem. As indicated by the double-sided arrows associated with each stage in Figure 4.6, the decision makers may conduct the rational process stage by stage and go back to any previous stages if needed.

To better model the real world visual decision making practice, the PVDM process incorporates the naturalistic decision making process to complement the analytical approach. We elucidate the naturalistic process in detail in next subsection.

4.3.2 Naturalistic Process

The rational process describes the ideal situation of decision making which simplifies or neglects the complexities involved in visual decision making (Langley et al., 1995). It is widely acknowledged that visual decision making may be significantly affected by many human factors such as emotion, intuition, inspiration, imagination, memory and even personal characteristics (Vessey, 1991; Vessey & Galletta, 1991; Eisenhardt & Zbaracki, 1992; Langley et al., 1995; Sauter, 1999). To better model the real world visual decision making practice, the PVDM process incorporates the naturalistic decision making process, which complements the rational process. Sinclair and Ashkanasy (2005) summarize four categories of influential factors, that is, decision characteristics, problem characteristics, personal disposition and decision making context. As illustrated in Figure 4.6, the naturalistic process involves these influential factors as well as their representative examples such as fear and greed.
The naturalistic decision making may take place at any rational decision making stages and interact with them in a seamless fashion. Such interaction may bring both positive and negative influences on visual decision making. For example, a decision maker with strong intuitive/judgmental skills and rich experience of solving the decisional problem of interest may accelerate the rational decision making process. However, the intuition and experience may also mislead the decision maker to take wrong decisions if the problem situation looks similar to but is actually quite different from the problem of which the decision maker has experience.

In this chapter, we proposed the definition of purposeful visualizations and the PV model to aid the understanding, analysis and evaluation of PV. We also demonstrated the application process of PV within the context of visual decision making. To support the development of PV and the PVDM process, we propose purposeful visualization systems, discuss the key requirements for developing such systems, and design and implement frameworks and architecture to prove the validity of the proposed theory in the following chapter.
CHAPTER 5 PURPOSEFUL VISUALIZATION SYSTEMS

In this chapter, we propose Purposeful Visualization Systems (PVS) to explicate the concepts of purposeful visualization introduced in the previous chapter and to address the complexities involved in visualization problems and support the changing nature of visualizations. PVS are purpose-driven, context-adaptable, visual-interactive platforms that help an individual to solve problems by utilizing data, decision/visual models, solvers and scenarios to create and manage effective visual compositions. The ideas and designs of PVS are inspired by the classical designs of decision support systems (Brennan & Elam, 1986; Ramirez, Ching & Louis, 1990; Sundaram, Chen & Srinivasan, 2001), information visualization reference models (Haber & McNabb, 1990; Card et al., 1999; Card & Mackinlay, 1997; Chi & Riedl, 1998; Spence, 2007; Bai, White & Sundaram, 2010a), the ideas of scenario planning (Schoemaker, 1993; Chermack, 2004; Chermack, 2005; Avin, 2007; Keough & Shanahan, 2008), and the concepts of model management lifecycle. PVS provide flexible support for (1) creating/manipulating/evaluating/improving/disposing visualizations and (2) maintaining visualization effectiveness under the changing and evolving visualization contexts.

This chapter elaborates on the fundamental system features, frameworks and architectures of PVS, and contributes to addressing the following research questions:

- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?

In this chapter, we specify the system requirements that distinguish PVS from other systems supporting visual decision making in section 5.1. To realize these indispensable system requirements, we propose the present PVS frameworks and architectures in sections 5.2 and 5.3 respectively.

5.1 Distinguishing Features of PVS

The distinguishing system features of PVS aim to support the requirements for visualization creation, modification, transformation, integration and adaption, which are summarized in subsection in subsection 2.7.6. PVS system features can be classified into three main categories: information generation, information representation, information presentation.
This classification is also aligned with the three purposeful visualization development stages shown in the PVDM process. We detail the requirements pertaining to each category in subsections 5.1.1 – 5.1.3.

**5.1.1 Information Generation**

Visualization stakeholders often need to deal with large volumes of source data that are relevant to understanding and resolving a complex decision problem. The source data could be derived from diverse data sources (i.e. various information systems such as SAP and Microsoft Bing maps) and stored in a variety of formats (such as Excel spreadsheets, Oracle/MySQL/SQL Server databases, and XML files). The requirement of information generation aims to transform the source data into a series of data sets, which contain only data useful and effective for addressing the decision problem. In other words, such data sets present high intelligence density through removing any noise (i.e. irrelevant and ineffective information) and conducting necessary data calculations/reconciliations. In addition, the structure of these data sets should facilitate the subsequent step of information representation. The requirements for supporting information representation are specified in next subsection.
5.1.2 Information Representation

The system requirements for supporting information representation are to ensure the extracted data are transformed and encoded into appropriate visual structures in a flexible manner. A visual structure defines a primitive visual object that can be used to represent the semantics of data. Some typical examples are spatial substrate, marks (including points, lines, areas and volumes), connections, enclosure, retinal properties and temporal encoding (Card et al., 1999). PVS should allow visualization stakeholders to flexibly create, modify, transform and integrate visual representations.

Representation Creation

The changing nature of visualization purpose, stakeholder and context leads to the variation of visualization requirements, which may cause visualizations to become ineffective or irrelevant under the changed visualization contexts. Consequently, stakeholders would need to create new visual representations to better address their interested problem. Accordingly, a visualization system should enable the stakeholders to create new visual representations in a flexible fashion. The stakeholders should be able to develop new representations from scratch or based on existing reusable representations.

Representation Modification/Customization/Enhancement

Visual representations, which fit a particular problem/task/purpose under a specific context, may not be able to achieve the same level effectiveness when visualization contexts change over time and space. Therefore, a visualization system should offer the capabilities for stakeholders to flexibly modify, customize and enhance visual representations so as to suit the changing requirements.

Representation Transformation

To match the cognitive styles and preferences of different stakeholders, visualization systems should enable visual presentations to be transformed from one type to another in a flexible, seamless and efficient manner. This is particularly helpful for stakeholders to highlight different features or perspectives of the underlying data.
**Representation Integration**

Visual representation integration requires seamlessly integrating multiple representations of data created by visualization techniques so as to present a rich view of the underlying source data. Due to the changing visualization purposes, contexts and stakeholders, visual representations are often required to reveal different features of the source data. However, visualization techniques often have their specific focus on handling specific types of data and reflecting particular features of the data (Chi et al., 1997). In other words, no single visualization technique can be effective for addressing all data types and all visualization purposes. Consequently integrating multiple visualization techniques within a single visualization system becomes a natural and effective way to enable stakeholders to explore more features of the source data (Hibbard, 1999).

**5.1.3 Information Presentation**

We argue that the key to realize flexible visual presentations in visualization systems is to decouple information representation and presentation. Separating presentation from representation has long been advocated due to its benefits to (1) understanding and assessing features of visualizations, (2) comparing across visualizations of a similar type, and (3) identifying opportunities to create new visualizations by trying different combinations of representations and presentations (Chi & Riedl, 1998; Chi, 2000; Carpendale & Montagnese, 2001; Spence, 2007). It is also a recommended design pattern of visualization systems due to the improved flexibility, reusability and extensibility (Heer & Agrawala, 2006). With regard to information presentation in particular, we identify the following fundamental requirements to support presentation flexibility and to guide visualization system design and development.

**Presentation Creation and Customization**

With a visualization system offering good presentation flexibility, stakeholders should be able to create a new presentation from scratch or by reusing and modifying existing presentations. Moreover, the system should allow stakeholders to modify, customize and enhance presentations. This is often accompanied by relevant representation level customizations,
and is quite helpful to accommodate different user preferences against the same presentation.

**Presentation Integration**

Presentation integration is to enable stakeholders to combine presentations that view the visual representations for various purposes. Similar to information representation, visual presentation techniques nowadays usually have their particular strengths in revealing certain features of data (Card et al., 1999). No single visual presentation technique is effective to address all visualization requirements. This, in turn, raises the requirement for integrating different visual presentations so that stakeholders can explore more features of the underlying data. It is worth to note that seamless presentation integration often requires the support from the integration at data and representation levels.

**Presentation Transformation**

To match the cognitive styles and preferences of different stakeholders, visualization systems should enable visual presentations to be transformed from one type to another in a flexible, seamless and efficient manner. This is particularly helpful for stakeholders to highlight different features or perspectives of the underlying data. Similar to presentation integration, presentation transformation also needs the support from related representation transformation.

**Context Adaptation**

The visualization contexts of a presentation involve the problem context within which the presentation is deployed and the situational context of its stakeholders (Bai, White & Sundaram, 2013a). When the contexts are changed, visual presentations should be adaptive or adaptable to the context change. To achieve context adaptation, visualization systems should support modelling different contexts and defining the adaptation rules for presentations to follow under context changes. This is particularly useful to implement role based data security.

The above requirements for presentation customization, integration, transformation and context adaptation are indispensable to achieve high presentation flexibility. Fulfilling these
requirements often require the corresponding support at information representation level, i.e. representation integration and transformation. The ultimate goal of implementing these requirements is to ensure the effectiveness of visualizations for stakeholders under changing contexts.

The above system requirements of information generation, representation and presentation depict the disguising features of PVS. To support implementing these requirements, we introduce PVS frameworks in the subsequent section.

5.2 PVS Frameworks

PVS frameworks are developed by exploring and examining representative information visualization reference models (Card et al., 1999; Chi, 2000; Card & Mackinlay, 1997; Spence, 2007), representative visualization techniques (Heer et al., 2010; Spence, 2007; Chen, 2006; Card et al., 1999; Chi, 2000), and the ideas/concepts involved in visualization-oriented decision support system generators (Bai, White & Sundaram, 2009a). To operationalize the concept of PVS, we propose a conceptual-level PVS framework that illustrates the key building blocks of a purposeful visualization system in sub-section 5.2.1. From the perspective of data transformation, we introduce a framework to unveil how PVS transform data into purpose-driven context-sensitive interactive visualizations in subsection 5.2.2. We then reinforce the concept of PVS by presenting a system-level framework to depict how PVS building blocks cooperate with each other to create and manage purposeful visualizations in sub-section 5.2.3. The design of PVS frameworks also demonstrates a generic approach of modelling and developing visualizations.

5.2.1 Component Oriented Structural Framework

The PVS structural framework is composed of four fundamental building blocks: data, model, solver, and scenario (Figure 5.2). Each building block can be further divided into several component types. In particular, PVS may incorporate two broad types of data, that is, user data required by the system execution, and the data depicting the characteristics of stakeholders and contexts. They also involve four essential groups of models for
accomplishing information generation, representation, and presentation. Accordingly, there are four categories of solvers for manipulating these models. When a model is instigated with data and operated via a property solver, it becomes an instantiated model, namely, a scenario. These PVS components are managed and connected together by a central component – kernel, which enables the communication and cooperation among different components.

**Figure 5.2** Purposeful Visualization System – a component oriented structural framework, adapted from author’s creation (Bai, White & Sundaram, 2010b)

Among these PVS components, information generation related models, solvers and scenarios are responsible for enhancing the quality, relevance and effectiveness of the raw data in terms of addressing the visualization problem of interest. In contrast to these components, the models/solvers/scenarios associated with information representation and presentation define and manage the way of how the ready to be visualized data sets, encapsulated in instantiated IG Models, are transformed into appropriate visual representations and eventually views. The IG/IR/IP related data, models, solvers and scenarios are used to construct a visual scenario. A visual scenario that is effective to the underlying stakeholders, purposes and contexts becomes a purposeful visualization.

Furthermore, according to whether a component is dividable or not, the PVS components can be classified into two types: primitive components and composite components. As demonstrated in Figure 5.3, the former category consists of data components, and IG/IR/IP related models and solvers, while the latter involves IG/IR/IP related scenarios. The IG/IR/IP
scenarios are generated through selecting, validating and mapping data, models and solvers of their corresponding type. Then, by selecting and integrating proper IG, IR and IP scenarios, a visual scenario can be created as a possible solution to address the interested visualization problem and fulfill the intended visualization purposes.

Figure 5.3  Decomposition of a visual scenario, from author’s creation (Bai, White & Sundaram, 2010a; Bai, White & Sundaram, 2010b)

The definitions and functionalities of these PVS components are detailed through the data flow oriented behavioral framework in the following subsection.

5.2.2 Data Flow Oriented Behavioural Framework

The PVS behavioural framework depicts the process of how raw data can be transformed into purposeful visualizations via a set of models and solvers (Figure 5.4). This process comprises three major steps, i.e. information generation, information representation, and information presentation. These steps are consistent with the rational process of the PVDM process.
With PVS, the process of information visualization starts with generating a series of data sets that are relevant and effective to the underlying decision problem from a variety of data sources. It is to ensure that the extracted data sets contain only the information relevant to solving the intended decision problem and are organized into formats suitable for later transformations. The generation of the data sets is accomplished by applying information generation related models and solvers. An IG Model describes the way of how data can be structured and organized to define the problem situation of interest. IG Solvers are algorithms for manipulating IG Models to improve the quality, relevance and effectiveness of the source data.

Information Representation

The extracted data sets are then transformed and encoded into proper visual representations through applying information representation related models and solvers. A visual representation illustrates a primitive visual object deployed to reflect the semantics of the extracted data sets. It is used to define the visual syntax of understanding/designing/implementing visualizations. Typical examples of visual representations are spatial substrate, marks (including points, lines, areas and volumes), connections, enclosure, retinal properties and temporal encoding (Card et al., 1999). Encoding the extracted data into visual representations is achieved through applying IR Models and IR Solvers. An IR Model defines the way of how input data can be mapped to a particular visual representation, while an IR Solver refers to the algorithm for manipulating the IR Model. Visual representations attracting
some stakeholders may not be relevant to other stakeholders. Therefore, when PV and PVS theories are applied to develop field specific visualization systems, relevant IR models should be examined and selected for the system development.

**Information Presentation**

With the support of information presentation related models and solvers, visual representations of data can be transformed into views that visualization stakeholders can display, manipulate, interact with, observe, perceive and interpret. An IP Model defines the layouts and mechanisms of organizing, integrating and presenting visual structures. An IP Solver is the algorithm that can be applied to manipulate the IP Model to facilitate the generation of views. It is through the views that the useful and effective information embedded within the underlying data is eventually conveyed to stakeholders to assist with their decision making.

From the perspective of cognition and interpretation, the contents of a visualization can be decomposed into a set of semantic layers, each of which may present different facets and information of the underlying data (Figure 5.5). This multi-layered view implies that the effectiveness of a visualization can be enhanced by applying proper IG, IR and IP related models and solvers to organize its visual contents into a set of presentation layers that support the semantic layers of the visualization. Moreover, visualization effectiveness in regard to how well the delivered information can be perceived and comprehended by stakeholders may be significantly influenced by both the decisional problem context within which the visualization is deployed and the situational context of the stakeholders. This implies that the selection and application of IG, IR and IP related models and solvers should concern the underlying visualization contexts.

Successfully transforming the raw data into certain views does not mean the achievement of purposeful visualizations. The views must fit the requirements of visualization stakeholders, purposes and contexts to become purposeful visualizations.
The structural and behavioral PVS frameworks highlight the fundamental system building blocks and their key functionalities. To support the implementation of PVS frameworks, we introduce PVS architectures in detail in next section.

5.3 PVS Architectures

Purposeful visualization system architectures define the generic structure and behaviors of PVS and provide guidance for the implementation of a purposeful visualization system. In this section, we present a high level system architecture in subsection 5.3.1, followed by a detailed architecture in subsection 5.3.2.

5.3.1 High Level Architecture

This high level PVS architecture deploys data and IG/IR/IP related models, solvers and scenarios in a loosely coupled fashion and aims to demonstrate the cooperation and communication among these components (Figure 5.6). The bottom tier is a persistent
components storage that consists of a set of resource pools, namely, data, model, solver and scenario pools. All these PVS components are stored in their corresponding resource pools.

![Figure 5.6 Purposeful Visualization System](Bai, White & Sundaram, 2009b; Bai, White & Sundaram, 2010a; Bai, White & Sundaram, 2010b; Bai, White & Sundaram, 2013a)

The lifecycle of each of these models, solvers and scenarios is managed by the kernel component at the middle tier. The kernel offers two essential services: component management, and scenario management. Component management is mainly concerned with the creating, instantiating, executing, modifying and disposing of IG/IR/IP models. In comparison, scenario management supports the whole development lifecycle of IG/IR/IP/
Visual scenarios. With the support from component management, data/model/solver components are prepared and then serve as the input to build IG/IR/IP scenarios. Through leveraging scenario management, the IG/IR/IP scenarios can then be developed, and provide input resources to generate Visual Scenarios that fit the underlying the visualization contexts and address the intended decision problem. The top tier is a flexible user interface, which enables different types of visualization stakeholders to interact with the whole system.

This high level PVS architecture is further depicted through the detailed level system architecture in the following subsection.

5.3.2 Detailed Level Architecture

To guide the design and development of a purposeful visualization system, we propose a three-tier system architecture, as illustrated in Figure 5.7. At the interface layer, a visualization designer communicates his visualization requirements with the system via its user interface. The visualization requirements are then handled by the Visual Scenario Manager at the application layer.

The Visual Scenario Manager looks after the creation, modification, integration, transformation and context adaption of Visual Scenarios. More specifically, it allows stakeholders to check if any existing scenarios match or can be easily modified to support the visualization requirements. If such scenario exists, the Visual Scenario Manager will assist the designer with reusing the visualization solution. If not, it will support the designer to develop new PVS scenarios through selecting, validating, and mapping existing IG, IR, IP and evaluation scenarios. Apart from creating new scenarios, it also supports the transformation, integration and adaptation of visual representations and those of visual presentations in order to achieve the expected system flexibilities. If the existing IG, IR, IP and visual scenarios do not meet the visualization requirements, it will then facilitate the designer to construct new scenarios through the underlying subsystems.

The Visual Scenario Manager is supported by three fundamental subsystems: Information Generation Subsystem, Information Representation Subsystem, and Information Presentation Subsystem. These subsystems support the key data transformation stages
involved the PVS data-level framework, and allow the designer to manage the lifecycles of IG, IR, and IP scenarios respectively. The IG Subsystem allows the designer to set connections to various types of data sources via the Data Source Manager. A data source connection is independent of any IG Models and can be saved in a separate definition file for later reuse. The source data connection is then passed on to the IG Manager as an input to create IG Models. The IG Manager can parse the data source definition, map it to a new IG Model, and validate the mappings to instantiate and execute the model. It may also parse any IG/IR/IP/Scenario definition files to extract their involved data source definition(s) and IG Model definition(s). The created IG Model definition can be assigned to an IR Model as its input.

Figure 5.7 Purposeful Visualization System – a detailed architecture, adapted from author’s creation (Bai, White & Sundaram, 2009b; Bai, White & Sundaram, 2010a; Bai, White & Sundaram, 2010b; Bai, White & Sundaram, 2013a)
The IR Subsystem is responsible for parsing the input IG Model definition and enabling the designer to create/modify an IR Model by configure and validate the IG-IR mappings needed to instantiate the model. It enables the designer to render the instantiated IR Model to see the visual representations. Compared to the IG Subsystem, the IR Subsystem also allows saving the instantiated IR Models to separate definition files for later reuse, and parsing IR/IP/Scenario definitions to derive the valid IR Models involved. IR Model definitions serve as the input to create new IP Models.

The IP Subsystem is able to take a series of IR Scenarios as the inputs and leverage them to create, instantiate, execute, modify, dispose IP Models. This process also requires configuring and validating IR-IP mappings and rendering the model to display visual presentations. The instantiated IP Models (i.e. IP Scenarios) can be saved in separate definition files and then be applied to build visual scenarios.

All the data, model, solver and scenario related components are stored in file folders or proper content database(s), which forms the resource layer for the entire system. The proposed system frameworks and architectures facilitated us to design and develop a fully-fledged prototypical purposeful visualization system. We demonstrate the application of the prototype through a sequence of scenario-driven visualizations in chapter 6, and then explicate the implementation of the prototype in detail in chapter 7.
CHAPTER 6  PURPOSEFUL VISUALIZATION ARTEFACTS

APPLICATION

To validate the proposed frameworks and architectures of purposeful visualization systems, we implemented a prototypical purposeful visualization system that supports all the distinguishing features of PVS discussed in section 5.1. We leave the implementation details of the prototype to chapter 7. In this chapter, we focus on demonstrating the application of the prototype to construct purposeful visualizations and support visual decision making. In particular, we apply the prototype to support analyzing the network reliability for electricity distribution businesses and managing capital investment for improving network performance.

This case caught up our attention because of the complexities involved. In particular, it incorporates a series of decision problems that feature complex and interwoven (lateral and sequential) decision making; it is in the nexus of multiple visualization application domains such as statistical data analysis, history management, and network data management. Moreover, it needs to accommodate a variety of visualization stakeholders, purposes and contexts and their corresponding diverse and changing visualization requirements; it requires dealing with large volumes of complex network load data with spatial, temporal and multi-dimensional features. Most importantly, the complexities involved in this case naturally raise requirements for the design and implementation of flexible purposeful effective visualizations.

With the goal of demonstrating the support of the prototype, we do not attempt to tackle all the complexities involved in this case. Instead, we focus on several representative decision problems that are derived from this case and range from simple to complex. These decision problems allow us to form a series of scenarios in order to illustrate how the prototype supports developing purposeful visualizations under diverse and varying visualization stakeholders, purposes and contexts. Moreover, we also demonstrate how the purposeful visual decision making process can be applied to address these decision problems and how the prototype supports this process.

This chapter is organized as follows. We provide a brief description of the case in section 6.1, and then outline the decision problems derived from the case and analyze the involved
complexities in section 6.2. To address these problems, we proceed to demonstrate the application of the prototype to develop purposeful visualizations for diverse stakeholders, purposes and contexts in section 6.3. We also demonstrate how the prototype helps to sustain visualization effectiveness under changing visualization contexts in section 6.4. Finally, in section 6.5 we show how the purposeful visual decision making process, with the support from the prototype, can be deployed to solve the decision problems.

6.1 Case Background

Electricity is vital to modern economy in the sense that it enables many other fundamental technologies in various domains like telecommunications, transportations, pharmaceuticals, hospital healthcare, educations, etc. An electricity network delivers electrical power from suppliers to residential, commercial and industrial consumers. If a fault is incurred to any part of the network such as substations, transformers, circuits, switches, poles, and so forth, the power supply can be disrupted. When a power outage happens, the network operating company will arrange crews to investigate the issue and recover the power supply. Understanding the impacts of individual power outages contributes to the understanding of the average network reliability performance.

A common requirement for many electricity distribution businesses is to monitor and report both the frequency and duration of network interruptions through the internationally recognized measures of SAIDI (system average interruption duration index) and SAIFI (system average interruption frequency index). One of their strategic goals is to deliver network reliability that meets the SAIDI and SAIFI targets set by regulatory requirements and to meet customer expectations that are reflected in customer service levels. This again contributes to directing maintenance efforts and identifying trends and anomalies in performance.

Monitoring the average network reliability performance enables to analyze the trends/patterns of problematic network areas that cause frequent network interruptions and high maintenance costs. To choose appropriate technologies to invest in to improve such network areas, a key requirement is to analyze and discover patterns and trends of historical
network load variations and estimate the future demand growth in short, medium and long terms.

Though this case involves a series of decision problems, we select only a few representative problems for demonstrating the support of the prototype. The selected decision problems are specified in the following section.

### 6.2 Decision Problems

For the demonstration of the purposeful visualization system prototype, we select a set of decision problems ranging from simple to complex, which are around four essential areas: outage impact identification, network reliability analysis, problem area discovery, and network improvement strategy. These decision problems are as follows:

- What is the impact of a power outage?
- How reliable is an entire electricity network?
- What are the problematic areas on the network?
- How to choose the optimal solution to upgrade the problematic network areas?

These decision problems are interrelated and interwoven with each other (Figure 6.1). Specifically, choosing the appropriate network improvement strategy requires discovering the frequent problem areas on the electricity distribution network, which, in turn, needs both a clear view of the overall network reliability performance and the ability to identify and investigate the impacted areas of individual network outages. Meanwhile, analyzing the overall network performance often relies on the analysis of individual network outages to achieve a deeper understanding of the root causes of the issues with the overall performance.

Furthermore, each decision problem could involve a number of visualization purposes that require the support from the prototype. We restrict the demonstration to a list of representative task-based visualization purposes (Table 6.1).
Figure 6.1 Relationships among the selected decision problems

<table>
<thead>
<tr>
<th>Decision Problem</th>
<th>Representative Visualization Purposes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outage impact identification</td>
<td>▪ To present a power outage that occurred in an electricity distribution network</td>
</tr>
<tr>
<td>Network reliability analysis</td>
<td>▪ To monitor and analyze electricity network faults &amp; reliability</td>
</tr>
<tr>
<td>Problem area discovery</td>
<td>▪ To discover trends/patterns of problematic network areas</td>
</tr>
<tr>
<td>Network improvement strategy</td>
<td>▪ To forecast future demands&lt;br&gt;▪ To interpret how future network demands are affected by the growth in population, air conditioning demand and employment</td>
</tr>
</tbody>
</table>

Table 6.1 PVS demonstration scope

Based on the above decision problems, we create three cases for presenting the support of the prototype, each of which has a unique purpose of demonstration. The first case concentrates on demonstrating how the prototype supports diverse visualization stakeholders, purposes and contexts (in section 6.3). The second case aims to demonstrate how the prototype assist with sustaining visualization effectiveness under visualization
context changes (in section 6.4). The third case focuses on the decision problem of network improvement strategy and demonstrates the application of the PVDM process (in section 6.5).

6.3 Support for Diverse Visualization Stakeholders, Purposes and Contexts

Based on decision problem of network reliability analysis, we select two representative scenarios with different visualization problems, stakeholders, purposes and visualization requirements, i.e. operational-level network fault monitoring, and managerial-level KPI monitoring. Electricity distribution businesses may be required to disclose their performance against its SAIDI and SAIFI targets in an annual quality compliance statement, which could be subjected to external audit, in order to establish a high level of assurance in the reported information. In some countries like Australia and New Zealand, failing to meet the annual agreed performance targets can incur significant penalties. Therefore, it is rather important to deliver the right information to the right stakeholders at the right time and place. The scenarios used in this section reflect the real visualization contexts and requirements at a New Zealand lines company. Due to the concerns of the sensitivity and confidentiality of their network fault data, we use make-up data to generate the visualizations presented in this section. The scenarios are explicated in subsections 6.3.1-6.3.2.

6.3.1 Scenario 1: Operational-Level Network Fault Monitoring

Electricity service/network operations engineers often need to monitor daily network faults, investigate their impact on the network and customers and direct maintenance efforts in a timely manner. With the prototype, stakeholders can use a semantic zoom model to present both the overview and detailed view of daily network faults. In the example demonstrated in Figures 6.2 and 6.3, the top layer applies small multiples to indicate the number of customers affected by different types of network faults (Figure 6.2). The size of each bubble represents the affected customers count. In the second layer (Figure 6.3), Bing map is utilized to present the distribution of the affected feeder areas with map polygon colors indicating the amount of customer minutes lost. The darker the color is, the larger the amount of customer minutes is lost.
6.3.2 Scenario 2: Managerial-Level KPI Monitoring

Compared to the service/network operations engineers, the senior management is interested in not only the network reliability performance in a short period but also patterns, trends and anomalies in performance including exceptions to service level agreements. Such information
is helpful to their decision making in network planning, demand side management, system operations, maintenance and financial planning. With the prototype, stakeholders can create visualization solutions to show high-level summaries of network reliability performance, drill down to details and compare/analyze performance in various network areas. For instance, in the visualization solution below, stakeholders can display the distribution of monthly network faults and SAIDI, drill down to view the relevant daily SAIDI and customer minutes lost that contribute to the monthly KPI, and even compare multiple KPIs of different periods in the same timeline viewer (Figure 6.4).

![Figure 6.4 Network reliability KPIs of different periods](image)

Both operational-level and managerial-level stakeholders may need to investigate specific network outages in order to identify root causes and obtain a deeper understanding of the overall network reliability performance. We hence create the third scenario based on the decision problem of outage impact identification. Apart from the stakeholders within electricity distribution businesses, this problem also interests external customers who are affected by the power outage.
6.3.3 Scenario 3: Power outage impact investigation

This scenario is created based on makeup data of a massive power outage that occurred in Auckland. It caused over 80,000 homes to have no power at the peak of the outage and some suburbs to lose power for up to 48 hours. An overview of the areas affected by this power outage is presented in Figure 6.5, while the key snapshots reflecting how the outage is resolved is shown in Figure 6.6.

Figure 6.5 Overall network outage distribution

Figure 6.6 Key network outage snapshots
As illustrated in the above contexts, the IP Scenario that is relevant and useful for service/network operators becomes insufficient for the senior management staff. To maintain visualization effectiveness and leverage the effort already put in developing the visualization solution for scenario 1, stakeholders can continue to use the IR Scenario (created for scenario 1) with replacing the underlying visualization and context data. The impacts from the changing visualization contexts are eventually reflected through visualization requirement changes. To respond to them, stakeholders can adapt the visualization solution by adjusting the data, models and solvers settings of the involved visualization and context components. This example highlights the importance of sustaining visualization effectiveness. We further expand on this requirement and demonstrate the support of the prototype for it in the following section.

### 6.4 Support for Sustaining Visualization Effectiveness

Based on the decision problems of analyzing network reliability and identifying problematic network areas, we create four scenarios to demonstrate the support of the prototype for adapting to various problem/stakeholder/purpose/time context changes and the associated visualization requirement changes. These scenarios are discussed in subsections 6.4.1-6.4.4

#### 6.4.1 Scenario 1: Problem Context Adaptability

The purposeful visualization system prototype supports adapting to various problem contexts by creating and manipulating appropriate visualization solutions. For example, a typical problem context in network reliability management is about supporting service/network operations engineers to monitor daily network faults and investigate their impact on the network to assist in isolating/removing the faults. Another one is about helping customer relations managers to run customer surveys from time to time in order to understand customer experience/expectations about the network performance and services. To support network fault management, the prototype allows designing and developing visualization solutions showing daily network faults distribution (Figure. 6.7). When the problem context changes to customer satisfaction management, it allows creating visualizations displaying customer survey results (Figure. 6.8).
Figure 6.7  Daily customer minutes lost by feeder area

Figure 6.8  Sample customer survey results
6.4.2 Scenario 2: Purpose Context Adaptability

In electricity distribution businesses, monitoring and investigating network faults may aid in fulfilling various purposes. It can benefit service/network operations engineers to facilitate minimizing risks to the health and safety of people and network equipment and to direct maintenance efforts in a timely manner. It may also assist the senior management to analyze and identify patterns, trends and anomalies in performance including exceptions to service level agreements. With the prototype, visualization solutions can be created to support the purpose of service/network operations engineers. For example, it could be helpful to have a two-layer semantic zoom model presenting both the overview and detailed view of daily network faults. The top layer applies small multiples to represent the number of customers affected by different types of network faults on a selected date (Figure 6.9). The size of each bubble represents the affected customers count. In the second layer (Figure 6.7), Bing map is utilized to present the distribution of feeder areas affected by the faults and map polygon colors indicate the amount of customer minutes lost in each affected feeder area. The darker the color is, the larger the amount of customer minutes is lost.

![Figure 6.9 Daily affected customers by fault type](image)

Figure 6.9 Daily affected customers by fault type
When it comes to support the purpose context of discovering patterns of problematic network areas, the prototype can be used to create a deep zoom model allowing stakeholders to compare SAIDI distributions cross years (Figure 6.10).

![Yearly SAIDI distributions from 2007 to 2011](image)

Figure 6.10 Yearly SAIDI distributions from 2007 to 2011

### 6.4.3 Scenario 3: Stakeholder Context Adaptability

The purposeful visualization system prototype supports stakeholders with different cognitive and skill acquisition characteristics, prior knowledge and visual preferences. For example, it allows beginner-level stakeholders to work with less number of network reliability KPIs during shorter time-periods (normally involving less volumes of performance data) while for proficient stakeholders it supports to plot and compare multiple network reliability KPIs in the same timeline model (Figure 6.11) and zoom in to view relevant details. It also allows stakeholders and/or context-aware systems to leverage the efforts already put into developing visualization solutions. Existing visualization solutions can be reused by replacing or adjusting the underlying data, models and solvers settings of the involved visualization and context components to support new visualization requirements.
6.4.4 Scenario 4: Time Context Adaptability

With the purposeful visualization system prototype, stakeholders can show network reliability performance in different time periods and drill up/down to display the performance data in long-term/short-term periods. For instance, in the visualization solution below, stakeholders can display the distribution of network faults with map polygon color indicating the monthly SAIDI (Figure 6.12). The darker the color is, the more significant the monthly
SAIDI is to be. If needed, stakeholders can drill down to view the relevant daily SAIDI and customer minutes lost that contribute to the monthly KPI.

![Image](image_url)

**Figure 6.12** Monthly SAIDI by feeder area

It deserves to be noted that the visualization solutions demonstrated in the scenarios in sections 6.3 and 6.4 were created by applying the proposed purposeful visual decision making process. We focus on demonstrating the application of this process to support decision making in the subsequent section.

### 6.5 Support for Visual Decision Making through Flexible Presentations

To demonstrate the application of the proposed purposeful visual decision making (PVDM) process (Figure 4.6), we apply it to fulfill the selected visualization requirements for the problem of developing network improvement strategy. The support for visual decision
making is articulated and illustrated from the perspective of providing flexible visual presentations. The way how flexible visual presentations are achieved in the prototypical purposeful visualization system is through decoupling information representation and presentation. Hence, apart from demonstrating the PVDM process, such feature will also be reflected in the demonstration in this section. Specifically, subsection 6.5.1 focuses on understanding the decision problem and visualization requirements, which is the intelligence phase of the PVDM process. Subsection 6.5.2 demonstrates how to support decision making by developing flexible visual presentations in the prototype, which corresponds to the design phase of the PVDM process. Subsection 6.5.3 discusses the choice and implementation phases of the PVDM process.

6.5.1 VDM – Intelligence

As discussed in section 4.3, the intelligence phase of the PVDM process involves two important stages: problem identification and decision environment understanding. They aim to assist decision makers with understanding a decision problem, the involved visualization purposes and how the effectiveness of purpose fulfillment will be assessed. With regard to developing electricity network improvement strategy, we choose to focus on the involved decision problem of understanding the impact of different sets of influential factors on network demand forecasts for the purpose of demonstration.

Electricity demand forecasting is concerned with analyzing and discovering patterns and trends of historical load variations and estimating future load growth in short, medium and long terms. It is particularly useful for electric utility businesses to support their decision making in energy trading and generation, system operations and maintenance, transmission/distribution planning, demand side management and financial planning. Accurate load forecasts effectively help them to minimize engineering and financial risk and optimize operational efficiency and reliability. Due to the critical role demand forecasting plays, many electric utility operators develop in-house solutions to generate load forecasts.

Demand forecasting involves multiple users from different business units with different backgrounds, requirements and purposes. Producing forecasts for them requires selecting
and applying appropriate forecasting models/processes to deal with (1) large volumes of complex historical load data with spatial, temporal and multi-dimensional features and (2) forecast adjustment factors like weather, economic and demographic trends, customer profiles, and end technology trends at the forecasted area. Dealing with such complexities necessitates purposeful visualizations for supporting decision makers with different interests, purposes and visualization requirements.

To obtain a good understanding of the decision problem and visualization requirements, we have conducted a series of interviews with relevant stakeholders from network control, maintenance and planning teams at a local utility company in past three years. We have also performed regular field observations to study how the users prefer to work with their demand forecasts. As an important output of this process, we have identified a variety of functional and cognitive requirements for their preferred visualizations.

It deserves to be pointed out that the relevance and importance of the visualization requirements are not same for all decision makers because they may apply the forecasts for different work purposes. Therefore, we collected both role-level and individual-level evaluation criteria from the users in order to assess the overall visualization effectiveness and flexibility. Understanding the user requirements has went through many iterations of the proposed process due to the learning curve of users. Our experience with collecting and understanding the requirements for developing purposeful visualization have reinforced the necessity of following an iterative PVDM process to ensure user satisfaction on presentations.

Based on the collected visualization requirements, we designed and developed a series of reusable visual scenarios by using the prototypical purposeful visualization system, which are illustrated in the section below.

6.5.2 VDM – Design

The design phase of the PVDM process focuses on designing and developing alternative purposeful visualization solutions for the interested decision problem and requirements. Given that many visualization systems provide weak support for flexible visual presentations, our demonstration focuses on showing the flexible support of the prototypical purposeful
visualization system for information presentation. The demonstrations of the support information generation and representation will be provided in chapter 7.

Though the impact analysis of influential forecast factors requires a series of presentation designs, this section, for the purpose of the demonstration, highlights one of its presentation requirements (i.e. comparing the results from multiple forecast trials) and demonstrates the support of the prototype for flexible information presentation. In particular, this subsection focuses on demonstrating the support for developing flexible alternative visual presentations, while the details of the alternative visual presentations are provided in the following subsection. To assist users with developing flexible visual presentations, this prototype implementation supports presentation customization, integration, transformation and context adaptation.

More specifically, the prototype allows creating new presentations from scratch or existing presentations. With this prototype, a presentation model instantiated with appropriated data and solver can be saved to a reusable xml-based definition file. Such definition file stores model configurations and can serve as a directly input to model customization and context adaptation. For instance, Figure 6.13 shows the settings window of an instantiated sequential scenes model, at which users may customize and enhance the representations and adapt it to different visualization contexts in accordance with their requirements. Additionally, the prototype allows displaying the same set of visual representations in various presentation fashions and transform the presentation from one type to another by switching presentation models and adjusting the model configurations. Furthermore, the prototype enables to integrate presentations to support more sophisticated user requirements. For example, Figure 6.14 illustrates how a sequential scenes model integrates another presentation model of the same kind and a semantic layering model.

To support the requirement of comparing the results from multiple forecast trials, we created a set of alternative visual presentations by using the prototype. These alternatives and how to choose each alternative are detailed in the next subsection.
Figure 6.13  Customization of a sequential scenes model

Figure 6.14  Examples of presentation model integration
6.5.3 VDM – Choice and Implementation

The choice and implementation phases of the PVDM process aim to evaluate the effectiveness of alternative visual solutions, select the optimal solution, and address the interested decision problem by implementing the selected solution. To compare the results from multiple forecast trials, we created four presentation models by instantiating from a series of generic information presentation models, i.e. landscape, nested spaces, semantic layering, and sequential scenes. The effectiveness of each visual presentation model was evaluated against the visualization purposes and preferences of decision makers. The demonstration of the presentation models is conducted by applying the prototype to display and compare different versions of demand forecasts. To facilitate the demonstration, we introduce three forecast trials with each adopting different sets of forecast factors.

For decision makers who prefer to compare the overall differences among the trials, the landscape model may be considered to form a view including forecasts of all three trials. An example of this is demonstrated in Figure 6.15, which presents all trial forecasts for 2016. Each colored polygon of a map indicates a feeder area. The darker the color, the higher forecast the feeder is assigned.

![Image](6.15.png)

Figure 6.15 An example of the landscape model comparing three forecast trials

For decision makers who need to focus on a single forecast trial at a time, the semantic layering model can be particularly helpful (Figure 6.16). If the decision makers want to display
the forecasts of each trial by year/season/month, they may consider the sequential scenes model (Figure 6.17).

Figure 6.16 An example of the semantic layering model highlighting each forecast trial at a time

Figure 6.17 An example of the sequential scenes model
In addition, decision makers may seek to create a story pertinent to a certain subject matter and then suppress its involved complexities when it is embedded within a bigger story. Under such circumstances, using the nested spaces model would be convenient. Figure 6.18 provides an example of nested spaces and the visual contents of the embedded story can be popped up when the decision makers want to view them (Figure 6.19). All the implemented presentation models allow the decision makers to navigate through the visual contents by zooming, panning, scrolling and rotating.

Figure 6.18 An example of the nested spaces model

Figure 6.19 An example of a nested space
In this chapter, we introduced the case of analyzing the network reliability for electricity distribution businesses and managing capital investment for improving network performance. This case was used to provide an application context for the demonstrations of the purposeful visualization system prototype. Based on this case, we demonstrated the support of the prototype for (1) developing effective visualization solutions for diverse visualization stakeholders, purposes and contexts, (2) maintaining visualization effectiveness under various context changes, and (3) applying the purposeful visual decision making process. To highlight the support for these aspects, we, in the demonstrations, intentionally hided the actual tasks and processes of developing the information generation/representation/presentation models and scenarios. Details of how the prototype supports the entire information visualization pipeline and the PVS distinguishing features are provided in the following chapter.
CHAPTER 7  PURPOSEFUL VISUALIZATION SYSTEM IMPLEMENTATION

Purposeful Visualization Systems aim to facilitate stakeholders to visually display and communicate their understandings of data and potential ways to apply the data to support decision making. It does so by allowing them to create visualizations against the underlying source data, utilize the visualizations to form visual scenarios and eventually solve the problems by presenting the scenarios in a flexible fashion. An important goal of PVS is to bridge the gap between owning the data and efficiently leveraging the data. To validate the concepts, frameworks and architectures of PVS, we implemented a prototypical purposeful visualization system by following exactly the PVS detailed level architecture presented in subsection 5.3.2. The demonstrations of the prototype features aim to prove the flexibilities that the prototype offers for designing and developing purposeful visualizations.

In this chapter, we present an overview of the system structure and functionalities of the prototype in section 7.1, followed by a brief discussion of how the PVS was developed in section 7.2. We then demonstrate the key system functionalities required throughout the purposeful visualization development process in sections 7.3 – 7.5.

7.1 Overview of the Purposeful Visualization System Prototype

The purposeful visualization system prototype has a particular focus on supporting the design phase of PV decision making process. It allows users to generate purpose relevant information, visually represent the information, integrate multiple visual representations, transform visual representation from one type to others, flexibly present the visual representations, integrate multiple visual presentations, create purposeful visual scenarios, and adapt the existing visualization components for multiple purposes.

The prototype is implemented based on the proposed PVS detailed-level architecture. It employs a Visual Scenario Manager and three subsystems with each supporting a fundamental stage of the PVS data flow oriented behavioral framework (section 5.3.2). The IG Subsystem manages data source connection and IG Model development, the IR Subsystem
creates and manages IR Models, the IP Subsystem looks after the lifecycle of IP Models, and
the Visual Scenario Manager enables stakeholders to interact with manipulate visual
scenarios for decision making. Moreover, the prototype leverages network/local file
directories to store data source connections and PVS scenarios in order to form the resource
layer of the PVS architecture. The implementation details of these essential building blocks
are presented in appendices A-F, as illustrated in Figure 7.1.

Figure 7.1  Implementation of the prototypical purposeful visualization system

The prototype is implemented by leveraging a set of Microsoft technologies, i.e. Windows
Presentation Foundation (WPF), ADO.NET entity framework, Language Integrated Query
(LINQ), SQL Server, Bing map and Visual C#. However, a similar system could be developed by
utilizing equivalent technologies from other vendors such as Oracle and IBM. WPF is a
presentation technology introduced by Microsoft specifically for designing and developing
visually stunning user interfaces and experience (Microsoft, 2015a). It is particularly powerful
for separating application logic from user interface. Furthermore, apart from the major
implementation technologies, the prototype also employs several third-party controls to support the development of presentation models. For example, an open-source WPF control, namely timeline viewer (Silverlight & WPF Timeline Control, 2015), has been integrated into the prototype to implement the sequential scenes.

A summary of the key functionalities of the prototype subsystems is outlined in Table 7.1. Each subsystem has a data flow and a visual flow. The data/visual inputs and outputs of each subsystem are presented in Table 7.2. The development process of the prototype is detailed in the following section.

<table>
<thead>
<tr>
<th>Key System Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>IG</td>
</tr>
<tr>
<td>- Create/Edit/Delete data source</td>
</tr>
<tr>
<td>- Extract valid data source(s) from any types of PVS component definitions</td>
</tr>
<tr>
<td>- Save data source definitions for reuse</td>
</tr>
<tr>
<td>- Create/Edit/Delete IG Model</td>
</tr>
<tr>
<td>- Save IG Model definitions for reuse</td>
</tr>
<tr>
<td>IR</td>
</tr>
<tr>
<td>- Create/Edit/Delete IR Model</td>
</tr>
<tr>
<td>- Extract valid IG Model input from any types of PVS component definitions except Data Source connections</td>
</tr>
<tr>
<td>- Support model integration</td>
</tr>
<tr>
<td>- Support navigable spots in an IR Model</td>
</tr>
<tr>
<td>- Assign multiple linked IR Models to the same navigable spot</td>
</tr>
<tr>
<td>- Save IR Model definitions for reuse</td>
</tr>
<tr>
<td>IP</td>
</tr>
<tr>
<td>- Create/Edit/Delete IP Model</td>
</tr>
<tr>
<td>- Create/Edit/Delete Visual Scenario</td>
</tr>
<tr>
<td>- Use multiple IR Models and/or IP Models as the input to IP Model creation</td>
</tr>
<tr>
<td>- Use multiple IG Models, IR Models and/or IP Models as the input to Visual Scenario creation</td>
</tr>
<tr>
<td>- Extract valid IR or IP Models from existing component definitions (except Data Source and IG Model)</td>
</tr>
<tr>
<td>- Support model integration</td>
</tr>
<tr>
<td>- Save IP Model definitions for reuse</td>
</tr>
<tr>
<td>- Save Visual Scenario definitions for reuse</td>
</tr>
</tbody>
</table>

Table 7.1 Key system functionalities of the purposeful visualization system prototype
Table 7.2 Data and visual flows in the purposeful visualization system prototype

<table>
<thead>
<tr>
<th>IG</th>
<th>Data Input</th>
<th>Data Output</th>
<th>Visual Input</th>
<th>Visual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data source definition that may come from any instantiated Data Sources, IG Models, IR Models or IP Models</td>
<td>IG Model definition</td>
<td>Data source visual</td>
<td>IG Model visual</td>
</tr>
<tr>
<td>IR</td>
<td>IG Model definition that may come from any instantiated IG Models, IR Models or IP Models</td>
<td>IR Model definition</td>
<td>IG Model visual</td>
<td>IR Model visual</td>
</tr>
<tr>
<td>IP</td>
<td>IR Model definitions and/or IP Model definitions</td>
<td>IP Model definition; Visual Scenario definition</td>
<td>IR Model visual and/or IP Model visual</td>
<td>IP Model visual; Visual Scenario visual</td>
</tr>
</tbody>
</table>

7.2 Prototype Development

As we discussed in subsection 3.4.3, we adopted an evolutionary prototyping approach to guide the development of purposeful visualization systems. This approach is particularly helpful for identifying and understanding unknown or neglected system requirements. Following this approach, we started with implementing well-understood requirements for information generation and representation. For example, we developed the support for the prototype connecting to multiple data sources if required by a user and enabling the user to create IG Models from scratch or by reusing existing visualization components. We also implemented a set of template IR Models and Solvers for the user to instantiate them to achieve his own purpose. These requirements to us were known system requirements and we understood how the system should behave when it responds to such user requirements. Compared to the implementation of well-understood system requirements, we went through
multiple iterations to figure out how IP Models should be designed to enhance flexibility and usability. At the end of each implementation iteration, we got a working prototype to demonstrate to end users. The experience we gained from each implementation iteration and the feedback we collected through communicating with domain experts and users continue to help to clarify the system requirements and refine our proposed theory.

To meet the requirements of PVS development and our research process, we have gone through three major implementation iterations, as illustrated in Figure 7.2. Each iteration comprises a series of minor implementation iterations to ensure the prototype to meet the predefined system requirements.

Figure 7.2 The Evolution of the purposeful visualization system prototype

In Iteration I, we implemented the support for users to connect to various data sources and to create and manage IG Models. We also developed some commonly used statistical IR Models and a zoomable canvas as the default IP Model. In other words, this prototype could only provide limited support for the representation and presentation stages of Information Visualization Pipeline by the end of Iteration I. Such support was hence further strengthened during Iteration II and III.

The focus of Iteration II was to create a sound Information Representation Subsystem with better support for transforming an IG Model output into meaningful visual forms. Apart from the IR support already implemented in Iteration I, this IR Subsystem offers users more standard IR Models and Solvers for representing static/animated flow line, pushpin collection,
tag cloud, survey and survey result. It also allows users to flexibly instantiate and integrate these models to create more interesting and purposeful visual contents.

Despite providing a few more standard IP Models and Solvers, the prototype was still weak when it came to developing visual scenarios. This weakness was addressed through Iteration III during which we implemented the IP Subsystem to handle the system requirements for integrating and presenting visual stories. In addition, to better fulfill the system requirements, we implemented new IR Models and Solvers such as static and animated polygon collection, and made improvements to the existing standard IR and IP Models, for example, enabling new features such as static and dynamic context propagation among IR Models.

This prototype supports the distinguishing system requirements and features of PVS discussed in section 5.1. We demonstrate the key system functionalities required for developing purposeful visualizations in the subsequent sections 7.3 – 7.5.

7.3 Information Generation

Information generation aims to extract data relevant and effective to address a decision problem, as discussed in subsection 5.1.1. This is realized through selecting or creating appropriate data sources and IG Models in the prototype. Accordingly, we categorize the key system functionalities for supporting information generation into two parts: data source management, and IG Model Management. The following two subsections will specify these functionalities respectively.

7.3.1 Data Source Management

The data source component defines an abstraction (i.e. a logical model) of an underlying data source, which becomes an indispensable input to IG Model development. It serves as the initial data extraction step for users to specify the scope of source data relevant to a decision problem. Such function is commonly demanded and supported by many existing visualization and reporting systems. For example, Microsoft SQL Server Data Tools use data source and data source view to connect to an underlying relational database and enable users to build reports and/or multidimensional analysis services cubes against the source data. In PVS, we
deploy a third party visual query builder to support data source connection. This query builder, namely EasyQuery, is developed by KORZH.com and supports the integration with WPF applications (EasyQuery, 2015).

To add a new data source, a user can drag the Data Source component from the Information Generation toolbox and drop it on to the scenario design canvas (Figure 7.3). Then, the user can double click on the added data source icon and start the configuration. If the required data source is already defined before, the user can simply reuse the data source by loading its definition file and make any changes if needed (Figure 7.3). In other words, the same data source can be used in multiple visual scenarios and provide input to multiple IG Models at the same time. The same visual scenario can use as many data sources as required by the user to fulfill a certain purpose.

After a new data source is created, the user can instantiated it by configuring its connection settings and content properties. The configuration of a new data source is started with selecting the connection type and building the connection string (Figure 7.4). The connection settings of the data source can be configured by clicking the build connection button. In the Connection Properties window, the user can select the relevant data source type to connect, specify the connection properties and the system will generate the connection string automatically (Figure 7.5). Please note that the options of data source types supported by the prototype can be extended by implementing appropriate interfaces.
An example of connecting a Microsoft SQL Server database is provided as follows. Once the data source is connected (Figure 7.5), the system will automatically pop up the add table(s) window to allow the user to select data tables relevant to his visual scenario development (Figure 7.6). The system can also automatically detect possible linkages among the selected tables if required.
Each selected table is referred as an entity while its table columns are defined as attributes (Figure 7.7). For each entity attribute, the user can define the preferred operations that can be applied and choose the preferred way to edit its values if needed.

To reconfigure an existing data source component, the user should first load it into Visual Scenario Manager if it has not been made available for modifying settings. Then, double
clicking on the component icon, the user can open up its settings window and make any changes required (Figure 7.8).

![Figure 7.8 Data source modification](image)

When a data source component is properly configured, the system allows the user to save the component settings to a separate definition file. The user may right click on the component to open up the context menu and choose “Save Definition” from the menu list (Figure 7.9). The definition file being saved is XML-based and is designed for supporting component reusability. An example of such definition file is provided in Figure 7.10.

Deleting a data source from an existing visual scenario is fairly straightforward. Selecting the data source component to be removed, the user can click the Delete button from the Edit pane on the top and then the data source will be deleted (Figure 7.9). Deleting a data source means that it is removed from the hosting scenario but will not affect it to be used by others. If its definition has already been saved by the user, the data source can always be added back and/or used for other scenario development.

The above system support for data source management allows users to flexibly create, modify and reuse data sources, which, in turn, provide the input to create IG Models. The system support for managing IG Models is depicted in next subsection.
7.3.2 IG Model Management

An IG Model defines an abstraction of the useful relevant data for solving a decisional problem. In the prototype, an IG Model is implemented as a data query that fetches a data set from the underlying source data in order to reflect particular aspects or features of the data. It is always created against a certain data source that can be an existing data source component or any other PV components containing a data source component. When creating the IG Model from scratch, the user can drag an IG Model component from the Information Generation toolbox and drop it on to the scenario design canvas. By selecting the interested data source, the user can assign it to the IG Model as an input (Figure 7.11). It deserves to be pointed out that the input does not necessarily be a data source component. Any instantiated PVS component containing a data source can become a valid input to build a new IG Model.
Double clicking on the model icon, the user can open up IG Manager to configure the settings of the model. As demonstrated in Figure 7.12, the Entities and Attributes pane offers an overview of the selected data source component and allows the user to select relevant elements to build a query that extracts only the useful data to be visualized. Defining appropriate result columns, sorting and conditions, the user can execute the query to browse the actual result data (Figure 7.12). The IG Model can be saved in a separate definition file for later reuse (Figure 7.13). An example of an IG Model definition is provided in Figure 7.14.
Figure 7.13  Save an IG Model

Figure 7.14  An example of IG Model definition
If the new IG Model is to be created by reusing an existing IG Model, it does not need to be assigned with any particular data source component. The existing model definition can be loaded directly into the new model by clicking the Load button in IG Manager. Then the user can make any changes required and save the new IG Model for reuse. In the prototype, the same data source can be assigned to multiple IG Models simultaneously (Figure 7.15). When an IG Model is created, the user can use it as an input to create and instantiate IR Models. The functionalities to ensure flexible IR Model development are detailed in the following section.

![Image of IG Models](image)

Figure 7.15  Instantiating multiple IG Models from the same data source

### 7.4 Information Representation

IR Model defines a way to represent the extracted source data in an appropriate visual form. To support the development of IR Models, we implemented a series of template IR Models in
the prototype. In this section, we briefly introduce these template IR models in subsection 7.4.1, and then discuss how IR Model creation, instantiation, execution, deletion and reuse are supported by the prototype in subsection 7.4.2. The support for visual representation transformation and presentation are demonstrated in subsection 7.4.3 and 7.4.4 respectively.

### 7.4.1 Template IR Models

For the purposeful visualization system prototype, we selected and implemented 14 template IR Models, as summarized in Table 7.3. New IR Models can be created based on these template IR Models or by reusing existing instantiated IR Models.

<table>
<thead>
<tr>
<th>IR Model</th>
<th>Model Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>IR_QuestionList</td>
<td>To represent a questionnaire and its contained subject categories and questions that require structured or non-structured feedback</td>
</tr>
<tr>
<td>IR_DimensionChart</td>
<td>To show the feedback collected via a questionnaire</td>
</tr>
<tr>
<td>IR_StatisticalChart</td>
<td>To represent data in line/column/area/bubble charts</td>
</tr>
<tr>
<td>IR_FlowLines</td>
<td>To show data through a set of line segments in a static fashion</td>
</tr>
<tr>
<td>IR_AnimatedFlowLines</td>
<td>To show data through a set of line segments in an animated fashion based on when they become relevant</td>
</tr>
<tr>
<td>IR_PushpinSet</td>
<td>To indicate a set of locations on a map</td>
</tr>
<tr>
<td>IR_PolygonSet</td>
<td>To indicate a set of polygon areas on a map</td>
</tr>
<tr>
<td>IR_SmallMultiples</td>
<td>To show trend/pattern through multiple bubbles or text fields</td>
</tr>
<tr>
<td>IR_AnimatedPushpinSet</td>
<td>To display pushpins on a map in an animated fashion based on when they become relevant</td>
</tr>
<tr>
<td>IR_AnimatedPolygonSet</td>
<td>To display polygons on a map in an animated fashion based on when they become relevant</td>
</tr>
<tr>
<td>IR_TagCloud</td>
<td>To represent data via a tag cloud</td>
</tr>
<tr>
<td>IR_Image</td>
<td>To show an image</td>
</tr>
<tr>
<td>IR_Annotation</td>
<td>To add a descriptive text</td>
</tr>
<tr>
<td>IR_Media</td>
<td>To display audio/video files</td>
</tr>
</tbody>
</table>

Table 7.3 Template IR Models implemented in the purposeful visualization system prototype
The selection and development of these model types are based upon the visualization requirements of application domains where the capabilities of the prototype are demonstrated through a sequence of scenario-driven illustrations. However, the prototype is not limited to these IR Models, but can be extended to support more application domains by implementing more domain-related IR Models. The system support for applying these template IR Models to develop IR Scenarios are presented in the following subsection.

7.4.2 IR Model Management

A new IR Model can be created either from scratch based on the model templates or by reusing existing IR Models. To create an IR Model from scratch, a user should first select a template IR Model from the Information Representation toolbox and drop it on to the scenario design canvas. To instantiate the IR Model, the user chooses an instantiated IG Model and assigns it to the template IR Model by mapping the IG Model’s data output to the IR Model attributes. For instance, the data generated for a power outage may be represented via an IR Model that deploys a set of polygons to indicate the highlighted area (i.e. IR_PolygonSet). Selecting both the IG Model and IR Model, the user may right click on the selected models to open up the context menu (Figure 7.16).

When the connection is created between the selected IG Model and the IR Model, the user may open the property window of the IR Model to configure the IG-IR model mappings. An
example of mapping the IG Model of a power outage to a Static Polygon Set IR Model is presented in Figure 7.17. When the IG-IR mappings are properly configured, the IR Model is instantiated (Figure 7.18). The user may execute the instantiated IR model to preview its visual output (Figure 7.19). The model preview function is quite handy when the user needs to fine-tune the model.

Figure 7.17  An example of IR Model instantiation

Figure 7.18  Executing an instantiated IR Model to help with adjusting IG-IR mappings

Figure 7.19  Example visual output of an instantiated IR Model
The IR Model configurations can be saved in a component definition file for later reuse. An example of IR Model definition is presented in Figure 7.20. Such definition files enable the user to create new IR Models from existing models. The prototype supports two modes of IR Model reuse, i.e. direct-load and mapping-inheritance. With the former type, the user can directly load an existing IR Model from its definition file without manually recreating the underlying data source and IG Model. The user may then modify the configurations of the loaded model to support different visualization requirements. Compared to the direct-load mode, the latter mode allows the user to apply only the IG-IR mappings of an existing IR Model to the new IR Model. This model mapping-inheritance model is supported by the Load function in the IR Model property window.

Figure 7.20 An example of IR Model definition
It deserves to be pointed out that the input model to an IR Model does not necessarily to be an IG Model. In fact, any PV component definition containing an IG Model can be used as an input to the IR Model. This feature further enables the transformation and integration of visual representations, which will be discussed in detail in the following two subsections.

### 7.4.3 Transform Representation

As no visualization technique is able to show all possible features of complex source data, each IR Model attempts to reveal certain features of the data from its particular perspective. Visualization transformation is essentially about visually encoding the same source data through different IR Models. It enables a user to achieve a more complete view of the underlying data or fulfill different purposes by leveraging the strengths of different IR Models, which more likely leads to a better understanding of the data. For example, the user may consider to use an Animated Polygon Set Model to show when the network faults, unresolved as at a given snapshot time, start and are resolved and in which areas residents are affected (Figure 7.21). Though it provides a view of the start/end time and duration of the network faults occur, it is not helpful if the user simply wants to see all the affected areas as at the snapshot time. Under such context, a Static Polygon Set Model better fits the purpose compared the animated model.

![An example parent IR Model showing the distribution of network faults in an animated fashion](image.png)
To support visualization transformation, the prototype allows creating new IR Models by inheriting the entire IG Model from an existing IR Model. Further to the above example, the user can create a new static polygon model in the design canvas and then assign the Animated Polygon Set Model (source) as an input to the new model (target) (Figure 7.22). Once the input is assigned, the user will notice that the target model owns the same IG Model (IG_OutageDetails) as the source model (Figure 7.23). Configuring the target IR Model, the user may preview what the instantiated target model (Figure 7.24). The visual representation transformation results are presented in Figure 7.25.

![Figure 7.22 Transform visual representation by sharing IG Model](image)

![Figure 7.23 An example of IG Model inheritance](image)
7.4.4 Integrate Representations

As we discussed in section 5.1, visualization integration is required at three different levels: data, representation, and presentation. The data-level integration is already well supported by many existing visualization systems hence is not the focus of our prototype implementation. The prototype supports the data-level integration by allowing users to create new IG Models against a data source combining all the required data. At the representation-level, the prototype realizes visualization integration through dynamic context propagation. Such functionality will be demonstrated with examples in this section.
while the prototype support for the presentation-level visualization integration will be explicated in section 7.5.4.

Dynamic context propagation is concerned with context sharing between a parent IR Model and all directly or indirectly connected child IR Models. The problem and user contexts associated with an instantiated IR Model can be passed on to its connected IR Models if required. The connected model could be an instantiated IR Model or a newly created IR Model that has not been configured yet. Moreover, apart from sharing the overall context, the highlighted context of the parent model at runtime can be propagated to all the child models.

For example, a user may choose a Small Multiples Model as the parent IR Model to display a high-level summary of the impact of a power outage (Figure 7.26). Each yellow bubble indicates an outage snapshot, which represents the total number of customers without power at the snapshot time through the bubble size. The size of each bubble reflects the number of affected customers. Each snapshot is also accompanied with three descriptive fields. For this power outage summary, the upper field above the bubble displays the date time when the outage snapshot is taken while the bottom two fields show the progress of power recovery and the total number of faults occurred as at the time of the snapshot. The user can then use a Polygon Set Model to show a static view of the distribution of areas affected by the power outage (Figure 7.27).

![Figure 7.26 An example of parent IR Model](image)
As demonstrated in Figure 7.27, the user can select both IR Models and assign the context of the outage summary model (the parent model) to the detailed outage distribution model (the child model). Opening up the property window of the child model, the user defines how exactly the parent and child IR Models should be integrated. For instance, the detailed outage distribution model can inherit the same outage snapshot from the parent model context (Figures 7.28-7.29). The model integration configuration allows the visual output of the child IR model to change automatically when the focus of the parent model changes (Figures 7.30-7.31).
Figure 7.29  Execute the linked child IR Model to preview

Figure 7.30  An example of overall context sharing

Figure 7.31  The propagation of the highlighted context at runtime
In the prototype, the same parent IR Model can propagate its context to multiple child IR Models simultaneously. The system also supports multi-level context propagation. That is, a parent IR Model may have multiple levels of child IR Models and pass its context on to all descendant IR Models if required.

The instantiated IR Models may serve as the input to instantiate the IP Models. The system functionalities for supporting IP Model development are illustrated in the subsequent section.

7.5 Information Presentation

An information presentation model defines the way of how visual representations of data are organized and transformed into views that users can display, manipulate, interact with and leverage to support various decision making and problem solving tasks. The prototype offers five template IP Models, which are designed and implemented based on the general presentation models summarized in the subsection 2.6.3. Similar to the implementation of template IR Models, these IP Models are selected for fulfilling the presentation requirements raised from the application domains where our proposed theory and system are validated. In this section, we introduce the template IP Models in subsection 7.5.1 and proceed to demonstrate the creation, instantiation, execution, deletion and reuse of IP Models in subsection 7.5.2. The support for transformation and integration of visual presentations are outlined in subsections 7.5.3 and 7.5.4 respectively.

7.5.1 Template IP Models

By reviewing and synthesizing representative presentation techniques and systems, Bai, White and Sundaram (2015) generalize into five essential presentation models, as discussed in subsection 2.2.3. These models are landscape, semantic layering, nested spaces, sequential scenes, and integrated presentations. Instantiating from these generic presentation models, we have created five template IP Models for the prototype, that is, Deep Zoom Model, Semantic Zoom Model, Spatial Semantic Zoom Model, Timeline Model, and Process Model. Table 7.4 dictates how these template models relate to the generic presentation models. All these template models are designed to support presentation transformation and integration,
and allow users to navigate through the visual contents by zooming, panning, scrolling and rotating.

<table>
<thead>
<tr>
<th>Template IP Model</th>
<th>Relationship to Generic Presentation Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP_DeepZoom</td>
<td>Inherit from the Nested Spaces Model with each nested presentation space indicating a Landscape Model</td>
</tr>
<tr>
<td>IP_SemanticZoom</td>
<td>Inherit from the Semantic Layering Model with each layer represented by a Landscape Model</td>
</tr>
<tr>
<td>IP_SpatialSemanticZoom</td>
<td>Inherit from the Semantic Layering Model with each spatial layer represented by a Landscape Model</td>
</tr>
<tr>
<td>IP_Timeline</td>
<td>Inherit from the Sequential Scene Model</td>
</tr>
<tr>
<td>IP_ProcessModel</td>
<td>Inherit from the Sequential Scene Model</td>
</tr>
</tbody>
</table>

Table 7.4 An overview of template IP Models

_Deep Zoom Model_

A Deep Zoom Model defines a virtual space that contains multiple nested/parallel landscapes with each presenting a two-dimensional display canvas. It allows users to navigate through the involved landscapes by zooming, panning and rotating. A simple example of this is demonstrated in Figure 7.32, which highlights how the power outage affected areas and customers vary over time by presenting the visual represented power outage snapshots. Each outage snapshot is presented as an embedded landscape and is functional to support user interactions such as zooming in/out to show/hide details, panning to focus on particular areas and rotating the whole scene.

Figure 7.32 Power outage snapshots presented in a Deep Zoom Model
**Semantic Zoom Model**

Similar to the Deep Zoom Model, a Semantic Zoom Model also presents visual representations via a series of landscapes. However, such landscapes are organized in a layered fashion, which is unlike the parallel or nested modes in the Deep Zoom Model. That is, users can view only a single landscape layer at a time. It is particularly designed to make the message carried by the visual representations in the same landscape layer more prominent. As a result, each landscape layer is used to highlight different levels of details/features of the underlying data. Moreover, the model allows users to navigate through the landscapes to show details on demand by zooming in and out. For instance, in section 7.4.4 we demonstrated how to visually encode the power outage data through (1) the Small Multiples IR Model (the power outage summary), (2) the Static Polygon Set IR Model (the distribution of outage affected areas and customers) and (3) the Animated Polygon Set IR Model (when network faults occurred and are resolved). Given that each IR Model intends to reflect different features of the power outage data, they can be presented in a three-layer Semantic Zoom Model, which highlights these features via different layers (Figure 7.33).

![Figure 7.33 Power outage visual representations presented in a Semantic Zoom Model](image-url)
**Spatial Semantic Zoom Model**

The Spatial Semantic Zoom Model behaves quite similarly to the Semantic Zoom Model except that each involved layer is presented through a spatial landscape model. For example, the power outage summary and details may be presented in a three-layer Spatial Semantic Zoom Model (Figure 7.34). In particular, the top layer presents a brief summary of the affected customers by power outage snapshot (Figure 7.34); the middle layer shows suburb-based details of affected customers by customer type (Figure 7.35); the bottom layer displays more information through a video of the power outage (Figure 7.36).

---

Figure 7.34   The top layer in an example Spatial Semantic Zoom Model

Figure 7.35   The middle layer in the example Spatial Semantic Zoom Model
Figure 7.36 The bottom layer in the example Spatial Semantic Zoom Model

**Timeline Model**

The Timeline Model organizes visual representations into a chronological sequence of landscapes and presents the landscapes through a flexible linear time scale. In the prototype, it is implemented based on an open-source WPF control, namely timeline viewer (Silverlight & WPF Timeline Control, 2015). An example of this presentation model is provided against the power outage snapshots. As illustrated in Figure 7.37, four key outage snapshots are plotted against the time axis of a Timeline Model based on the time when each outage snapshot is taken. Users may interact with the timeline scale to show/hide details of the outage snapshot representations on demand.

Figure 7.37 Power outage visual representations presented in a Timeline Model
**Process Model**

Like the Timeline Model, a Process Model also inherits from the generic sequential scenes model. It organizes and presents visual representations through a sequence of landscapes. Users may navigate through the landscapes following a predefined logical order. The prototype deploys an open-source WPF control – 3D Tab Carousel (WPF 3D Tab Carousel, 2015) to implement this model. Figure 7.38 shows an example of this model, which presents six power outage snapshots and allows users to view the interested scene by rotating and zooming.

![Image of a Process Model](image)

Figure 7.38 Power outage visual representations presented in a Process Model

The above template IP Models are implemented in the prototype to guide the creation of meaningful visual representations. They can be instantiated and integrated with one another at multiple levels to support various presentation requirements. We will illustrate how IP Models can be created, instantiated, modified, executed and saved for reuse in the following subsection.

### 7.5.2 IP Model Management

A new IP Model can be created either from scratch based on the model templates or by reusing existing IP Models. To create an IP Model from scratch, a user should add the visual representations to be presented (i.e. instantiated IR Models) to the visual scenario design canvas, and choose an appropriate template IP Model to start with. Then, the representations
need to be assigned to the selected IP Model as its input. Once the template IP Model has the input, it can then be instantiated by creating and configuring the content within the IP Manager. For example, we demonstrated a simple Deep Zoom Model that presents key power outage snapshots to highlight how the affected areas and customers various over time in the previous section (Figure 7.32). This IP Model is created from scratch by assigning six instantiated Static Polygon Set IR Models (i.e. visual representations of power outage snapshots) to a template Deep Zoom Model (Figure 7.39). Then the interested outage snapshots serve as building blocks to create and configure the content of the IP Model (Figure 7.40).

Figure 7.39 Assign inputs to a template IP Model

Figure 7.40 Instantiate an IP Model by creating contents from the selected visual representations
Considering the requirements for configuring template IP Models, we implemented two types of IP Manager to accomplish IR-IP mappings. With the first type, creating model content follows a WYSIWYG (what you see is what you get) fashion, for example, the IP Manager for configuring the Deep Zoom Model. It does not require executing the model to preview its content. In contrast, the second type of IP Manager requires configuring the settings of the model content. For instance, in the previous subsection we illustrated a three-layer Semantic Zoom Model that reveals different features of the power outage data via different semantic layers (Figure 7.41). The configuration of such model requires a simple layer setting for each displayed IR Model (Figure 7.42). Once the model is properly configured, it can be executed to preview the model content. The instantiated IP Model can be saved to an XML-based definition file for later reuse. An example of this is given in Figure 7.43, which shows the definition of the above Semantic Zoom Model.
Figure 7.43 An example of the definition of a Semantic Zoom Model
Similar to IR Models, IP Models can be rather easily modified by changing the input models and adjusting the model settings in the IP Manager. It deserves to be pointed out that the input to an IP Model is not necessarily to be an IR Model. Any instantiated IP Models can also serve as valid inputs. This feature lays a foundation for transforming and integrating visual presentations, which will be delineated in the next two subsections.

7.5.3 Transform Presentation

To better support the visualization requirements, cognitive styles and preferences of different users, the prototype allows users to transform visual presentations from one type to another. This is a necessities because different IP Models tend to enable visual presentations to be transformed from one type to another. An example is provided against the requirement of presenting the power outage data. As demonstrated in the previous section, the outage snapshots presented in the Deep Zoom Model highlights how the power outage affected areas and customers vary over time. To transform it into a Timeline Model that reflects when the outage snapshots are created, this Deep Zoom Model can be assigned to the Timeline Model as an input. It, in turn, enables the involved Polygon Set Models (i.e. outage snapshot representations) to be reconfigured in the IP Manager and then presented against a time scale (Figure 7.44).

Figure 7.44 An example of presentation transformation from Deep Zoom to Timeline
7.5.4 Integrate Presentations

As no presentation models are able to address all visualization requirements, the prototype enables to combine the existing visual representations and presentations to form a rich view of the underlying data. The presentation level integration may involve both IR Models integration via context propagation and IP Models integration. There is no limitation against what type of IR or IP Scenarios can be integrated into a new IP Model. This functionality is demonstrated through integrating the three example presentation models into a Process Model. In particular, in subsection 7.5.1 we demonstrated a three-layer Spatial Semantic Zoom Model that presents both the power outage summary and details (Figure 7.34); a Timeline Model that shows key power outage snapshots (Figure 7.37); and a Process Model that displays all outage snapshots (Figure 7.38). For illustrating the system support for visual presentation integration, we integrate these three models into a new Process Model. These instantiated IP models are assigned to the new model as inputs (Figure 7.45). When the new model is properly configured, users may preview how the three input IP Models are nested within the new Process Model (Figure 7.46).

![An example of visual presentation integration](image-url)
Figure 7.46 Integrating multiple instantiated IP Models into a new IP Model

So far, we have demonstrated how the prototype supports the distinguishing features and system requirements of PVS with regard to information generation, representation and presentation. The demonstrations of the prototype validate our proposed PVS concepts, frameworks and architectures at the system functionality level. The capabilities of the prototype also lay a solid foundation for (1) achieving the flexibility required for purposeful visualization development, (2) accommodating the diversity of stakeholders, purposes and contexts and (3) sustaining visualization effectiveness under various context changes. To validate the prototype and other research artefacts of purposeful visualizations, we proceed to discuss how they are evaluated to ensure the rigor and relevance of our research in the following chapter.
CHAPTER 8 PURPOSEFUL VISUALIZATION ARTEFACTS EVALUATION

Our research on purposeful visualizations and purposeful visualization systems has produced a number of artefacts, as presented in chapters 4-7. These artefacts are PV definition, PV Model, PVDM process, PVS definition, PVS structural and behavioral frameworks, PVS high level and detailed level architectures, and the purposeful visualization system prototype. To validate them, we adopted a variety of evaluation methods in our research methodology design, as discussed in section 3.4. A summary of these evaluation methods is provided in Table 8.1.

<table>
<thead>
<tr>
<th>Evaluation Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Analysis</td>
<td>Assessing the fit of PVS frameworks and architectures into technical IS architectures.</td>
</tr>
<tr>
<td>Prototyping</td>
<td>Validating the proposed theory of PV and PVS through system implementations.</td>
</tr>
<tr>
<td>Static Analysis</td>
<td>Examining the prototype at unit, technology, and system levels.</td>
</tr>
<tr>
<td>Functional Testing</td>
<td>Testing how the key functionalities of the prototype in regards to visualization creation, modification, enhancement, integration, transformation and adaptation.</td>
</tr>
<tr>
<td>Structural Testing</td>
<td>Testing the internal structure of the prototype.</td>
</tr>
<tr>
<td>Computer Simulations</td>
<td>Testing and executing the prototype and key functionalities with artificial data.</td>
</tr>
<tr>
<td>Case Study</td>
<td>Using a typical case in electricity distribution businesses to examine the application of the PVDM process and the prototype.</td>
</tr>
<tr>
<td>Illustrative Scenarios</td>
<td>Demonstrating how the prototype supports PVS distinguishing features and how it supports developing PV and sustaining visualization effectiveness.</td>
</tr>
<tr>
<td>Informed Argument</td>
<td>Evaluating the support for PVS distinguishing features among the prototype and other typical visualization systems.</td>
</tr>
<tr>
<td>Expert Evaluation</td>
<td>Presenting, publishing and validating our research artefacts through peer (system, domain and academic experts) reviewed sessions, seminars, conferences and journals.</td>
</tr>
</tbody>
</table>

Table 8.1 Evaluation methods

For each research artefact, one or more evaluation methods are employed to assess its validity according to the nature and evaluation requirements of the research artefact. A
summary of our research artefacts and their selected evaluation methods is presented in Table 8.2. In this chapter, we elaborate on the evaluation of each research artefact in detail in sections 8.1 - 8.5.

<table>
<thead>
<tr>
<th>PV Definition</th>
<th>PV Model</th>
<th>PVDM Process</th>
<th>PVS Definition &amp; Features</th>
<th>PVS Frameworks</th>
<th>PVS Architectures</th>
<th>PVS Prototype</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td>8.4.1</td>
<td>8.4.1</td>
<td></td>
</tr>
<tr>
<td><strong>Prototyping</strong></td>
<td>8.2.1</td>
<td>8.3.1</td>
<td>8.4.2</td>
<td>8.4.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Static Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5.1</td>
<td></td>
</tr>
<tr>
<td><strong>Functional Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5.2</td>
<td></td>
</tr>
<tr>
<td><strong>Structural Testing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5.3</td>
<td></td>
</tr>
<tr>
<td><strong>Computer Simulations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5.4</td>
<td></td>
</tr>
<tr>
<td><strong>Case Study</strong></td>
<td>8.2.2</td>
<td></td>
<td>8.3.2</td>
<td></td>
<td>8.5.5</td>
<td></td>
</tr>
<tr>
<td><strong>Illustrative Scenarios</strong></td>
<td>8.2.3</td>
<td></td>
<td>8.3.3</td>
<td></td>
<td>8.5.6</td>
<td></td>
</tr>
<tr>
<td><strong>Informed Argument</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.5.7</td>
<td></td>
</tr>
<tr>
<td><strong>Expert Evaluation</strong></td>
<td>8.1</td>
<td>8.1</td>
<td>8.2.4</td>
<td>8.3.4</td>
<td>8.4.3</td>
<td>8.4.3</td>
</tr>
</tbody>
</table>

Table 8.2 Research artefacts evaluation

8.1 Purposeful Visualization – Definition & Model

The definition and model of purposeful visualizations address the research question, “What are purposeful visualizations?” We validated the PV concepts mainly through gathering
feedback and suggestions from domain/system experts coming from a variety of disciplines. Such feedback, in turn, deepens our understanding of purposeful visualizations and assists in refining the PV concepts. To support the feedback collection and achieve a thorough understanding of purposeful visualizations, we conducted a seminar titled “Beautiful Evidence: Contextual Measures for the Ephemeral” at the University of Auckland on 23 December 2009. We invited people from a wide range of fields/disciplines such as business, science, engineering, the arts, psychology, education and industry. In the seminar, we presented the audience with a variety of visualizations ranging from the good to the bad to the ugly. Their discussions and feedback around the beautiful evidence provided useful and rich data for understanding and evaluating purposeful visualizations.

In this section, we present a brief description of how this seminar was conducted in subsection 8.1.1., and then analyze the data collected from the seminar in subsection 8.1.2. We proceed to discuss how the findings gained from the seminar reinforce the PV concepts in subsection 8.1.3.

8.1.1 Beautiful Evidence Seminar

The Beautiful Evidence seminar was conducted to collect useful ideas/suggestions for addressing a key issue - what can enable visualizations to provide illuminating, useful, relevant and actionable information. People from divergent fields and disciplines were invited to attend this seminar and have an open conversation around the following questions:

- What is beautiful evidence?
- What does it mean to say a particular visualization is beautiful?
- How can beauty help in understanding and making decisions based on evidence?
- What visual elements or components make a visualization beautiful?
- What criteria do you use to evaluate a visualization?
- How do you create beautiful evidence?

Their discussions and feedback around these questions provided useful and rich data for analyzing the key issue.

This seminar was organized into five main sections: exhibition of evidence, quiet reflection, active engagement, background & demo, and re-engagement. In the exhibition of evidence
part, the key questions to be explored were first explained to the seminar attendees, and then a large set of visualizations were presented to them. The attendees were asked to reflect on these visualizations and figure out their own answers to the key questions. In addition, handouts with a pre-defined set of visualizations were given to the attendees to capture their feedback on how beautiful/ugly/effective/ineffective they may think each visualization was with considering their own purposes of using the visualization. After the quiet reflection, the attendees were invited to share their ideas with others and participate in the discussions on the key questions. Next, the research on visual intelligence density already conducted by the researchers was introduced briefly and an example system demonstrating how VID could be improved was presented. With the understanding of how higher density of visual intelligence may be achieved, the attendees were asked to reengage with the key questions and comment on the visual elements or components that can make a visualization beautiful/effective.

The visualizations in the handouts were selected from a wide range of sources such as paintings, hand drawings, photos, cartoons, sculptures, information visualization systems, and so on. Seminar attendees were asked to evaluate these visualizations from two perspectives, i.e. beauty and effectiveness. The first measure (beauty) reflects how beautiful/ugly a visualization is for a viewer. The second measure (effectiveness) indicates the capability of a visualization in terms of conveying the intended message of its designer/presenter. The rating mechanism deployed by these measures is a 7-point Likert scale ranging from 1 (ugly or ineffective) to 7 (beautiful or effective). The attendees were also asked to justify their ratings regarding the beauty and effectiveness of the selected visualizations.

Their discussions and feedback around beautiful, effective and purposeful visualizations collected during the seminar provided useful and rich data. The following subsection discusses how the collected data were analyzed.

### 8.1.2 Seminar Data Analysis

We designed and implemented a prototype to aid in understanding, analyzing, and evaluating visualizations. This prototype allows us to store both visualizations and people’s perceptions
of the visualizations, such as their opinions on the effectiveness/beauty of a particular visualization and why they think the visualization is effective/beautiful. It also presents the audience feedback on visualizations in graphs such as the distribution of the visualizations against various evaluation criteria.

The prototype is composed of two subsystems: data collection and data analysis. The data collection subsystem records the data around six areas: viewers, visualizations, visualization measures, visualization assessments, assessment criteria and visual elements (Figure 8.1). Among these areas, the first four store measure ratings given by viewers against the selected visualizations, while the last two record the comments justifying the ratings.

The data analysis subsystem facilitates us to analyze the collected viewer feedback on visualizations. This subsystem integrates functionalities like presenting overviews of all measure ratings, analyzing ratings by viewer/visualization, plotting rating distributions and querying assessment criteria (Figure 8.2).
8.1.3 Discussion

In this seminar, we collected rich feedback from the audience, which incorporates the information around two essential areas: measure ratings and visualization assessment criteria. During the exhibition of evidence, we observed and recorded the audience response to different presentation styles used by the presenters, e.g. quiet presentation with no explanation on the visualizations displayed, presentation with reasonable amount of explanations, and teaching the audience via explaining the visualizations in detail. Within the quiet reflection section, the measure ratings given by the audience and their justifications of the ratings were collected through the handouts. With the active engagement and the later re-engagement with the audience, their discussions and comments on the above six questions were recorded.

Measure Ratings

An overview of all measure ratings captured through the seminar is presented in Figure 8.3. This rating overview is like a table with using visualizations as column headers on the top and seminar attendees as the row headers on the left. In each cell of this table, two rating bars are used to indicate the ratings of the two measures (i.e. beauty and effectiveness). Figure 8.3 illustrates that the measure ratings of beauty and effectiveness are not directly correlated. That is, high beauty ratings are not always accompanied with high or low effectiveness ratings, and vice versa. The measure ratings given to the same visualization may vary considerably among the attendees.
Inspired by Tufte’s (2006) idea of improved scatter diagrams, we implemented a scatter diagram to present the average measure ratings of all visualizations (Figure 8.4). As implied by the distribution of these ratings, beautiful visualizations are not always effective and effective visualizations are not always beautiful. Nonetheless, visualization beauty and effectiveness tend to be positively correlated. As illustrated in Figure 8.5, we identified four representative visualization groups: visualizations presenting high effectiveness but looking ugly; visualizations that are effective and beautiful; visualizations looking beautiful but ineffective; and ineffective and ugly visualizations. The analysis of these groups highlighted the fact that even the most effective and striking visualizations, as assessed by the world, may be considered as ineffective to a viewer if he does not understand the context and purpose of the visualization.
Figure 8.4 Distribution of average measure ratings (image view), from author’s creation (Bai, White & Sundaram, 2011c)

Figure 8.5 Four representative visualization groups, from author’s creation (Bai, White & Sundaram, 2011c)
Visualization Assessment Criteria

After conducting this seminar, we collected a rich set of data about visualization assessment criteria, which comprises over 500 written justifications of measure ratings as well as a one-hour video recording the open conversations among the seminar attendees. By analyzing the collected data, we roughly categorized the collected evaluation criteria into six major groups, i.e. visualization design, details/context, information quality, visual impact, pre-knowledge and cognitive/personal characteristics. Examples of these groups and their associated original feedback from the seminar attendees are provided in Figures 8.6 and 8.7.

![Figure 8.6](image1.png) Audience feedback regarding visualization design, from author’s creation (Bai, White & Sundaram, 2011c)

![Figure 8.7](image2.png) Audience feedback on details/context provided by visualizations, from author’s creation (Bai, White & Sundaram, 2011c)

The Evaluation Taxonomy

By analyzing and generalizing the collected audience feedback, we identified a set of visualization evaluation criteria that covers seven essential assessment areas. They are visual representation, information presentation, psychology of the observer, information quality,
visual impact, overall design style, and overall performance. Under each assessment area, there are a set of evaluation categories and sub-categories. In addition, we also reviewed and synthesized some existing representative visualization evaluation metrics/criteria/frameworks such as the works of Brath (1997), Yang-Pelaez and Flowers (2000), Freitas et al. (2002), O’Connell and Choong (2008), and Bai, White and Sundaram (2009a). Integrating the findings of the seminar with the literature on visualization evaluation lead us to propose a taxonomy that can be useful for designing and evaluating purposeful visualizations (Figure 8.8).

![Visualization Evaluation Taxonomy](image)

Figure 8.8  A visualization evaluation taxonomy, from author’s creation (Bai, White & Sundaram, 2011c)

The above findings from the seminar underpin the fundamentals of the PV definition and model. Additionally, they were communicated to wider academic and research communities, which is briefly discussed in the following subsection.
8.1.4 Evaluation by Experts

To communicate the concepts of PV to the wider academic and research communities, we published our findings obtained from the Beautiful Evidence seminar through the paper listed in Table 8.3. Presenting the paper at the HICSS 2011 (Hawaii International Conference on System Sciences) conference allowed us to gather valuable feedback and suggestions on purposeful visualizations from system, domain and academic experts. To operationalize the concepts of PV, we proposed a process to guide the application of purposeful visualizations under the context of decision making. The evaluation of this process is detailed in the next section.

<table>
<thead>
<tr>
<th>Presentation</th>
<th>Publication</th>
</tr>
</thead>
</table>

Table 8.3 Presentation and publication for validating purposeful visualization concepts

8.2 Purposeful Visual Decision Making Process

The purposeful visual decision making (PVDM) process guides the development and application of purposeful visualizations to support decision making under complex and changing visualization contexts. It answers the research question, “How can we apply purposeful visualizations to support various tasks in decision making?” This process was validated through four research evaluation methods: prototyping, case study, illustrative scenarios, and expert evaluation. Different methods evaluate the proposed process from different perspectives. In particular, the prototyping approach focuses on validating the development phase of the rational decision making process and prove the feasibility of adapting visualizations to the context changes caused by the naturalistic decision making. The case study approach validates the PVDM process by demonstrating its application to a typical decision problem in electricity distribution businesses. The illustrative scenarios included in the case demonstration reinforces the validity of the PVDM process. The process was also presented in publications to collect valuable feedback from wider academic and research
communities. We explicate how these four evaluation methods were applied to validate the PVDM process in subsections 8.2.1 – 8.2.4.

8.2.1 Evaluation by Prototyping

The prototyping approach is deployed to validate two parts of the PVDM process, as highlighted in Figure 8.9. Firstly, it proves the validity of the development phase of the rational process, i.e. developing purposeful visualization through information generation, representation and presentation. Secondly, it proves the feasibility of utilizing the prototype to handle the impact caused by the naturalistic process.

![Figure 8.9 The evaluation of the PVDM Process by prototyping](image)

As we demonstrated in chapter 7, the prototype supports the fundamental phases of the information visualization process: information generation, representation and presentation. We also demonstrated its support for all the distinguishing features of PVS: visualization creation, modification/enhancement, integration, transformation, and context adaptation. Such features provide the system flexibility required by purposeful visualization development,
which is particularly vital to the development phase in the rational side of the PVDM process. Furthermore, the system features of the prototype facilitate adapting visualizations to various context changes so as to maintain their effectiveness. The support of the prototype for problem, stakeholder, purpose and time context changes was demonstrated in section 6.4. We proved the capability of the prototype to assist decision makers with handling the visualization context and requirement changes caused by the naturalistic process. This is further corroborated by the application of the PVDM process to support a real world decision problem, which is discussed in the following subsection.

### 8.2.2 Evaluation by Case Study

In section 6.5, we introduced the decision problem of understanding and quantifying how future network demands are affected by influential forecasting factors such as growth in population, air conditioning demand and employment. Solving this problem is a prerequisite to addressing the complex problem faced by many electricity distribution businesses, that is, how to choose the optimal solution to upgrade a problematic electricity network area. To demonstrate the support of the PVDM process, we applied it to support visualizing the impact of different sets of influential forecasting factors on network demand forecasts. In our demonstration in section 6.5, we outlined the key tasks that should be performed at every stage in the rational PVDM process. Moreover, following this decision making process, we illustrated how to create various presentation models in the prototype and achieve the flexibility with presenting and comparing multiple forecast trials in order to fulfill different visualization purposes. The flexibility of information presentation is a key requirement for dealing with the visualization context and requirement changes caused by naturalistic decision making. The case study contributes to validating the whole PVDM process, which is further strengthened by the scenarios discussed in the following subsection.

### 8.2.3 Evaluation by Illustrative Scenarios

The illustrative scenarios approach contributes to validating the PVDM process by providing evidence that the PVDM process can guide decision making with supported by the purposeful visualization system prototype. More specifically, in section 6.5 we presented a sequence of
scenario-driven illustrations against the typical decision problem of understanding and quantifying how future network demands are affected by influential forecasting factors. These illustrations demonstrate how various forecast trials can be presented to assist decision makers with understanding and interpreting the problem; they show how the visual presentations of forecast trials can be modified, transformed, integrated and adapted to support different purposes and requirements. These illustrations prove that a possible way to apply the PVDM process to guide decision making is by leveraging the prototype.

Apart from validating the PVDM process by prototyping, case study and illustrative scenarios, we also communicated this artefact with wider academic communities through publications, which is discussed in the following subsection.

8.2.4 Evaluation by Experts

As we presented in section 4.3, the PVDM process is composed of both rational and naturalistic decision making processes. The rational process delineates the ideal situation of decision making by simplifying the complexities in decision making, while the naturalistic process highlights the influential factors of decision making in the real world such as human emotion, intuition, inspiration, imagination, and bias. The two processes together reflect the real world decision making practice. We communicated the ideas and concepts of the PVDM process with wider academic and research communities through the publications in Table 8.4 to obtain valuable feedback from system, domain and academic experts.

As we pointed out in this section, a possible way to apply the PVDM process is by leveraging the support from purposeful visualization systems. Therefore, it is crucial for us to validate the concepts, frameworks, architectures and implementation of PVS. We present the evaluation of the PVS definition and distinguishing features in the subsequent section.
The Rational Side of the PVDM Process

<table>
<thead>
<tr>
<th>Presentation</th>
<th>Publication</th>
</tr>
</thead>
</table>

Table 8.4 Presentations and publications for the PVDM process

8.3 Purposeful Visualization Systems – Definition & Distinguishing Features

The PVS definition and distinguishing features portray how a purposeful visualization system should behave and offer to support the development of purposeful visualizations. They contribute to addressing the following research questions by highlighting the critical functionalities that are required for achieving sufficient system flexibilities:

- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?

Similar to the way of how the PVDM process is validated, we evaluate the PVS definition and distinguishing features by the same set of methods: prototyping, case study, illustrative
scenarios, and expert evaluation. We describe how these four evaluation methods were deployed to validate the PVS definition and distinguishing features in subsections 8.3.1 – 8.3.4.

8.3.1 Evaluation by Prototyping

As we highlighted in section 2.7, to support the changing stakeholders, purposes and contexts of visualizations and sustain visualization effectives, visualization systems should enable stakeholders to create, modify, integrate, transform and adapt visualizations in a flexible manner. A summary of these system requirements is presented in Table 8.3. Accordingly, we proposed the concept of purposeful visualization systems as one possible way to accommodate these requirements. The distinguishing system features of PVS are designed specifically for operationalizing these requirements.

<table>
<thead>
<tr>
<th>System Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization Creation</td>
<td>To develop new visualizations either from scratch or by reusing existing visualizations or visual components.</td>
</tr>
<tr>
<td>Visualization Modification</td>
<td>To flexibly modify/customize/enhance visualizations.</td>
</tr>
<tr>
<td>Visualization Integration</td>
<td>To flexibly combine the visual contents generated by different visualization techniques in order to present a rich view of the underlying data.</td>
</tr>
<tr>
<td>Visualization Transformation</td>
<td>To transform visualizations from one type to another in a flexible and seamless manner with less time and effort.</td>
</tr>
<tr>
<td>Visualization Context Adaptation</td>
<td>To adapt visualizations to problem/stakeholder/purpose/context changes so as to sustain visualization effectiveness.</td>
</tr>
</tbody>
</table>

Table 8.5 A summary of critical system requirements of PVS

As illustrated in Figure 8.10, the visualization creation requirement is addressed primarily by the PVS features of information generation, representation creation, and presentation creation. The visualization modification requirement is supported by the visual representation modification and presentation modification features of PVS. Visualization transformation can happen at both visual representation and presentation levels. Likewise, visualization integration may be realized through visual representation integration and presentation integration. The requirement for visualization context adaptation is fulfilled by
the context adaptation capabilities at information presentation level together with the system functionalities about visual representation integration and transformation.

We implemented a fully-fledged prototype to validate the definition and features of PVS, which, in turn, enabled us to further validate the PVS concept through a case study in chapter 6 and a series of illustrative scenarios in chapter 7. We describe the case study based evaluation in subsection 8.3.2 and the scenario-based evaluation in subsection 8.3.3.

8.3.2 Evaluation by Case Study

In chapter 6, we introduced a case of analyzing network reliability and improving network performance, which is common in many electricity distribution businesses. For this case, we conducted in-depth and detailed analyses about the interested decision problems and their visualization contexts, and selected several typical visualization requirements for
demonstration purposes. We validated the concept of PVS through demonstrating how the features of the prototype may facilitate stakeholders to develop purposeful visualizations under diverse visualization contexts (in section 6.3) and deal with various problem/stakeholder/purpose/time context changes (in section 6.4).

Furthermore, the case study provides a good basis for demonstrating each individual distinguishing feature of PVS. Specifically, among the decision problems highlighted in section 6.2, we picked up the problem of identifying and presenting the impact of a power outage for all the illustrations of PVS features in chapter 7. This is further expanded in the subsection below.

8.3.3 Evaluation by Illustrative Scenarios

To validate the concept of PVS, we adopted illustrative scenarios to demonstrate how each distinguishing feature of PVS can be implemented and applied in chapters 6 and 7 (Figure 8.11). Most of the feature illustrations are based on the selected decision problem of identifying and presenting the impact of a power outage. More specifically, at the information representation level, we demonstrated how the power outage data could be visually encoded through different types of IR Models for different visualization purposes: the Small Multiples Model that shows the outage summary information (Figure 7.26); the Static Polygon Set Model that presents a snapshot of the outage affected areas as at a certain time during the outage (Figure 7.18); the Animated Polygon Set Model that displays the involved network faults in an animated style based on the time when they occurred (Figure 7.21). Through these examples, we illustrated how these IR Models can be created, instantiated, modified, executed, integrated, transformed, and saved for later reuse.

Furthermore, we validated and demonstrated the system features of information presentation by organizing and exhibiting these IR Models in various ways: displaying both the outage summary and details in a Spatial Semantic Zoom Model (Figures 7.34-7.36); portraying key outage snapshots in a Timeline Model (Figure 7.37); and presenting all outage snapshots in a Process Model (Figure 7.38). Through these scenarios, we demonstrated how IP Models could be developed and adapted to fulfill different purposes.
Apart from validating the PVS concept by prototyping, case study and illustrative scenarios, we communicate it with wider academic and research communities for further improvements.

### 8.3.4 Evaluation by Experts

Suggestions and feedback, gathered from system, domain and academic experts, are highly valuable as they may help to identify hidden issues and new opportunities with our research and refine our research artefacts. This again allows us to assess and ensure the relevance, rigor and validity of our research. Hence, we presented and published the concepts and distinguishing features of PVS and collected the feedback from experts coming from various disciplines such as information visualization, decision support systems, computer science, and knowledge management. A summary of the presentations and publications for these research artefacts is listed in Table 8.6.

To operationalize the concept of PVS, we proposed frameworks and architectures to guide the design and development of a purposeful visualization system in sections 5.2 and 5.3 respectively. We elucidate the validation of the PVS frameworks and architectures in the following section.
Table 8.6  Presentations and publications for PVS definition and distinguishing features

<table>
<thead>
<tr>
<th>Presentation</th>
<th>Publication</th>
</tr>
</thead>
</table>

8.4 Purposeful Visualization Systems – Frameworks & Architectures

The PVS frameworks and architectures guide the design and development of a purpose visualization system. They help to address the following research questions by outlining a feasible way to accommodate the distinguishing features and requirements of PVS and support the design and development of purposeful visualizations:

- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?

The PVS frameworks and architectures are evaluated by three essential methods: architecture analysis, prototyping, and expert evaluation. The validation against the system designs of PVS is explicated in subsections 8.4.1-8.4.3.
8.4.1 Evaluation by Architecture Analysis

To help with validating the PVS frameworks and architectures, we carried out architecture analysis for each development iteration of the purposeful visualization system prototype. This method is adopted in our evaluation for three main reasons. Firstly, it allows us to assure that the PVS distinguishing features are fully supported and to detect any flaws with system designs at an earlier stage during a prototype development iteration. Secondly, it facilitates us to follow good practices for visualization system designs. Thirdly, it contributes to improving the overall usability, effectiveness and sustainability of PVS. According to Almorsy, Grundy and Ibrahim (2013), architecture analysis can be conducted by two main techniques: (1) creating a set of evaluation scenarios against system requirements, and (2) developing metrics to assess software architecture. We adopted the first technique by setting up a set of evaluation scenarios to ensure all the known PVS features and requirements are addressed. We performed the first round of architecture analysis against the PVS designs before starting the prototype development. Each development iteration deepened our understanding of the system features and requirements of PVS. Accordingly, we improved the system framework and architecture designs in the following development iteration and conducted the analysis again before making any improvements/changes to the prototype.

8.4.2 Evaluation by Prototyping

To validate the proposed PVS frameworks and architectures, we implemented a fully-fledged purposeful visualization system, as delineated in chapter 7. This prototype follows the detailed level PVS architecture design and supports all the distinguishing features of PVS. With the purpose of identifying and presenting the affected areas in a power outage, we presented a series of scenario-driven illustrations of how the prototype accommodates the system requirements for information generation, representation and presentation in chapter 7. Furthermore, using the case of analyzing network reliability and improving network performance, we demonstrated the application of the prototype to (1) develop purposeful visualizations under diverse stakeholders, purposes and contexts (section 6.3), (2) sustain visualization effectiveness under various context changes (section 6.4) and (3) support the purposeful visual decision making process (section 6.5).
8.4.3 Evaluation by Experts

The PVS framework and architecture designs have evolved and been improved over time. During this continuous improvement process, we communicated the system designs with wider academic and research communities. The valuable feedback obtained from system, domain and academic experts further helped us to refine the PVS frameworks and architectures. The presentations and publications for PVS frameworks and architectures are summarized in Table 8.7.

<table>
<thead>
<tr>
<th>Presentation</th>
<th>Publication</th>
</tr>
</thead>
</table>

Table 8.7 Presentations and publications for the frameworks and architectures of PVS
A common method used for validating all research artefacts of PVS is through prototyping. Hence, it is crucial for us to evaluate the purposeful visualization system prototype and ensure its validity and relevance. The evaluation of the prototype is discussed in the following section.

8.5 **The Purposeful Visualization System Prototype**

The purposeful visualization system prototype was implemented based on the proposed PVS frameworks and architectures. Whenever there were major improvements made to the system architectures of PVS, the prototype was adjusted accordingly throughout the iterations of prototype development. The prototype implementation validates the proposed theory of PVS and supports users to develop purposeful visualizations and sustain their effectiveness when visualization stakeholders, purposes and contexts change over time. It particularly contributes to addressing the following research questions:

- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?

Due to the critical role that the prototype plays in validating other research artefacts, we adopted eight methods to evaluate the prototype. These evaluation methods are static analysis, structural testing, functional testing, computer simulations, case study, illustrative scenarios, informed argument, and expert evaluation. How the prototype was validated by these methods is discussed in subsections 8.5.1-8.5.8.

8.5.1 **Evaluation by Static Analysis**

To validate the purposeful visualization system prototype, static analysis was performed against the source code of the prototype at unit, technology and system levels. It is used to identify potential design/performance/reliability/security defects in the system source code by referring to best programming practices. For our prototype, such analysis was conducted regularly throughout the system development process by using its code analysis tool in Microsoft Visual Studio (Microsoft, 2015c). An example of the output of static analysis is presented in Figure 8.12.
Figure 8.12   An example of the output of static code analysis

An important feature of static code analysis is that this evaluation method is conducted without actually executing a program. In contrast to the static analysis, the evaluation performed by functional and structural testing does require executing the prototype source code. Functional and structural testing are briefly presented in subsection 8.5.2 and 8.5.3 respectively.

8.5.2 Evaluation by Functional Testing

Functional testing aims to evaluate against the coverage of all the predefined system specifications and examine whether a certain system feature is well supported. We adopted this method to assess how well the purposeful visualization system prototype can support the key functionalities of visualization creation, modification, enhancement, integration, transformation and adaptation. More specifically, we created a series of test cases against the main system specifications of the prototype (refer to Table 7.1). These test cases were implemented by leveraging the unit test tools in Microsoft Visual Studio (Microsoft, 2015c), which allowed us to automate and reuse the test cases throughout the prototype development process.

Functional testing is a typical type of black box testing that focuses on evaluating the overall behaviors of a system. It is often used together with structural testing, which is a type of white box testing and complements functional tests by examining the detailed internal design of the system. We briefly outline how structural testing was used in our research in the next subsection.
8.5.3 Evaluation by Structural Testing

We performed structural testing against the internal structure of the prototypical purposeful visualization system to detect flaws or bugs in the prototype implementation. Such testing was performed predominantly against the control flows, data flows and prime paths within/among different units and subsystems of the prototype. Similar to functional testing, the test cases created for structural testing were also implemented by using unit test tools in Microsoft Visual Studio (Microsoft, 2015c) and reused throughout the prototype development process. Unlike functional tests, performing structural testing needs an in-depth understanding of the system source code and ensure test cases cover all system units of the prototype. An effective complement to both functional testing and structural testing is the evaluation performed through computer simulations, which is delineated in the following subsection.

8.5.4 Evaluation by Computer Simulations

To facilitate the evaluation of the purposeful visualization system prototype, we deployed computer simulations to test its essential system functionalities with artificial data. Such data not only hide business sensitive information but also reveal the complexities involved in the selected typical decision problems involved in outage impact identification, network reliability analysis, problem network area discovery, and network improvement strategy development (refer to section 6.2). Specifically, they involve spatial, temporal and multi-dimensional features and require a variety of representation and presentation techniques to accommodate various visualization purposes and requirements. Computer simulations were conducted throughout the prototype development process to ensure the system performance, flexibility, robustness and reliability. All the representations and presentations demonstrated in this thesis were created based on the artificial data.

8.5.5 Evaluation by Case Study

To prove the validity of the purposeful visualization system prototype, we applied it to support the case of analyzing network reliability and improving network performance, which
is discussed in detail in chapter 6. We demonstrated the support of the prototype to fulfill a variety of visualization purposes, such as presenting a power outage, monitoring network faults and reliability, forecasting future demands, and showing how demand forecasts are affected by the growth in population, air conditioning demand and employment. By constructing visual scenarios in this prototype, we demonstrated how the prototype helps stakeholders to develop purposeful visualizations under diverse visualization contexts (in section 6.3) and deal with various visualization context changes (in section 6.4).

Selecting a single visualization requirement from this case (i.e. presenting the impact of a power outage), we illustrated how the prototype supports the distinguishing feature of PVS, which is further expanded in the following subsection.

### 8.5.6 Evaluation by Illustrative Scenarios

To help with validating the purposeful visualization system prototype, we created a series of illustrative scenarios that demonstrate how each distinguishing feature of PVS can be supported in the prototype (refer to chapter 7). These scenarios all contribute to fulfilling the visualization purpose of presenting the impact of a power outage. They illustrate how the power outage impact can be visual encoded through different types of IR Models and then presented to meet different visualization purposes by employing different IP Models in the prototype. Though these scenarios, we demonstrated how the prototype supports visualization creation, modification, customization, integration, transformation and adaptation. Demonstrating these system features facilitates us to analyze the strengths and weaknesses of the prototype in the next subsection.

### 8.5.7 Evaluation by Informed Argument

We compared the purposeful visualization system prototype to other typical visualization systems in terms of the support for the distinguishing features of PVS. This is particularly helpful to understand the strengths and weaknesses of the prototype. According to our evaluation, a significant strength of the prototype is that it successfully decouples information representation and presentation, which enables the prototype to achieve the flexibilities
required for purposeful visualization development. As we discussed in subsection 2.6.4, many existing visualization systems fail to accommodate the requirement of developing flexible visual presentations, albeit this requirement is indispensable for supporting the changing visualization stakeholders, purposes and contexts. We highlighted the strengths and weaknesses of some typical visualization systems in regard to fulfilling this requirement in Table 2.5. Building on top of this, we further expanded the evaluation by analyzing the support of the prototype against the same set of system requirements (Table 8.8). The strong support of the prototype for visual presentation development is again demonstrated and proved by the scenario-driven illustrations presented in chapters 6 and 7.

Furthermore, it deserves to be pointed out that the overall support that the prototype provides for information generation is limited compared to some other visualization systems. Nevertheless, this does not prevent us from validating the proposed theories of PV and PVS since the focal problems and issues in visual decision support reside in the representation and presentation of information.

Apart from the above evaluation methods that we have discussed in this section, we also sought feedback from system, domain and academic experts. The evaluation by experts is briefly discussed in the subsequent subsection.

### 8.5.8 Evaluation by Experts

We demonstrated the support of the purposeful visualization system prototype through a series of presentations and publications to collect valuable feedback from system, domain and academic experts. Their feedback helped us to improve the design and implementation of the prototype and further refine the theories of PV and PVS. The major presentations and publications for the prototype are listed in Table 8.9.

In this chapter, we outlined the essential research artefacts of PV and PVS, highlighted how they contribute to address the research questions, and specified how they were validated to ensure their rigor and relevance. We proceed to conclude the whole thesis in the following chapter.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Landscape</strong></td>
<td><strong>Nested spaces</strong></td>
<td><strong>Semantic layering</strong></td>
<td><strong>Sequential scenes</strong></td>
<td></td>
</tr>
<tr>
<td>Microsoft Excel</td>
<td>weak support</td>
<td>strong support</td>
<td>limited support</td>
<td></td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reporting Services</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP BusinessObjects</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Crystal Reports</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP Xcelsius</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAP BW</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPSS Clementine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAS Enterprise Miner</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tableau</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xtraction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIMMS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lavastorm</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purposeful Visualization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Prototype</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.8 An evaluation against the system support for information presentation
<table>
<thead>
<tr>
<th>Presentation</th>
<th>Publication</th>
</tr>
</thead>
</table>

Table 8.9  Presentations and publications for the purposeful visualization system prototype
CHAPTER 9     CONCLUSION

This research explored the focal problems, issues and requirement in the interdisciplinary field of visual decision support. Some of these problems, issues and requirements originate from the ever-changing facets of visualizations (i.e. purpose, context, and stakeholder), while others are caused by the increasingly high complexities involved in decision problems. To solve these problems, address these issues and fulfill these requirements, we proposed concepts, models, processes, frameworks, and architectures of purposeful visualizations (PV) and purposeful visualization systems (PVS), and implemented prototypical purposeful visualization systems as proof of concept.

We conclude this thesis by summarizing the research as a whole, identifying the key contributions and highlighting the limitations and potential future research opportunities. Specifically, in section 9.1, we summarize the research by outlining (1) the problems, issues and requirements pertinent to purposeful visualization of information, (2) our research objectives, (3) the research methodology that has guided us to fulfill the objectives, (4) the essential research artefacts of purposeful visualizations and purposeful visualization systems, and (5) how our research artefacts have been validated to ensure their rigor and relevance. We then elucidate the theoretical and practical contributions of our research in section 9.2. Finally, we discuss the existing limitations and future directions of our research in section 9.3.

9.1 Summary of Research

In this research, we reviewed visualization concepts, models, technologies and applications from the intertwined triumvirate perspectives of purpose, context, and stakeholder. Based on the review, we synthesized the focal problems and issues of visual decision support from five essential categories, that is, visualization, stakeholder, purpose, context and system. Some of the problems are caused by high data complexities, large data volumes, dynamic visualization paradigms, and visualization quality and aesthetics; others are incurred due to the changing facets of visualizations. Such complexities and changing nature of visualizations may significantly impair the effectiveness of a visualization in terms of how well it supports its stakeholders to solve the decisional problem of interest and achieve the intended purpose.
However, many existing visualization techniques and systems focus more on supporting particular paradigms, domains and data types, but less on addressing the complexities involved in multi-paradigm, multi-domain problems that deal with complex data and diverse representation/presentation requirements. Though they may provide reasonable support for developing purposeful effective visualizations under certain contexts, they are weak when it comes to adapting the visualizations to the changing decision problem/situational context and the subsequently varying visualization requirements. Consequently, visualizations created from such systems are often context insensitive, data dense, and sparse in intelligence. The lack of concern on the diversity/dynamics of stakeholders, purposes and contexts may result in impaired effectiveness, limited usage and even the misuse of visualizations.

To address the above problems and issues, visualization systems should allow users to flexibly create, instantiate, modify, integrate, transform and execute visualizations. Achieving such system flexibilities is the key to (1) developing purposeful effective visualizations and (2) sustaining visualization effectiveness when the underlying contexts, stakeholders and purposes change across time and domains. Motivated by this need, we conducted research on purposeful visualizations and purposeful visualization systems. Our research was guided by a multi-methodological research framework, which was instantiated from a generic framework integrating four fundamental research strategies: observation, theory building, system development, and experimentation. This framework has guided us to explore the concepts, development and application of purposeful effective visualizations and to seek answers to the following research questions:

- What are purposeful visualizations?
- How can we create purposeful visualizations?
- How can we sustain the purposefulness of a visualization under various contexts?
- How can we apply purposeful visualizations to support various tasks in decision making?

Table 9.1 provides a summary of how the above questions are addressed by our proposed research artefacts and how the artefacts are evaluated.
<table>
<thead>
<tr>
<th>Research Question</th>
<th>Research Artefacts</th>
<th>Artefact Evaluation</th>
</tr>
</thead>
</table>
| **What are purposeful visualizations?** | ▪ A formal definition of purposeful visualizations (*section 4.1*)  
▪ Purposeful visualization model (*section 4.2*) | ▪ Expert evaluation (*section 8.1*) |
| **How can we create purposeful visualizations?** | ▪ Concepts and distinguishing features of PVS (*section 5.1*)  
▪ The component oriented structural PVS framework (*subsection 5.2.1*)  
▪ The data flow oriented behavioural PVS framework (*subsection 5.2.2*)  
▪ The high level PVS architecture (*subsection 5.3.1*)  
▪ The detailed level PVS architecture (*subsection 5.3.2*)  
▪ The purposeful visualization system prototype (*chapter 7*) | ▪ Architectural analysis (*subsection 8.4.1*)  
▪ Prototyping (*subsections 8.3.1 & 8.4.2*)  
▪ Static analysis (*subsection 8.5.1*)  
▪ Functional testing (*subsection 8.5.2*)  
▪ Structural testing (*subsection 8.5.3*)  
▪ Computer simulations (*subsection 8.5.4*)  
▪ Case study (*subsections 8.3.2 & 8.5.5*)  
▪ Illustrative scenarios (*subsections 8.3.3 & 8.5.6*)  
▪ Informed argument (*subsection 8.5.7*)  
▪ Expert evaluation (*subsections 8.3.4, 8.4.3 & 8.5.8*) |
| **How can we sustain the purposefulness of a visualization under various contexts?** | ▪ Purposeful visual decision making (PVDM) process (*section 4.3*) | ▪ Prototyping (*subsection 8.2.1*)  
▪ Case study (*subsection 8.2.2*)  
▪ Illustrative scenarios (*subsection 8.2.3*)  
▪ Expert evaluation (*subsection 8.2.4*) |

Table 9.1  
Research questions, artefacts and evaluation

More specifically, to communicate the concept of purposeful visualizations, we introduced and formally defined the concept of purposeful visualizations, proposed a model to facilitate understanding, creating and evaluating purposeful visualizations. To support developing purposeful visualizations and sustaining visualization effectiveness, we designed purposeful visualization system frameworks and architectures to guide system implementations that
allow flexible creation, instantiation, manipulation, integration, execution, evaluation and modification of visualizations. We validated these concepts, models, processes, frameworks and architectures by implementing a prototypical purposeful visualization system that can be used to create purpose-driven, context-sensitive, stakeholder-relevant visualizations. To guide the application of purposeful visualizations, we proposed the PVDM process to develop and apply visualizations to address various purposes in the context of decision making. Moreover, we demonstrated the functionalities and capabilities of the purposeful visualization system prototype through a sequence of real world scenario-driven illustrations drawn from the utility sector.

To validate the proposed research artefacts of purposeful visualizations and purposeful visualization systems, we adopted a set of evaluation methods including informed argument, prototyping, case study, illustrative scenarios, static analysis, architecture analysis, computer simulations, functional testing, structural testing and expert evaluation. For each individual research artefact, one or more evaluation methods were selected based on the nature and evaluation requirements of the artefact. These research artefacts led to a series of theoretical and practical contributions to the domain of purposeful visualization of information. Our research contributions are detailed in the following section.

### 9.2 Contributions

Many difficulties associated with visual decision making arise from (1) the changing nature of visualizations (i.e. visualization stakeholders, purposes and contexts varying over time and application domains) and (2) the increasingly high complexities involved in decision problems (e.g. high data volumes, multiple combined and complex data types, combined dynamic paradigms/domains). Dealing with such complex decision problems often requires visualizations to be developed in a manner that they can be flexibly created, manipulated, modified, customized, integrated, transformed, and adapted. However, existing visualization techniques and systems tend to provide weak support for addressing these problems and issues and fulfilling the system requirements. To mitigate this gap, we introduced new concepts of purposeful visualizations and purposeful visualization systems, and proposed a series of processes, models, frameworks and architecture to realize PVS and develop purpose-
driven, context-sensitive, interactive visualizations. These artefacts have made manifold contributions to the literature and practice of information visualization. They particularly contribute to (1) supporting the development of purposeful effective visualizations under changing visualization stakeholders/purposes/contexts, (2) sustaining visualization effectiveness over time and domains, and (3) relieving the problems caused by the complexities involved in decision problems. In this section, we highlight and articulate the significant contributions of these research artefacts.

**Purposeful Visualization Definition**

Our research provides a unique theoretical contribution to the extant literature of information visualization by proposing the concept of purposeful visualizations. We formally defined purposeful visualizations, and highlighted the critical changing facets of visualizations: stakeholder, purpose, and context. The concept of PV may help researchers and practitioners to better understand visualizations especially their changing nature. Additionally, this concept reveals the challenges and complexities caused by the changing nature of visualizations, and advocates that people evaluate the effectiveness of a visualization based on its visual capability of achieving an intended purpose. It contributes to helping people to understand the key facets of visualizations that cause the varied visualization effectiveness among visualization stakeholders. Additionally, it portrays what kind of visualizations are capable of providing illuminating, useful, relevant and actionable information for stakeholders.

**Purposeful Visualization Model**

The purposeful visualization model contributes to understanding and communicating purposeful effective visualizations. It serves as a conceptual-level structure of purposeful visualizations and points out a way to create, analyze and evaluate such visualizations. It highlights the critical factors of achieving purposeful visualizations and sheds a light on the root causes of unstable visualization effectiveness. A stakeholder’s judgment on visualization effectiveness is primarily affected by two factors. The first factor is the match between the semantic layers of a visualization and the visualization profile of the stakeholder, while the second is how effectively a visualization presenter can articulate and demonstrate this match. This PV model may assist researchers and practitioners in managing the issues raised by the
changing purposes/contexts/decision-makers when they select, design and implement visualizations.

The purposeful visualization model is useful and relevant to various types of visualization stakeholders. For visualization viewers, the endeavor of aligning the semantic layers of a visualization with their subjective/objective visualization requirements may inspire them to refine their visualization purposes and requirements. For visualization designer, such alignment process may help them to identify and mitigate the gap between their interpretation of visualization requirements and the real expectation of viewers.

**Purposeful Visual Decision Making Process**

The purposeful visual decision making process contributes to the development and application of purposeful visualizations in order to support decision making under complex and changing visualization contexts. It serves as a generic approach for people to follow to leverage the power of visualization technologies to (1) support various decision making tasks and (2) deal with complexities involved in decision making. It not only outlines the key steps involved in rational visual decision making, but also highlights potential visualization context changes that may be caused by many human factors such as emotion, intuition, inspiration, imagination, memory and even personal characteristics. Such changes often lead to diverse and changing visualization requirements, which, in turn, lead to the iterations in the rational side process.

**Purposeful Visualization Systems – Definition & Distinguishing Features**

The concept of purposeful visualization systems is introduced particularly to support developing and sustaining purposeful effective visualizations. The distinguishing system features of PVS contribute to helping researchers and practitioners to understand what functionalities a visualization system should support to become a purposeful visualization system. Such functionalities are crucial to address/relieve challenges caused by the complexities associated with (1) decision problems and (2) diverse and changing visualization contexts. The functionality-level specifications can be used to form the basis for designing, developing, testing, and implementing purposeful visualization systems.
The proposed purposeful visualization frameworks and architectures can be adopted to guide the design and development of flexible visualization systems. They may support developing both standalone visualization systems and visualization subsystems embedded within other decision support systems. A significant contribution of these frameworks and architectures is that they demonstrate one possible way to separate information presentation from representation, which is the key to realize flexible information presentation in visualization systems. They also demonstrate how to achieve the system flexibilities required for creating, instantiating, modifying, integrating, transforming, and adapting visualizations.

The PVS frameworks and architectures suggest that a visualization technique can be developed by selecting and integrating a set of information generation/representation/presentation related models and solvers that are relevant and effective to address a visualization problem of interest. Purposeful visualizations can then be generated by applying the selected models and solvers to the underlying data to address the problem. Furthermore, the PVS frameworks and architectures demonstrate a flexible way to create new visualization techniques. They highlight the importance of keeping information generation, representation, and presentation models separated from each other in order to achieve improved flexibility, reusability and extensibility. Specifically, separating the generation models from the representation models allows the same data set to be visually encoded in multiple ways, while separating the presentation models from the representation models enables the same visual representations to presented in diverse fashions. By implementing loosely coupled system components, new visualization techniques can be built through selecting, modifying and integrating relevant information generation/representation/presentation models and solvers.

Purposeful Visualization System Prototype

The purposeful visualization system prototype contributes to illustrating and validating our proposed concepts, models, processes, frameworks and architectures, and proves one possible way to operationalize these artefacts. It is particularly helpful for researchers and practitioners to understand the key visualization system requirements for supporting the flexible design and development of purposeful effective visualizations. While it serves as a
proof-of-concept system, the prototypical system can be further extended to become a commercially viable system.

**The Integrated Multi-Methodological Research Approach**

As we discussed in Chapter 3, Information systems (IS) research is complex in nature. Such complexities are caused by the diversity of IS research areas, interests and artefacts, the various dimensions of problem situations and materials and the dynamic social and personal research contexts. The complexities involved in IS research require the adoption of appropriate multi-methodological approaches to provide theoretical and practical guidelines for conducting IS research. However, integrating and mapping multiple research methodologies is not an easy task to accomplish. By examining and synthesizing representative multi-methodological research approaches, we realize that each multi-methodological research framework has its own strengths and weaknesses. The integrated multi-methodological research approach, proposed and adopted by our research, contributes to mitigating the gap between the need of multi-methodological approaches and the support from the existing research frameworks. The proposed multi-methodological research approach integrates the strengths of representative multi-methodological research frameworks and remediating their deficiencies. As a demonstration of its usage, we instantiated and leveraged it to guide our research in defining, designing and developing purposeful visualizations and purposeful visualization systems.

It deserves to be noted that the above research artefacts of purposeful visualizations and purposeful visualization systems contribute to a variety of fields/disciplines due to the multi-disciplinary nature of information visualization. The theory about these artefacts has been developed and disseminated along the research journey, as illustrated in Table 9.2. By demonstrating a generic approach to visualization design, creation, customization, integration and transformation, they may benefit any information systems that require flexible designs and implementations of visualizations, such as geospatial information systems, knowledge management systems, context-aware systems, and business intelligence systems. Apart from the contributions, equally important is to understand the current limitations and future directions of our research, which are outlined in the following section.
<table>
<thead>
<tr>
<th>Publication</th>
<th>PV Definition</th>
<th>PV Model</th>
<th>PVDM Process</th>
<th>PVS Definition &amp; Features</th>
<th>PVS Frameworks</th>
<th>PVS Architectures</th>
<th>PVS Prototype</th>
<th>Research Methodology</th>
</tr>
</thead>
</table>

Table 9.2 Development and dissemination of key research artefacts
9.3 Limitations and Future Research

All research has certain limitations no matter how well it is structured, and so does our research on purposeful visualizations and purposeful visualization systems. This section highlights the limitations of our research in order to facilitate researchers and practitioners to better understand, reproduce and further extend our research. Though our research has fulfilled the research objectives listed in the introduction (section 1.5), it still has some limitations with the implementation of the purposeful visualization system prototype. More specifically, compared to other representative visualization systems (Table 8.8), the prototype provides limited support for flexibly integrating information generation models. However, it does not prevent us from demonstrating and validating the ideas of purposeful visualizations and purposeful visualization systems since our research focuses on the problems and issues with information representation and presentation. In addition, the prototype currently does not support users to develop new representation/presentation model templates. Though it does not affect the demonstration of our proposed theory, such system extensibility is particularly relevant and useful for a commercial version of the prototype.

Although it is a fully functioning system, our current purposeful visualization system prototype implements only the core distinguishing features of PVS for demonstrating the ideas of our proposed concepts, models, processes, frameworks and architectures. We will be continuing to work on extending the prototype to a production system in order to accomplish the knowledge transfer from academic research to commercial outcomes. Furthermore, the application of the prototype in a variety of context/scenarios ranging from business to art to history to engineering to science will be accomplished in our future research. More experimental and field studies of purposeful visualization systems are also envisaged.
APPENDICES

In this research, we proposed concepts, models, processes, frameworks, and architectures of purposeful visualizations and purposeful visualization systems. To validate these artefacts, we implemented a prototypical purposeful visualization system based on the proposed the system architectures. The implementation details of the essential building blocks of the prototype are presented in Appendices A-F. The focus of each appendix is highlighted in the figure below.

Appendix A: Interface Layer

The key classes supporting the interface layer of the purposeful visualization system prototype are presented in Figure A.1.
Figure A.1 Class diagram of the user interface layer
The essential code snippets implementing the above classes are shown as follows:

**Class - DesignerCanvas**

```csharp
using System;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
namespace CAVE_VS
{
    public partial class DesignerCanvas : Canvas
    {
        private Point? rubberbandSelectionStartPoint = null;
        private SelectionService selectionService;
        public SelectionService SelectionService
        {
            get
            { return selectionService; }
            set
            { selectionService = new SelectionService(this); return selectionService; }
        }
        #region SelectedItem
        public static readonly DependencyProperty SelectedItemProperty = DependencyProperty.Register("SelectedItem", typeof(ISelectable), typeof(DesignerCanvas));
        public ISelectable SelectedItem
        {
            get { return (ISelectable)GetValue(SelectedItemProperty); }
            set { SetValue(SelectedItemProperty, value); }
        }
        #endregion
        void OnMouseDown(MouseButtonEventArgs e)
        {
            base.OnMouseDown(e);
            if (e.Handled)
            {
                rubberbandSelectionStartPoint = e.GetPosition(this);
                selectionService.ClearSelection();
                Focus();
                e.Handled = true;
            }
        }
        void OnMouseMove(MouseEventArgs e)
        {
            base.OnMouseMove(e);
            if (e.Handled)
            {
                rubberbandSelectionStartPoint = e.GetPosition(this);
                selectionService.ClearSelection();
                Focus();
            }
        }
        protected override void OnDrop(DragEventArgs e)
        {
            base.OnDrop(e);
            if (e.Handled)
            {
                rubberbandSelectionStartPoint = e.GetPosition(this);
                selectionService.ClearSelection();
                Focus();
            }
        }
        protected override void OnMouseMove(MouseEventArgs e)
        {
            base.OnMouseMove(e);
            if (e.Handled)
            {
                rubberbandSelectionStartPoint = e.GetPosition(this);
                selectionService.ClearSelection();
                Focus();
            }
            else
            {
                if (dragObject != null)
                {
                    if (dragObject.Type == DragObjectType.File)
                    {
                        DragObject = null;
                        e.Data.GetData(DataFormats.FileDrop) as string[]
                        if (files != null)
                        {
                            foreach (string file in files)
                            {
                                FileDrop = file;
                            }
                        }
                    }
                    else
                    {
                        if (dragObject.Type == DragObjectType.Image)
                        {
                            DragObject = null;
                            e.Data.GetData(DataFormats.Image) as Image
                            if (image != null)
                            {
                                Image = image;
                            }
                        }
                        else
                        {
                            if (dragObject.Type == DragObjectType.Text)
                            {
                                DragObject = null;
                                e.Data.GetData(DataFormats.Text) as string
                                if (text != null)
                                {
                                    Text = text;
                                }
                            }
                        }
                    }
                }
            }
        }
        protected override void OnDrop(DragEventArgs e)
        {
            base.OnDrop(e);
            if (e.Handled)
            {
                rubberbandSelectionStartPoint = e.GetPosition(this);
                selectionService.ClearSelection();
                Focus();
            }
        }
    }
}
```

209
Class: DesignerItem

```xml
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:System="clr-namespace:System;assembly=mscorlib"
    xmlns:CAVE_VS="clr-namespace:CAVE_VS"
    xmlns:Vector="clr-namespace:Vector;assembly=Vector"
    xmlns:DesignCanvas="clr-namespace:DesignCanvas;assembly=DesignCanvas"
    xmlns:Designer="clr-namespace:Designer;assembly=Designer"
    xmlns:DesignerItem="clr-namespace:DesignerItem;assembly=DesignerItem"
    xmlns:ApplicationCommands="clr-namespace:ApplicationCommands;assembly=PresentationFramework">
    <DesignCanvas.ContextMenu x:Key="DesignerItemContextMenu">
        <DesignCanvas.MenuItem Header="Assign Context to Scenario" Command="{x:Static ApplicationCommands.AssignContextToScenario}">
            <DesignCanvas.MenuItem.Icon Source="Images/AssignScenarioContext.png" />
        </DesignCanvas.MenuItem>
        <DesignCanvas.MenuItem Header="Assign Context to IPModel" Command="{x:Static ApplicationCommands.AssignIGInput}">
            <DesignCanvas.MenuItem.Icon Source="Images/AssignIGInput.png" />
        </DesignCanvas.MenuItem>
        <DesignCanvas.MenuItem Header="Assign Context to IRModel" Command="{x:Static ApplicationCommands.AssignIRContext}">
            <DesignCanvas.MenuItem.Icon Source="Images/AssignContextToIRModel.png" />
        </DesignCanvas.MenuItem>
        <DesignCanvas.MenuItem Header="Assign Context to Scenario" Command="{x:Static ApplicationCommands.AssignScenarioContext}">
            <DesignCanvas.MenuItem.Icon Source="Images/AssignContextToScenario.png" />
        </DesignCanvas.MenuItem>
        <DesignCanvas.MenuItem Header="Copy" Command="{x:Static ApplicationCommands.Copy}"
            ImageSource="Images/Copy.png" Width="16" />
        <DesignCanvas.MenuItem Header="Paste" Command="{x:Static ApplicationCommands.Paste}"
            ImageSource="Images/Paste.png" Width="16" />
        <DesignCanvas.MenuItem Header="Delete" Command="{x:Static ApplicationCommands.Delete}"
            ImageSource="Images/Delete.png" Width="16" />
        <DesignCanvas.MenuItem Header="Group" Command="{x:Static ApplicationCommands.Group}"
            ImageSource="Images/Group.png" Width="16" />
        <DesignCanvas.MenuItem Header="Ungroup" Command="{x:Static ApplicationCommands.Ungroup}"
            ImageSource="Images/Ungroup.png" Width="16" />
        <DesignCanvas.MenuItem Header="Cut" Command="{x:Static ApplicationCommands.Cut}">
            <DesignCanvas.MenuItem.Icon Source="Images/Cut.png" />
        </DesignCanvas.MenuItem>
        <DesignCanvas.MenuItem Header="Copy" Command="{x:Static ApplicationCommands.Copy}"
            ImageSource="Images/Copy.png" Width="16" />
        <DesignCanvas.MenuItem Header="Paste" Command="{x:Static ApplicationCommands.Paste}"
            ImageSource="Images/Paste.png" Width="16" />
        <DesignCanvas.MenuItem Header="Delete" Command="{x:Static ApplicationCommands.Delete}"
            ImageSource="Images/Delete.png" Width="16" />
        <DesignCanvas.MenuItem Header="Group" Command="{x:Static ApplicationCommands.Group}"
            ImageSource="Images/Group.png" Width="16" />
        <DesignCanvas.MenuItem Header="Ungroup" Command="{x:Static ApplicationCommands.Ungroup}"
            ImageSource="Images/Ungroup.png" Width="16" />
        <DesignCanvas.MenuItem Header="Cut" Command="{x:Static ApplicationCommands.Cut}">
            <DesignCanvas.MenuItem.Icon Source="Images/Cut.png" />
        </DesignCanvas.MenuItem>
    </DesignCanvas.ContextMenu>
</ResourceDictionary>
```
using System;  
using System.Windows;  
using System.Windows.Controls;  
using System.Windows.Input;  
using System.Windows.Media;  
using System.ComponentModel;  
using System.Xml.Linq;  
namespace CAVE_VS  
{
    [TemplatePart(Name = "PART_DragThumb", Type = typeof(DragThumbTemplate))]
    [TemplatePart(Name = "PART_ResizerDecorator", Type = typeof(Control))]
    [TemplatePart(Name = "PART_ConnectorDecorator", Type = typeof(Control))]
    [TemplatePart(Name = "PART_ContentPresenter", Type = typeof(ContentPresenter))]
    public partial class DesignerItem : ContentControl, ISelectable, IGrowable  
    {
        const double DEFAULT_WIDTH = 250;  
        #region ID  
        private Guid id;  
        public Guid ID  
        {
            get { return id; }  
        }  
        #endregion  
        #region ParentID  
        public Guid ParentID  
        {
            get { return (Guid)GetValue(ParentIDProperty); }  
            set { SetValue(ParentIDProperty, value); }  
        }  
        #endregion  
        public static readonly DependencyProperty ParentIDProperty = DependencyProperty.Register("ParentID", typeof(Guid), typeof(DesignerItem));  
        #endregion  
        #region IsGroupProperty  
        public bool IsGroup  
        {
            get { return (bool)GetValue(IsGroupProperty); }  
            set { SetValue(IsGroupProperty, value); }  
        }  
        #endregion  
        public static readonly DependencyProperty IsGroupProperty = DependencyProperty.Register("IsGroup", typeof(bool), typeof(DesignerItem));  
        #endregion  
        #region IsSelectedProperty  
        public bool IsSelected  
        {
            get { return (bool)GetValue(IsSelectedProperty); }  
            set { SetValue(IsSelectedProperty, value); }  
        }  
        #endregion  
        public static readonly DependencyProperty IsSelectedProperty = DependencyProperty.Register("IsSelected", typeof(bool), typeof(DesignerItem));  
        #endregion  
        #region DragThumbTemplateProperty  
        public static readonly DependencyProperty DragThumbTemplateProperty = DependencyProperty.RegisterAttached("DragThumbTemplate", typeof(DragThumbTemplate), typeof(ControlTemplate));  
        #endregion  
        #endregion  
        #endregion  
    }
}
public static ControlTemplate
GetConnectorDecoratorTemplate(UIElement element)
{ return (ControlTemplate)element.GetValue(ConnectorDecoratorTemplatemetaProperty); }
public static void
SetConnectorDecoratorTemplate(UIElement element,
ControlTemplate value)
{ element.SetValue(ConnectorDecoratorTemplateProperty,
value); }
#region IsDragConnectionOver
public bool IsDragConnectionOver
{ get { return (bool)GetValue(IsDragConnectionOverProperty); } set { SetValue(IsDragConnectionOverProperty,
value); } }
#endregion
public static readonly DependencyProperty
IsDragConnectionOverProperty = DependencyProperty.Register("IsDragConnectionOver",
typeof(bool),
FrameworkPropertyMetadata(false));
#endregion
static DesignerItem()
{ FrameworkElement.DefaultStyleKeyProperty.OverrideMetadata(
typeof(DesignerItem), new
FrameworkPropertyMetadata(typeof(DesignerItem)));
}
public DesignerItem(Guid id)
{ this.id = id;
this.loaded = new
RoutedEventHandler(DesignerItem_loaded);
}
protected override void
OnMouseDoubleLeftMouseButtonDown(MouseButtonEventArgs e)
{ base.OnMouseLeftButtonDown(e);
try
{
DesignerCanvas parentCanvas =
VisualTreeHelper.GetParent(this) as DesignerCanvas;
// Find parent window
dependencyObject designer = this;
while (designer != null && !(designer is IScenarioDesigner))
{ designer =
VisualTreeHelper.GetParent(designer);
if (designer != null)
{ DesignerCanvas rootCanvas = (designer as IScenarioDesigner).ScenarioDesigner;
if ((Keyboard.Modifiers & ModifierKeys.Shift) != ModifierKeys.None)
if (this.IsSelected)
{ rootCanvas.SelectionService.RemoveFromSelection(this);
parentCanvas.SelectionService.RemoveFromSelection(this);
} else
{ rootCanvas.SelectionService.AddToSelection(this);
parentCanvas.SelectionService.AddToSelection(this);
} else if (this.IsSelected)
{ rootCanvas.SelectionService.SelectItem(this);
parentCanvas.SelectionService.SelectItem(this);
}
Focus();
}
} catch ()
e.Handled = false;
protected override void
OnMouseDoubleLeftMouseButtonDown(MouseButtonEventArgs e)
{ base.OnMouseLeftButtonDown(e);
if (this.Content != null && this.Content is ICavControl)
{ ICavControl component = this.Content as
ICavControl;
Viewbox contentPreview = component.Viewbox as Viewbox;
Point pt = e.GetPosition(contentPreview);
if (contentPreview is ICavControl.
Viewbox)
{ DragThumb =
(contentPreview, pt) == null)
component.ShowComponentSetting();
this.ToolTip =
component.ComponentsName;
} else
{ DependencyObject hitComponent =
(VisualTreeHelper.HitTest(contentPreview, pt)).VisualHit;
IComponentData while (hitComponent != null
&& !((hitComponent is ICavControl.
IComponentData))
hitComponent =
VisualTreeHelper.GetParent(hitComponent);
if (hitComponent != null &&
(hitComponent as
ICavControl.
IComponentData).ComponentData != null)
{ XElement contextDefinition = new
XElement((hitComponent as
ICavControl.
IComponentData).ComponentData);
Design.PropagateContext(this,
contextDefinition);
} } else
{ e.Handled = true;
private void DesignerItem_loaded(object sender,
RoutedEventArgs e)
{ if (base.Template != null)
{ ContentPresenter contentPresenter =
this.Template.FindName("PART_ComponentPresenter", this) as
ContentPresenter;
if (contentPresenter != null)
{ if (VisualTreeHelper.GetChild(contentPresenter, 0) as
UIElement)
if (contentVisual != null)
{ DragThumb =
this.Template.FindName("PART_DragThumb", this) as
DragThumb;
if (this != null)
{ ControlTemplate template =
DesignerItem.GetDragThumbTemplate(contentVisual) as
ControlTemplate;
if (template != null)
thumb.Template =
template;
}
} InitializeItemSize();
}
private void InitializeItemSize()
Appendix B: Application Layer – Visual Scenario Manager

The key classes supporting the Visual Scenario Manager at the application layer of the purposeful visualization system prototype are presented in Figure B.1.

Figure B.1 Class diagram of Visual Scenario Manager

The essential code snippets implementing the above classes are shown as follows:

**Class - VisualNarrativeDesigner**

```csharp
namespace CAVE_VN {
    /// <summary>
    /// Interaction logic for VisualNarrativeDesigner.xaml
    /// </summary>
    public partial class VisualNarrativeDesigner : Window, IScenarioDesigner {
        #region IScenarioDesigner
        public DesignerCanvas ScenarioDesigner { get { return this.DesignerCanvas; } }
        public Magnifier ScenarioMagnifier { get { return this.magnifier; } }
        #endregion
        #region Constructors
        public VisualNarrativeDesigner() { InitializeComponent(); }
        #endregion
    }
}
```

**Class: VisualScenarioDesigner**

```csharp
namespace CAVE_VS {
    public partial class CavVisualScenario : UserControl, ICavComponent, IVisual, IContext {
        #region Properties
        public string Description { get; set; }
        #endregion
    }
}
```
isValidCavComponentDefinition(string filename)
{
    bool isValid = false;
    if (filename != null)
    {
        string fileExtension = filename.Substring(filename.LastIndexOf(".")));
### Create CavComponent

Create a new CavComponent from XML definition.

```csharp
public static DesignerItem NewDesignerItem()
{
    DesignerItem designerItem = new DesignerItem()
    {
        Padding = new Thickness(2),
        BorderBrush = (SolidColorBrush)(new BrushConverter().ConvertFrom("##ff110f")),
        BorderThickness = new Thickness(1),
        HorizontalAlignment = HorizontalAlignment.Left,
        IsHitTestVisible = true
    }
    return designerItem;
}

#region Create CavComponent
// Create a CavComponent from xml definition
public static DesignerItem CavComponent(XElement componentDefinition)
{
    CavComponentType ComponentType = (CavComponentType)Enum.Parse(typeof(CavComponentType), componentDefinition.Attribute("ComponentType").Value);
    DesignerItem designerItem = Design.NewDesignerItem();
    designerItem.Type = ComponentType;
    XElement content = null;
    switch (ComponentType)
    {
    case CavComponentType.IRModel:
    content = new IRModel();
    break;
    case CavComponentType.IPModel:
    content = new IPModel();
    break;
    case CavComponentType.IGModel:
    content = new IGModel();
    break;
    case CavComponentType.VisualScenario:
    content = new VisualScenario();
    break;
    }
    if (content != null)
    {
        designerItem.Content = content;
        return designerItem;
    }
    else
    return null;
}
#endregion Create CavComponent

public static bool IsIRModel(CavComponentType type)
{
    return (type == CavComponentType.IRModel) ? true : false;
}

public static bool IsIPModel(CavComponentType type)
{
    return (type == CavComponentType.IPModel) ? true : false;
}

public static void GetConnectors(DependencyObject parent, List<Connector> connectors)
{
    int childrenCount = VisualTreeHelper.GetChildrenCount(parent);
    for (int i = 0; i < childrenCount; i++)
    {
        DependencyObject child = VisualTreeHelper.GetChild(parent, i);
        if (child is Connector) connectors.Add(child as Connector);
    }
}

// Propagate problem/user contexts when user double-clicks on an IRModel component.
public static void PropagateContext(DesignerItem item, XElement ContextDefinition)
{
    // For propagating context to connectedItem.sink.
    Control connectorDecorator = item.Template.FindName("PART_ConnectorDecorator", item) as Control;
    List<Connector> connectors = new List<Connector>();
    Design.GetConnectors(connectorDecorator, connectors);
    foreach (Connector connector in connectors)
    foreach (Connection conn in connector.Connections)
    {
        if (conn.Source == connector)
        {
            DesignerItem linkedItem = conn.Sink.ParentDesignerItem;
            if (linkedItem.Content is IContext)
            {
                linkedItem.Content as IContext).ApplyContext(ContextDefinition);
            }
        }
    }
}

// For propagating context to linked items other than the direct children, use default component context
if (Design.HasLinkedItems(linkedItem))
{
    // Find the first IComponentData as the default context
    DefaultComponentData = null;
    if (linkedItem.Content is ICavComponent)
    {
        ICavComponent DefaultComponentData = Representation.ComponentData((linkedItem.Content as ICavComponent).Output);
        if (DefaultComponentData != null)
        Design.PropagateContext(linkedItem, DefaultComponentData);
    }
```
Appendix C: Application Layer – Information Generation

The Information Generation Subsystem contains two main components: Data Source Manager, and IG Manager. Sections C.1 and C.2 demonstrates how these system components are implemented respectively.

C.1 Data Source Manager

The key class supporting the Data Source Manager at the application layer of the purposeful visualization system prototype is presented in Figure C.1.

![Class diagram of Data Source Manager](image-url)

Figure C.1  Class diagram of Data Source Manager
The essential cope snippets implementing the above class are shown as follows:
//Create DS directory
string DSDirectory = System.IO.Path.Combine(directoryName, "Data Source");
if (!System.IO.Directory.Exists(DSDirectory))
    System.IO.Directory.CreateDirectory(DSDirectory);

form = new ModelEditorForm()
    { WorkFolder = DSDirectory,
      BackColor = System.Drawing.Color.WhiteSmoke
    };

stream = assembly.GetManifestResourceStream("CAVE_VS.Resources.Images.DataSet.png");
form.Icon = new System.Drawing.Icon(stream);

public void New()
    { form.RunNewModelWizardOnStart = true;
      ((Form)form).ShowDialog();
    }

public void Edit(string modelDefinitionFile)
    { form.RunNewModelWizardOnStart = false;
      form.LoadModelFromFile(modelDefinitionFile);
      ((Form)form).ShowDialog();
    }

C.2 Information Generation Manager

The key class supporting the Information Generation Manager at the application layer of the purposeful visualization system prototype is presented in Figure C.2.

![Class diagram of Information Generation Manager](image)

The essential cope snippets implementing the above class are shown as follows:
namespace CAV_VS
{

    public partial class CavIGModel : UserControl,
    ICavComponent, IVisual
    {
        #region Properties
        public string IGModelQuery { get; set; }
        #endregion Properties

        #region ICavComponent
        public string ComponentName
        {
            get { return lblIGModelName.Text; }
            set { lblIGModelName.Text = value; }
        }
        #endregion ICavComponent

        public void ShowComponentSetting()
        {
            // If an IGModel has an assigned data source,
            // it will be loaded the IG Model input.
            if (this.Input != null)
            {
                IG_ModelDesigner modelDesigner = new
                IG_ModelDesigner(this.Input);
                modelDesigner.IGModelName = this componentName;
                modelDesigner.IGModelQuery = this.IGModelQuery =
                modelDesigner.IGModelQuery.LoadFromXmlReader(IGModelDefinition.Elem
                "DefinitionFile".Value);
                this.VisualPresentation();
                this.ShowDetailView();
                this.vbPreview.Child =
                this.VisualPresentation();
            }
            modelDesigner.Show();
        }
        #endregion Methods
    }
}
Appendix D: Application Layer – Information Representation

The key classes supporting the Information Representation Manager at the application layer of the purposeful visualization system prototype are presented in Figure D.1.

The essential cope snippets implementing the above classes are shown as follows:

**Class - IR_ModelDesigner**

```xml
<Window x:Class="CAVE_VS_IR_ModelDesigner"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    mc:Ignorable="d"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    WindowStartupLocation="CenterScreen"
    Title="IR Model Designer"
    SizeChanged="IR_ModelDesigner_SizeChanged">
    <Grid RowDefinitions="Auto" ToolBar Grid.Row="0">
        <ToolBar Grid.Row="0">"} {RelativeSource Images\InfoRepresentation.png"
        Source="../Resources/Images/InfoGeneration.png"
        Width="167" />
    </Grid>
    <StackPanel Orientation="Horizontal">
        <TextBox x:Name="txtInputModel" Width="150" Foreground="{(StaticResource ToolbarTextBrush)"
            VerticalContentAlignment="Center" TextWrapping="Wrap" IsReadOnly="True" />
        <Button x:Name="btnSelectInputModel" Margin="1" Padding="2" Style="{(StaticResource ToolbarButtonTextBrush)"
            Click="btnSelectInputModel_Click">
            Source="../Resources/Images/InfoSelect.png"
            Width="167" />
    </StackPanel>
</Grid>
```

**Class - IR_ModelParameter**

```xml
<Window x:Class="CAVE_VS_IR_ModelDesigner"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    mc:Ignorable="d"
    DataContext="{Binding RelativeSource={RelativeSource Self}}"
    WindowStartupLocation="CenterScreen"
    Title="IR Model Designer"
    SizeChanged="IR_ModelDesigner_SizeChanged">
    <Grid RowDefinitions="Auto" ToolBar Grid.Row="0">
        <ToolBar Grid.Row="0">"} {RelativeSource Images\InfoRepresentation.png"
        Source="../Resources/Images/InfoGeneration.png"
        Width="167" />
    </Grid>
    <StackPanel Orientation="Horizontal">
        <TextBox x:Name="txtInputModel" Width="150" Foreground="{(StaticResource ToolbarTextBrush)"
            VerticalContentAlignment="Center" TextWrapping="Wrap" IsReadOnly="True" />
        <Button x:Name="btnSelectInputModel" Margin="1" Padding="2" Style="{(StaticResource ToolbarButtonTextBrush)"
            Click="btnSelectInputModel_Click">
            Source="../Resources/Images/InfoSelect.png"
            Width="167" />
    </StackPanel>
</Grid>
```
public partial class IR_ModelDesigner : Window
{
    #region Properties
    public static readonly DependencyProperty ModelParametersProperty = DependencyProperty.Register("ModelParameters",
        typeof(ObservableCollection<IR_ModelParameter>)),
        new PropertyMetadata(new ObservableCollection<IR_ModelParameter>()));
    public ObservableCollection<IR_ModelParameter> ModelParameters
    {
        get { return (ObservableCollection<IR_ModelParameter>)GetValue(ModelParametersProperty); } 
        set { SetValue(ModelParametersProperty, value); } 
    }
    #endregion

    #region Events
    public XAttribute IGModelDefinition { set; get; }
    public XElement Parameters { set; get; }
    public Context ModelContext { get; set; }
    #endregion

    #region Constructors
    public IR_ModelDesigner(XElement Input)
    { InitializeComponent();
        this.IGModelDefinition = Input;
        txtInputModel.Text = IGModelDefinition.Attribute("Name").Value;
    }
    public IR_ModelDesigner() { InitializeComponent(); }
    #endregion

    #region Methods
    // Use IGModel as input
    private void btnExecuteModel_Click(object sender, RoutedEventArgs e)
    {
        try
        { vbPreview.ChildId = xml;
            catch(System.Exception ex)
            { MessageBox.Show(ex.Message); }
        }
        private void btnSaveModel_Click(object sender, RoutedEventArgs e)
        {
            saveFileDialog.AddExtension = true;
            saveFileDialog.DefaultExt = "xml";
            saveFileDialog.Filter = "XML files (*.xml)|*.xml";
            if (dlg.ShowDialog() == true)
            { XElement IRDefinition = XElement.Load(dlg.FileName);
                this.txtInputModel.Text = IRDefinition.Element("Name").Value;
                this.txtModelDescription.Text = IRDefinition.Element("Description").Value;
                this.IGModelDefinition = IRDefinition.Element("IGModel");
                // Initiate model parameters
                IRModelType = IRModelDefinition.IsIGModelRequired(this.IRModelType) &&
                this.ModelContext.ContextDefinition != null
                ? this.ModelContext.ContextDefinition
                : (ModelParameters is null
                ? new ObservableCollection<IR_ModelParameter>()
                : new ObservableCollection<IR_ModelParameter>(pars));
            }
        }
        private void btnLoadModel_Click(object sender, RoutedEventArgs e)
        {
            try
            { double dblRatio = (e.PreviousSize.Width > 0 ? e.PreviousSize.Width : 1);
                vbPreview.ActualWidth *= dblRatio;
                vbPreview.ActualHeight *= dblRatio;
            e.Handled = true;
        }
        private void btnClearModel_Click(object sender, RoutedEventArgs e)
        {
            foreach (IR_ModelParameter par in this.ModelParameters)
            { par.Clear();
                vbPreview.ChildId = null;
            }
            XElement xml = new XElement("IModel", new XAttribute("Name",
                txtModelName.Text),
                new XAttribute("ComponentType", "IModel"),
                new XAttribute("IModelType", this.IRModelType.ToString()),
                new XElement("Description", txtModelDescription.Text));
XElement definition = XElement.Load(dlg.FileName);
if (Representation.HasValue(definition))
 {
    this.IGModelDefinition = Representation.GetInput(definition);
    this.txtInputModel.Text = this.IGModelDefinition.Attribute("Name").Value;
    foreach (IR_ModelParameter par in this.ModelParameters)
    {
        par.SetParameterName(par.InputStyle == ModelParameterInputStyle.ParameterInput)
        {
            par.SetParameterValue(null);
        }
        else
        {
            MessageBox.Show("The selected input does not have a valid IGModel to use.");
        }
    }
    catch
    {
    }
}

Class - IR_ModelParameter

using System.Windows.Controls;
using System.Collections.Generic;
using System.Linq; using System;
namespace CAVE_VS
{
    /// Interaction logic for IR_ModelParameter.xaml
    /// IR model parameter types:
    /// ParameterInput, /// TextInput, /// BooleanInput, /// ColorInput, /// BoundTextInput,
    /// FileInput
    public partial class IR_ModelParameter : UserControl, IModelParameter
    {
        public partial class IR_ModelParameter : UserControl, IModelParameter
        {
            #region Properties
            public IR_ModelParameterInputStyle InputStyle { get; set; }
            public bool IsMandatory { get; set; }
            public List<string> ParameterValueCollection
            {
                get
                { return new List<string> { new XElement("ParameterValue", this.ParameterName), new XElement("ParameterAlias", this.ParameterAlias), new XElement("ParameterType", this.ParameterType), new XElement("ParameterInputStyle", this.ParameterInputStyle), XElement new XElement(definition).Value;
        }
    }
    }

    #endregion Properties

    #region Constructor
    public IR_ModelParameter(TextModelParameterInputStyle ParameterInputStyle, XElement IGModelDefinition)
    {
        //Set parameter value input style
        this.InputStyle = ParameterInputStyle;
        InitializeComponent();
        if (IGModelDefinition != null)
            SetDataContext(IGModelDefinition);
    }
    #endregion Constructor

    // Data context comes from the connected IGModel
    public void SetDataContext(XElement IGModelDefinition)
    {
ContextMenu contextMenu = new ContextMenu()
{
    Background = (SolidColorBrush)(new BrushConverter().ConvertFrom("#FFC1D9F1")),
    Padding = new Thickness(2)
};
foreach (XElement par in IGModelDefinition.Element("Parameters").Elements("Parameters"))
{
    MenuItem item = new MenuItem()
    {
        Header = par.Element("ParameterAlias").Value,
        Foreground = (SolidColorBrush)(new BrushConverter().ConvertFrom("#FF1542D8")),
        Background = new SolidColorBrush(Colors.Transparent),
        FontSize = 10,
        Margin = new Thickness(1),
        ToolTip = "Parameter Name: " + par.Element("ParameterName").Value + " Data Type: " + par.Element("ParameterType").Value,
        Tag = new XElement(par)
    };
    item.Click += (s, a) =>
    {
        XElement p = (s as MenuItem).Tag as XElement;
        this.btnParameterValue.ContextMenu = p.Element("ParameterName").Value;
        this.btnParameterValue.ToolTip = "Parameter Name: " + p.Element("ParameterName").Value + " Data Type: " + p.Element("ParameterType").Value;
        this.btnParameterValue.ContextMenu.Items.Add(contextMenuItem);
    };
    contextMenu.Items.Add(item);
    contextMenu.ContextMenu = contextMenu;
} // Parameter context comes from the parent VisualScenario
public void SetParameterContext(XElement ContextDefinition)
{
    ContextMenu contextMenu = new ContextMenu()
    {
        Background = (SolidColorBrush)(new BrushConverter().ConvertFrom("#fff11d0f")),
        Padding = new Thickness(2)
    };
    foreach (XElement context in ContextDefinition.Elements("Context"))
    {
        MenuItem contextMenuItem = new MenuItem()
        {
            Header = context.Attribute("ContextID").Value,
            Foreground = new SolidColorBrush(Colors.Gray),
            Background = new SolidColorBrush(Colors.Transparent),
            FontSize = FontWeights.Bold,
            FontSize = 10,
            Margin = new Thickness(1),
            Tag = new XElement(context)
        };
        foreach (XElement par in context.Element("Parameters").Elements("Parameter"))
        {
            MenuItem item = new MenuItem()
            {
                Header = par.Element("ParameterAlias").Value + " " + par.Element("ParameterValue").Value,
                Foreground = (SolidColorBrush)(new BrushConverter().ConvertFrom("#FF1542D8")),
                Background = new SolidColorBrush(Colors.Transparent),
                FontSize = 10,
                Margin = new Thickness(1),
                ToolTip = par.Element("ParameterValue").Value,
                Tag = new XElement(par)
            };
            item.Click += (s, a) =>
            {
                string contextID = "((s as MenuItem).Parent as MenuItem).Tag as XElement).Attribute("ContextID").Value;";
                XElement p = (s as MenuItem).Tag as XElement;
                this.btnParameterContext.Content = " + p.Element("ParameterValue").Value;
                this.btnParameterContext.ToolTip = "Parameter Name: " + p.Element("ParameterName").Value + " Data Type: " + p.Element("ParameterType").Value;
                this.btnParameterContext.ContextMenu = new XElement("ParameterContext");
                new XAttribute("ContextID", contextID);
                new XElement(p.Element("ParameterName"));
                new XElement(p.Element("ParameterType"));
                new XElement(p.Element("ParameterInputStyle"));
                contextMenu.ContextMenu = contextMenu;
                this.btnParameterContext.ContextMenu = contextMenu;
                this.btnParameterContext.Visiblility = Visibility.Visible;
                this.btnParameterContext.Content = "Click here to create a mapping to Context";
            };
            private void IR_ModelParameter_Loaded(object sender, RoutedEventArgs e)
            {
                if (this.InputStyle == ModelParameterInputStyle.ParameterInput)
                {
                    this.btnParameterValue.ContextMenu = contextMenu;
                    this.btnParameterValue.Content = "Click here to create a mapping to IGModel": this.ParameterValue;
                    this.btnParameterValue.Click += (s, a) => { this.btnParameterValue.ContextMenu.IsOpen = true; };
                    this.btnParameterValue.Visiblility = Visibility.Visible;
                    if (this.InputStyle == ModelParameterInputStyle.TextInput)
                    {
                        this.txtParameterValue.Watermark = this.ParameterValue;
                        this.txtParameterValue.TextChanged += (s, a) => { this.txtParameterValue.Text = null; };
                        this.txtParameterValue.Visiblility = Visibility.Visible;
                    } else if (this.InputStyle == ModelParameterInputStyle.ImageInput)
                    {
                        this.btnSelectFile.Visiblility = Visibility.Visible;
                        this.txtParameterValue.Watermark = this.ParameterValue;
                        this.txtParameterValue.Visiblility = Visibility.Visible;
                        this.txtParameterValue.Text = null;
                    } else if (this.InputStyle == ModelParameterInputStyle.MediaInput)
                    {
                        openFileDlg = new OpenFileDialog()
                        openFileDlg.Filter = "*.*|PNG files|*.jpg|PNG files|*.png|GIF files|*.gif";
                        openFileDlg.Filter = "*.jpg|PNG files|*.png|GIF files|*.gif";
                        if (result == true)
                        {
                            this.txtParameterValue.Text = openFileDlg.FileName;
                        }
                    } else if (this.InputStyle == ModelParameterInputStyle.MediaInput)
                    {
                        openFileDlg = new OpenFileDialog()
                        openFileDlg.Filter = "*.jpg|PNG files|*.png|GIF files|*.gif";
                        if (result == true)
                        {
                            this.txtParameterValue.Text = openFileDlg.FileName;
                        }
                    }
                }
            }
        }
    }
}
this.btnSelectFile.Visibility =
Visibility.Visible;
this.txtParameterValue.Visibility =
Visibility.Visible;
this.txtParameterValue.Watermark =
this.ParameterValue;
this.txtParameterValue.TextChanged += (s,
a) => { this.ParameterValue =
this.txtParameterValue.Text; };
this.btnSelectFile.Click += (s, a) =>
{
Microsoft.Win32.OpenFileDialog
openFileDlg = new Microsoft.Win32.OpenFileDialog();
openFileDlg.Filter =
"WMV|*.wmv|MPG|*.mpg|MPEG|*.mpeg|M1V|*.m1v|M2V|*.m2v|MPA|*
.mpa|MPE|*.mpe|AVI|*.avi";
System.Nullable<bool> result =
openFileDlg.ShowDialog();
if (result == true)
this.txtParameterValue.Text =
openFileDlg.FileName;
};
}
else if (this.InputStyle ==
ModelParameterInputStyle.BooleanInput)
{
this.chkParameterValue.Checked += (s, a)
=> { this.ParameterValue =
Representation.BoolToString((bool)chkParameterValue.IsChec
ked); };
this.chkParameterValue.Unchecked += (s, a)
=> { this.ParameterValue =
Representation.BoolToString((bool)chkParameterValue.IsChec
ked); };
this.chkParameterValue.Visibility =
Visibility.Visible;
}
else if (this.InputStyle ==
ModelParameterInputStyle.ColorInput)
{
this.cpParameterValue.SelectedColor =
Representation.HexToColor(this.ParameterValue);
this.cpParameterValue.SelectedColorChanged
+= (s, a) => { this.ParameterValue =
Representation.ColorToHex(cpParameterValue.SelectedColor,
true); };
this.cpParameterValue.Visibility =
Visibility.Visible;
}
else if (this.InputStyle ==
ModelParameterInputStyle.BoundTextInput)
{
this.cboParameterValue.SelectionChanged +=
(s, a) => { this.ParameterValue =
(string)(cboParameterValue.SelectedItem); };
this.cboParameterValue.Visibility =
Visibility.Visible;
}
// Turn on mandatory parameter indicators
imgMandatoryInput.Visibility =
(this.IsMandatory) ? Visibility.Visible :
Visibility.Hidden;
this.btnParameterContext.Click += (s, a) =>
{ this.btnParameterContext.ContextMenu.IsOpen = true; };
}
public void SetParameterValue(string
parameterValue)
{
switch (this.InputStyle)
{
case
ModelParameterInputStyle.ParameterInput:
this.btnParameterValue.Content =
(String.IsNullOrEmpty(parameterValue)) ? "Click here to
create a mapping" : parameterValue;
break;
case ModelParameterInputStyle.TextInput:
this.txtParameterValue.Text =
parameterValue;
break;
case ModelParameterInputStyle.ImageInput:
this.txtParameterValue.Text =
parameterValue;
break;
case ModelParameterInputStyle.MediaInput:
this.txtParameterValue.Text =
parameterValue;
break;
case ModelParameterInputStyle.ColorInput:

this.cpParameterValue.SelectedColor =
Representation.HexToColor(parameterValue);
break;
case
ModelParameterInputStyle.BooleanInput:
this.chkParameterValue.IsChecked =
(parameterValue == "Yes") ? true : false;
break;
case
ModelParameterInputStyle.BoundTextInput:
if
(String.IsNullOrEmpty(parameterValue))
this.cboParameterValue.SelectedIndex = -1;
else
this.cboParameterValue.SelectedItem = parameterValue;
break;
}
this.ParameterValue = parameterValue;
}
public void SetParameterContextValue(XElement
parameterContext)
{
if (parameterContext != null)
{
this.ParameterContext = new
XElement(parameterContext);
this.btnParameterContext.Content = "[ " +
parameterContext.Attribute("ContextID").Value + " ]." +
parameterContext.Value;
this.btnParameterContext.Tag =
parameterContext.Element("ParameterInputStyle").Value;
}
else
{
this.ParameterContext = null;
this.btnParameterContext.Content = "Click
here to create a mapping to Context";
this.btnParameterContext.Tag = null;
}
}
// Clear parameter value
public void Clear()
{
switch (this.InputStyle)
{
case
ModelParameterInputStyle.ParameterInput:
this.btnParameterValue.Content =
"Click here to create a mapping";
this.btnParameterValue.ToolTip = null;
break;
case ModelParameterInputStyle.TextInput:
this.txtParameterValue.Clear();
break;
case ModelParameterInputStyle.ImageInput:
this.txtParameterValue.Clear();
break;
case ModelParameterInputStyle.MediaInput:
this.txtParameterValue.Clear();
break;
case ModelParameterInputStyle.ColorInput:
this.cpParameterValue.SelectedColor =
Representation.HexToColor("");
break;
case
ModelParameterInputStyle.BooleanInput:
this.chkParameterValue.IsChecked =
false;
break;
case
ModelParameterInputStyle.BoundTextInput:
this.cboParameterValue.SelectedIndex =
-1;
break;
}
this.ParameterValue = null;
if (this.btnParameterContext.Visibility ==
{
this.btnParameterContext.Content = "Click
here to create a mapping to Context";
this.btnParameterContext.Tag = null;
}
}
#endregion Methods
}
}

225


Appendix E: Application Layer – Information Presentation

The key classes supporting the Information Presentation Manager at the application layer of the purposeful visualization system prototype are presented in Figure E.1.

![Figure E.1 Class diagram of Information Presentation Manager](image)

The essential code snippets implementing the above classes are shown as follows:

**Class - IP_ModelDesigner_TypeI**

```csharp
using System.Collections.Generic;
using System.Windows;
using System.Xml.Linq;
using System.Linq;
using System.Collections.ObjectModel;
using System;
using System.Collections.Generic;
using Xceed.Wpf.Toolkit;
namespace CAVE_VS {
    /// <summary>
    /// Interaction logic for IP_ModelDesigner_TypeI.xaml
    /// IPModelDesigner for IP_DeepZoom model
    /// </summary>
    public partial class IP_ModelDesigner_TypeI : Window,
        IScenarioDesigner, IContext {
        XElement inputModels;
        #region Properties
        public string IPModelName {
            get { return txtModelName.Text; }
            set { txtModelName.Text = value; }
        }
        public string IPModelDescription {
            get { return txtModelDescription.Text; }
            set { txtModelDescription.Text = value; }
        }
        public CavComponentType IPModelType {
            get; set; }
        #endregion
```

---

226
```csharp
#region Methods
public void InitializeInput(XElement InputModels) {
    InputModels = InputModels;
    ObservableCollection<DesignerItem> input = new ObservableCollection<DesignerItem>(){
        foreach (XElement item in InputModels.Element("[IModels]").Elements("[IModel]"))
            item.Add(GetInputModel(Definition, CavComponentType.IRModel));
    }
    foreach (XElement item in InputModels.Element("[IPModels]").Elements("[IPModel]"))
        item.Add(GetInputModel(Definition, CavComponentType.IPMODEL));
}
private ToolboxItem GetInputModel(XElement ModelDefinition, CavComponentType ComponentType) {
    ToolboxItem item = new ToolboxItem();
    if (ComponentType == CavComponentType.IPMODEL) item.Content = new CavIRModel()
    CAVIRModel ModelDefinition = (CavComponentType == ComponentType ?
        CavIRModel(ModelDefinition) : new CavIRModel());
    item.Content = new CavIRModel();
    item.Tag = new XElement(ModelDefinition); return item;
}
public void RenderVisualOutput(XElement IPDefinition) {
    this.IPModelName = IPDefinition.Attribute("[Name]").Value;
    this.IPModelDescription = IPDefinition.Element("[Description]").Value;
    this.IPModelType = (CavComponentType)Enum.Parse(typeof(CavComponentType),
        IPDefinition.Attribute("[IPModelType]").Value);
    if (this.IPModelType == CavComponentType.IP_DeepZoom)
        IP_DeepZoom.RenderVisualOutput(IPDefinition, this.DesignerCanvas);
    }
    this.IPModelName, new XAttribute("[Name]", "IPModel"),
        new XAttribute("[ComponentType]", "IPModel"),
            new XAttribute("[IPModelType]", "IPModelType"),
                new XElement("[Description]",
                    v => (item.Content
                        .DesignerCanvas.Children.OfType<
                            IXMLElement(ite *
                                XElement("[X]", new XElement("[Y]", Y)
                            )
                        )
                    xml.Element("[IModels]").Add(Definition); // Child IPModels
                    items = this.DesignerCanvas.Children.OfType<DesignerItem>().Where( item => (item.Content is CavIRModel))
                        if (items.Count() > 0) foreach (DesignerItem item in items)
                            XElement IPDefinition = new XElement((item.Content as ICavComponent).Output);
                            if (IPDefinition.Elements("[Presentation]").Any())
                                IPDefinition.Element("[Presentation]").Remove();
                                DesignerCanvas.GetLeft(item)
                                    double X =
                                DesignerCanvas.GetTop(item)
                                    IPDefinition.Add(new XElement("[Presentation]", new XElement("[X]", new XElement("[Y]", Y)
                                )
                            } xml.Element("[IPModels]").Add(IPDefinition); }
                            return xml;
                        } return null;
                    }
    private void btnSaveModel_Click(object sender, RoutedEventArgs e) {
        saveFileDialog.DefaultExt = "xml";
        saveFileDialog.Filter = "XML files (*.xml)|*.xml"
            System Nullable<int> result = saveFileDialog.ShowDialog();
                if (result == true)
                    (XML().Save(saveFileDialog.FileName); }
        private void btnClearModel_Click(object sender, RoutedEventArgs e) {
            this.DesignerCanvas.Children.Clear();
        }
        private void btnLoadModel_Click(object sender, RoutedEventArgs e) {
                openFileDialog.Filter = "XML files (*.xml)|*.xml";
                    Nullable<int> result = openFileDialog.ShowDialog();
                        if (result == true)
                            XElement IPDefinition = XElement.Load(openFileDialog.FileName);
                                if (IPDefinition.Attribute("[ComponentType]").Value == "IPModel")
                                    RenderVisualOutput(IPDefinition);
                                        private bool IsInstantiated() {
                                            bool isInstantiated = false;
                                            if (this.DesignerCanvas.Children.OfType<DesignerItem>().Any(i item => (item.Content is CavIRModel) || item.Content is CavIPModel))
                                                isInstantiated = true;
                                                foreach (DesignerItem item in this.DesignerCanvas.Children.OfType<DesignerItem>())
```
Class - IP_ModelDesigner_TypeII

using System.Windows;
using System.Xml.Linq;
using System.Linq;
using System.Collections.ObjectModel;
using System.Windows.Controls;
namespace CAVE_V5
{
    /// Interaction logic for IP_ModelDesigner_TypeII.xaml
    /// IPModelDesigner for IPModels except IP_DeepZoom model
    ///
    /// Assign context
    ///
    /// public partial class IP_ModelDesigner_TypeII : Window, IContext
    ///
    /// ObservableCollection<ListBoxItem> Components;
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #region Properties
    /// public string IPModelName
    /// {
    ///     get { return txtModelName.Text; }
    ///     set { txtModelName.Text = value; }
    /// }
    ///
    /// public string IPModelDescription
    /// {
    ///     get { return txtModelDescription.Text; }
    ///     set { txtModelDescription.Text = value; }
    /// }
    ///
    /// public CavComponentType IPModelType { get; set; }
    /// #endregion
    ///
    /// #region Constructors
    /// public IP_ModelDesigner_TypeII() { InitializeComponent(); }
    /// #endregion
    ///
    /// #region Methods
    /// public void InitializeInput(XElement InputDefinition)
    /// {
    ///     Components = new ObservableCollection<ListBoxItem>();
    ///     PresentationSettings = new ObservableCollection<ListBoxItem>();
    ///     foreach (XElement definition in InputDefinition.Elements("IPModels"))
    ///     {
    ///         Components.Add(GetComponent(definition));
    ///     }
    ///     foreach (XElement presentationSetting in InputDefinition.Elements("IPDefinition"))
    ///     {
    ///         PresentationSettings.Add(GetPresentationSetting(presentationSetting));
    ///     }
    /// }
    /// return instantiated;
    /// #endregion
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #endregion
    ///
    /// public partial class IP_ModelDesigner_TypeII : Window, IContext
    ///
    /// ObservableCollection<ListBoxItem> Components;
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #region Properties
    /// public string IPModelName
    /// {
    ///     get { return txtModelName.Text; }
    ///     set { txtModelName.Text = value; }
    /// }
    ///
    /// public string IPModelDescription
    /// {
    ///     get { return txtModelDescription.Text; }
    ///     set { txtModelDescription.Text = value; }
    /// }
    ///
    /// public CavComponentType IPModelType { get; set; }
    /// #endregion
    ///
    /// #region Constructors
    /// public IP_ModelDesigner_TypeII() { InitializeComponent(); }
    /// #endregion
    ///
    /// #region Methods
    /// public void InitializeInput(XElement InputDefinition)
    /// {
    ///     Components = new ObservableCollection<ListBoxItem>();
    ///     PresentationSettings = new ObservableCollection<ListBoxItem>();
    ///     foreach (XElement definition in InputDefinition.Elements("IPModels"))
    ///     {
    ///         Components.Add(GetComponent(definition));
    ///     }
    ///     foreach (XElement presentationSetting in InputDefinition.Elements("IPDefinition"))
    ///     {
    ///         PresentationSettings.Add(GetPresentationSetting(presentationSetting));
    ///     }
    /// }
    /// return instantiated;
    /// #endregion
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #endregion
    ///
    /// public partial class IP_ModelDesigner_TypeII : Window, IContext
    ///
    /// ObservableCollection<ListBoxItem> Components;
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #region Properties
    /// public string IPModelName
    /// {
    ///     get { return txtModelName.Text; }
    ///     set { txtModelName.Text = value; }
    /// }
    ///
    /// public string IPModelDescription
    /// {
    ///     get { return txtModelDescription.Text; }
    ///     set { txtModelDescription.Text = value; }
    /// }
    ///
    /// public CavComponentType IPModelType { get; set; }
    /// #endregion
    ///
    /// #region Constructors
    /// public IP_ModelDesigner_TypeII() { InitializeComponent(); }
    /// #endregion
    ///
    /// #region Methods
    /// public void InitializeInput(XElement InputDefinition)
    /// {
    ///     Components = new ObservableCollection<ListBoxItem>();
    ///     PresentationSettings = new ObservableCollection<ListBoxItem>();
    ///     foreach (XElement definition in InputDefinition.Elements("IPModels"))
    ///     {
    ///         Components.Add(GetComponent(definition));
    ///     }
    ///     foreach (XElement presentationSetting in InputDefinition.Elements("IPDefinition"))
    ///     {
    ///         PresentationSettings.Add(GetPresentationSetting(presentationSetting));
    ///     }
    /// }
    /// return instantiated;
    /// #endregion
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #endregion
    ///
    /// public partial class IP_ModelDesigner_TypeII : Window, IContext
    ///
    /// ObservableCollection<ListBoxItem> Components;
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
    /// #region Properties
    /// public string IPModelName
    /// {
    ///     get { return txtModelName.Text; }
    ///     set { txtModelName.Text = value; }
    /// }
    ///
    /// public string IPModelDescription
    /// {
    ///     get { return txtModelDescription.Text; }
    ///     set { txtModelDescription.Text = value; }
    /// }
    ///
    /// public CavComponentType IPModelType { get; set; }
    /// #endregion
    ///
    /// #region Constructors
    /// public IP_ModelDesigner_TypeII() { InitializeComponent(); }
    /// #endregion
    ///
    /// #region Methods
    /// public void InitializeInput(XElement InputDefinition)
    /// {
    ///     Components = new ObservableCollection<ListBoxItem>();
    ///     PresentationSettings = new ObservableCollection<ListBoxItem>();
    ///     foreach (XElement definition in InputDefinition.Elements("IPModels"))
    ///     {
    ///         Components.Add(GetComponent(definition));
    ///     }
    ///     foreach (XElement presentationSetting in InputDefinition.Elements("IPDefinition"))
    ///     {
    ///         PresentationSettings.Add(GetPresentationSetting(presentationSetting));
    ///     }
    /// }
    /// return instantiated;
    /// #endregion
    ///
    /// #region IContext
    /// public Context Context { get; set; }
    /// #endregion
    ///
private ItemsControl PresentationSetting(XElement presentationDefinition)
{
    ItemsControl items = new ItemsControl();
    ObservableCollection<IP_ModelParameter> parameters = null;
    switch (IPModelType)
    {
        case CavComponentType.IP_SemanticZoom:
            parameters = IP_SemanticZoom.Parameters();
            break;
        case CavComponentType.IP_SpatialSemanticZoom:
            parameters = IP_SpatialSemanticZoom.Parameters();
            break;
        case CavComponentType.IP_Lifecycle:
            parameters = IP_Lifecycle.Parameters();
            break;
    }
    if (parameters != null)
    {
        items.ItemsSource = parameters;
    }
    foreach (IP_ModelParameter par in items.Items)
    {
        XElement parameterValue = Presentation.GetParameterValue(presentationDefinition, par.ParameterName);
        par.SetParameterValue(parameterValue);
    }
    return items;
}

private XElement PresentationDefinition(ItemsControl presentationControl)
{
    XElement definition = new XElement("Presentation");
    foreach (var item in presentationControl.Items)
    {
        if (item is IP_ModelParameter)
        {
            IP_ModelParameter par = (IP_ModelParameter)item;
            definition.Add(par.ParameterSetting());
        }
    }
    return definition;
}

public XElement XML()
{
    if (IsInstantiated())
    {
        XElement xml = new XElement("IPModel",
            new XAttribute("IPModelName", this.IPModelName),
            new XAttribute("ComponentType", IPModelType.ToString()),
            this.IPModelDescription);
    }
    XElement context = null && this.Context.ContextDefinition != null)
    { xml.Add(this.Context.ContextDefinition); }
    XElement xml = new XElement("IPModels");
    foreach (int i = 0; i < lstCavComponents.Items.Count; i++)
    { xml.Add(lstCavComponents.Items[i] as ListBoxItem).Tag as XElement; }
    XElement definition = (CavComponentType)Enum.Parse(typeof(CavComponentType), Definition.Attribute("ComponentType").Value);
    if (Definition.Elements("Presentation").Any())
    { definition.Add(presentationSetting); }
    XElement presentationSetting = PresentationDefinition(lstPresentationSettings.Items[1] as ItemsControl.Content as ItemsControl).Add(presentationSetting);
using System.Diagnostics;  
using System.Collections.Generic;  
using System.Linq;  
using System;  
namespace CAVE_VS  
{
    /// <summary>
    /// Interaction logic for IP_ModelParameter.xaml
    /// IP model parameter types:
    /// TextInput,  
    /// BoundTextInput,  
    /// ColorInput,  
    /// DateTimeInput,  
    /// BooleanInput,  
    /// <summary>
    public partial class IP_ModelParameter : UserControl,  
IModelParameter  
    {  
        #region Properties
        public ModelParameterInputStyle InputStyle { get; set; }  
        public bool IsMandatory { get; set; }  
        #endregion Properties
        public List<string> ParameterValueCollection  
        {  
            get  
            {  
                List<string> parValues = new  
List<string>();  
                foreach (var item in  
this.cboParameterValue.Items)  
parValues.Add(item.ToString());  
return parValues;  
            }  
        }  
        #endregion Properties
        #region IModelParameter
        public string ParameterValue { get; set; }  
public string ParameterAlias { get; set; }  
public XElement ParameterSetting()  
        {  
return new XElement("Parameter",  
XElement("ParameterName", this.ParameterName),  
XElement("IsMandatory", (this.IsMandatory) ? "Yes" :  
"No"),  
XElement("ParameterAlias", this.ParameterAlias),  
XElement("ParameterType", this.ParameterType),  
XElement("ParameterInputStyle", this.InputStyle),  
new XElement("ParameterValue", this.ParameterValue)  
);  
        }  
        #endregion IModelParameter
        #region Constructors
        public IP_ModelParameter(ModelParameterInputStyle ParameterInputStyle)  
        {  
this.InputStyle = ParameterInputStyle;  
InitializeComponent();  
        }  
        #endregion Constructors
        public IP_ModelParameter(XElement Element)  
        {  
this.ParameterName =  
Definition.Element("ParameterName").Value;  
this.IsMandatory =  
Definition.Element("IsMandatory").Value == "Yes" ?  
true : false;  
this.ParameterAlias =  
Definition.Element("ParameterAlias").Value;  
this.ParameterType =  
Definition.Element("ParameterType").Value;  
this.InputStyle =  
(ModelParameterInputStyle)Enum.Parse(typeof(ModelParameterInputStyle),  
Definition.Element("ParameterInputStyle").Value);  
this.ParameterValue =  
Definition.Element("ParameterValue").Value;  
InitializeComponent();  
        }  
    }
}
The resource layer of the detailed-level purposeful visualization system architecture is composed of four resource pools, which store data, model, solver and scenario related components for information generation, representation and presentation. The implementation of these resource pools is detailed in sections F.1 - F.4.
F.1 Data Pool

The key class supporting the Data Pool at the resource layer of the purposeful visualization system prototype is presented in Figure F.1.

Figure F.1 Class diagram of data sources

The essential code snippets implementing the above class are shown as follows:

```csharp
using System.Windows.Controls;
using System.Windows;
using System.Xml.Linq;
namespace CAVE_VS
{
    public partial class CavDataSource : UserControl, ICavComponent, IVisual
    {
        #region Properties
        public static readonly DependencyProperty DataSourceNameProperty = DependencyProperty.Register("
            DataSourceName",
            typeof(string),
            typeof(CavDataSource),
            new PropertyMetadata(null));
        public string ComponentName { get; set; }
        public string ComponentSettingFile { get; set; }
        public string DataProvider { get; set; }
        public XElement Input { get; set; }
        public XElement Output { get; set; }
        public UIElement Preview { get { return this.vbPreview; } }
        public bool IsInstantiated()
        {
            return (!System.String.IsNullOrEmpty(this.ComponentSettingFile));
        }
        public void ShowComponentSetting()
        {
            DS_Designer designer = new DS_Designer();
            //If this is a new data source, show data model editor with NewModelWizard on start.
            //Otherwise, simply load the existing data model file.
            if (IsInstantiated())
            {
                designer.Edit(this.ComponentSettingFile);
            }
            else
            {
                designer.New();
                this.ComponentSettingFile = designer.DataSourceDefinitionFile;
            }
            this.DataConnectionString = designer.DataConnectionString;
            this.DataProvider = designer.DataProvider;
            this.ComponentName = designer.DataSourceName;
            this.Output = new XElement("DataSource",
                new XAttribute("Name", this.ComponentName),
                new XAttribute("ComponentType", this.ComponentType),
                new XAttribute("DataSource", this.DataConnectionString),
                new XAttribute("DataConnection", this.DataConnectionString),
                new XAttribute("Provider", this.DataProvider)
            );
            this.Input = new XElement("Input",
                new XAttribute("DataSource", this.DataSourceName),
                new XAttribute("Provider", this.DataProvider)
            );
        }
        #endregion
    }
}
```
F.2 Model Pool

The Model Pool stores models for information generation, representation and presentation. The key class supporting the Model Pool at the resource layer of the purposeful visualization system prototype are presented in Figure F.2 – F.4.

![Class diagram of information generation models](image)

Figure F.2 Class diagram of information generation models
Figure F.3  Class diagram of information representation models
The essential code snippets implementing the above classes are shown as follows:

**IG Models**

```csharp
using System.Windows.Controls;
using System.Windows;
using System.Xml.Linq;
using Korzh.EasyQuery;
namespace CAVE_VS
{
    /// <summary>
    /// Interaction logic for CavIGModel.xaml
    /// </summary>
    public partial class CavIGModel : UserControl, ICavComponent, IVisual
    {
        #region Properties
        public Query IGModelQuery { get; set; }
        public string QueryStatement { get; set; }
        #endregion Properties
        #region ICavComponent
        public string ComponentName
        {
            get { return lblIGModelName.Text; }
            set { lblIGModelName.Text = value; }
        }
        public string ComponentSettingFile { get; set; }
        public CavComponentType ComponentType
        {
            get { return (CavComponentType)IGModelType; }
        }
        public XElement Input { get; set; }
        public XElement Output { get; set; }
        public UIElement Preview { get { return DataContext; } }
        public bool IsInstantiated()
        { return !(this.IGModelQuery == null); }
        public void ShowComponentSetting()
        { if (this.Input != null) }
        #endregion ICavComponent
        #region Methods
        public void AddDataInput()
        { }
        public void ApplyContext()
        { }
        public CavIGModel(IGQueryModel + overload)
        { }
        public void InitializeQuery()
        { }
        public EventData<br>PreviewItemUpdated
        { }
        public void ShowComponentSetting()
        { }
        public void ShowDetails()
        { }
        public void ShowPreview()
        { }
        public void ShowPresentation()
        { }
        public void ShowTimeline()
        { }
        public void ShowLifecycle()
        { }
        #endregion Methods
        #region Events
        public delegate void EventData<br>PreviewItemUpdated
        { }
        public delegate void EventData<br>PreviewItemUpdated
        { }
        public delegate void EventData<br>PreviewItemUpdated
        { }
        public delegate void EventData<br>PreviewItemUpdated
        { }
        public delegate void EventData<br>PreviewItemUpdated
        { }
        #endregion Events
    }
}
```

**Figure F.4** Class diagram of information presentation models

The essential code snippets implementing the above classes are shown as follows:
The following code snippets implement the IR Models demonstrated in chapters 6 and 7, i.e.
small multiples model, polygon set model, animated polygon set model, dimension chart and
tag cloud model.

**IR Models**

```csharp
namespace IRModels
{
    public class IR_SmallMultiples : IVisual
    {
        #region Variables
        XElement Definition;
        MarkerType MarkerType;
        string ChartTitle, MarkerTitle, MarkerSize,
        MarkerDescription1, MarkerDescription2;
        double ChartMaxWidth, MinSize, MaxSize,
        MaxMarkerSize;
        Color MarkerColor;
        DataTable DataInput;
        #endregion

        #region Constructors
        public IR_SmallMultiples(XElement IRModelDefinition)
        {
            Definition = IRModelDefinition;
            InitializeComponent();
            this.FieldName = "GF Model";
        }
        // Initialize CavIGModel from xml definition
        public CavIGModel(XElement IRModelDefinition)
        {
            InitializeComponent();
            this.FieldName = IRModelDefinition.Attribute("Name").Value;
            // Initialize Input
            this.Input = IRModelDefinition.Element("DataSource");
            // Initialize Output
            this.Output = IRModelDefinition;
            // Initialize query
            this.IGModelQuery = new IGModelQuery();
            string connectionString = this.Input.Element("ConnectionString").Value;
            // Initialize IGMov and IGModel
            IGModel IGModel = new IGModel();
            IGModel.LoadFromFile(this.Input.Element("DefinitionFile").Value);
            this.IGModelQuery.Model = IGModel;
            this.IGModelQuery.LoadFromXmlReader(IGModelDefinition.Element("Query")CreateReader());
            ShowDetailView();
            this.vbPreview.Child = this.VisualPresentation();
            Region Constructor
        }
        #endregion

        #region Methods
        public void SetDataSource(XElement Definition)
        {
            if (generation.HasValidInput(Definition))
            { this.Input = generation.GetDataSource(Definition);
                this.IGModelQuery.Model = IGModel;
            }
            else
            { this.Input = IGModelDefinition;
                ShowDetailView();
                this.vbPreview.Child = this.VisualPresentation();
            }
        }
        #endregion
    }
}
```

The following code snippets implement the IR Models demonstrated in chapters 6 and 7, i.e.
small multiples model, polygon set model, animated polygon set model, dimension chart and
tag cloud model.
private void Initiation()
{
    XElement Parameters = Definition.Descendants("Parameters");
    IR_ModelParameter(ModelParameterInputStyle.ParameterInput, IGModelDefinition)
    {
        ParameterName = "MarkerTitle",
        ParameterAlias = "marker title",
        ParameterType = "String",
        IsMandatory = true
    });
    parameters.Add(
new IR_ModelParameter(ModelParameterInputStyle.ParameterInput, IGModelDefinition)
    {
        ParameterName = "MarkerDescription1",
        ParameterAlias = "marker description 1",
        ParameterType = "String",
        IsMandatory = false
    });
    parameters.Add(
new IR_ModelParameter(ModelParameterInputStyle.ParameterInput, IGModelDefinition)
    {
        ParameterName = "MarkerDescription2",
        ParameterAlias = "marker description 2",
        ParameterType = "String",
        IsMandatory = false
    });
    parameters.Add(
new IR_ModelParameter(ModelParameterInputStyle.TextInput, IGModelDefinition)
    {
        ParameterName = "MaxMarkerSize",
        ParameterAlias = "marker max size",
        ParameterType = "Double",
        ParameterValue = "40",
        IsMandatory = false
    });
    parameters.Add(
new IR_ModelParameter(ModelParameterInputStyle.TextInput, IGModelDefinition)
    {
        ParameterName = "ChartMaxWidth",
        ParameterAlias = "chart max width",
        ParameterType = "Double",
        ParameterValue = "400",
        IsMandatory = false
    });
    return parameters;
}

Region Methods
#region IVisual
public UIElement VisualRepresentation()
{
    Representation.ApplyContext(Definition, ref this.DataInput);
    List(UIElement) chartContent = new List(UIElements());
    foreach (DataRow row in this.DataInput.Rows)
    {
if (row.RowState != DataRowState.Deleted)
    chartContent.Add(GetMarker(row));
}
return new CollectionView(chartContent) {
    Title = ChartTitle,
    Margin = new Thickness(2)
};
}

public UIElement VisualPresentation()
{
    return VisualRepresentation();
}

// PolygonSet Model
using System;
using System.Xml.Linq;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
return Color.FromArgb((byte)b);
}
#endregion

#region IVisual
public UIElement VisualPresentation()
{
    // Apply context (Definition, ref this.DataInput);
    List<CavPolygon> polygons = new List<CavPolygon>();
    // Loop through all child nodes
    foreach (DataRow row in DataInput.Rows)
    {
        if (row.RowState != DataRowState.Deleted)
        {
            CavPolygon polygon = new CavPolygon();
        } else {
            polygon.PolygonDescription = row[PolygonDescription].ToString();
            polygon.PolygonShape = row[PolygonShape].ToString();
            polygon.PolygonLocations = row[PolygonLocations].ToString();
            polygon.PolygonColor = PolygonColor;
            polygon.PolygonAnimationStart = polygon.PolygonAnimationEnd = polygon.PolygonValue = polygon.PolygonOpacity = polygon.PolygonDescription = row[PolygonDescription].ToString();
            polygon.ComponentData(Definition, row);
            polygons.Add(polygon);
        }
    }
    return new PolygonCollection(polygons);
}
public UIElement VisualRepresentation()
{
    DistributionMapLayer maplayer = new DistributionMapLayer();
    maplayer.Add(VisualPresentation());
    return new DistributionMapLayer(maplayer);
}
#endregion

#region Animation
using System;
using System.Xml.Linq;
using System.Linq;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Collections.Generic;
using System.Linq;
using System;
namespace CAVE_VS
{
    public class IR_AnimatedPolygonSet_Visibility : IVisual
    {
        #region Variables
        XElement Definition;
        string Title, PolygonShape, PolygonValue, PolygonDescription, PolygonAnimationStart, PolygonAnimationEnd, TimelineInterval;
        Color StartColor, EndColor, double PolygonOpacity, MaxValue, MinValue, AnimationSpeed, MapWidth, MapHeight, DataInput;
        #endregion
        #region Constructors
        public IR_AnimatedPolygonSet_Visibility(XElement IRModelDefinition)
        {
            Definition = IRModelDefinition;
            Initialization();
        }
        #endregion
        #region Methods
        public static ObservableCollection<IR_ModelParameter> Parameters(XElement IXModelDefinition)
        {
            ObservableCollection<IR_ModelParameter> parameters = new ObservableCollection<IR_ModelParameter>();
            parameters.Add(new IR_ModelParameter(ModelParameterInputStyle.TextInput, IXModelDefinition)
            {
                ParameterName = "PolygonShape", ParameterAlias = "Polygon shape", ParameterType = "String", IsMandatory = true
            });
            parameters.Add(new IR_ModelParameter(ModelParameterInputStyle.TextInput, IXModelDefinition)
            {
                ParameterName = "PolygonValue", ParameterAlias = "Polygon value", ParameterType = "Double", IsMandatory = true
            });
            parameters.Add(new IR_ModelParameter(ModelParameterInputStyle.TextInput, IXModelDefinition)
            {
                ParameterName = "PolygonDescription", ParameterAlias = "Polygon description", ParameterType = "String", IsMandatory = true
            });
            return parameters;
        }
    }
    #endregion
}
TimelineInterval = Representation.GetParameterValue(Parameters, "TimelineInterval") ;
AnimationSpeed = Representation.GetParameterValue(Parameters, "AnimationSpeed") ;
DataInput = Generation.IGModelData(Definition.Element("IGModel"));
// Set polygon value range
List<double> polygonValues = new List<double>();
foreach (DataRow row in DataInput.Rows) {
    if (string.IsNullOrEmpty(row[PolygonValue].ToString()))
        polygonValues.Add(0);
    else
        polygonValues.Add(double.Parse(row[PolygonValue].ToString()));
}
MinValue = polygonValues.Min();
MaxValue = polygonValues.Max();
private Color ConvertDoubleToColor(double value) {
    double ratio = (value - MinValue) / (MaxValue - MinValue);
    return Color.FromArgb((byte)r, (byte)g, (byte)b);
}
#endregion Methods
#region IVisual
cIIDVisual ...
  { ... }
#endregion IVisual

240
using System;  
using System.Collections.Generic;  
using System.Data;  
using System.Windows;  
using IRControls;  
namespace CAVE_VS  
{  
    public class IR_DimensionChart : IVisual  
    {  
        #region Variables  
        XElement Definition;  
        string Title, DimensionBarCategory, DimensionBarDescription, DimensionBarScale, DimensionBarComponentDescriptionItemLabels, DimensionBarComponentDescription, DimensionBarComponentCurrentValue, double ContentLength;  
        bool Visibility, DimensionBarCategoryVisibility, DimensionBarDescriptionVisibility, DimensionBarScaleVisibility;  
        DataInput;  
        #endregion Variables  
        #region Methods  
        public static ObservableCollection<IR_ModelParameter> Parameters(XElement Definition)  
        {  
            ObservableCollection<IR_ModelParameter> parameters = new ObservableCollection<IR_ModelParameter>();  
            parameters.Add(  
                new IR_ModelParameter(ModelParameterInputStyle.ParameterInput, IModelDefinition)  
                {  
                    ParameterName = "DimensionBarCategory",  
                    ParameterAlias = "Dimension category",  
                    ParameterType = "String",  
                    ParameterValue = "String",  
                    IsMandatory = false  
                });  
            parameters.Add(  
                new IR_ModelParameter(ModelParameterInputStyle.ParameterInput, IModelDefinition)  
                {  
                    ParameterName = "DimensionBarDescription",  
                    ParameterAlias = "Dimension description",  
                    ParameterType = "String",  
                    ParameterValue = "String",  
                    IsMandatory = false  
                });  
            return parameters;  
        }  
        public Element CreateDefinition()  
        {  
            return new Element();  
        }  
        public Element CreateUIElement()  
        {  
            return new Element();  
        }  
        public Element CreateDefinitionAndUIElement()  
        {  
            return new Element();  
        }  
    }  
}  
}  
}
parameters.Add(new IR_ModelParameter(new ModelParameterInputStyle.BooleanInput, IGModelDefinition) {
    ParameterName = "DimensionBarDescriptionVisibility",
    ParameterAlias = "Is dimension description visible",
    ParameterType = "Boolean",
    IsMandatory = false
});
parameters.Add(new IR_ModelParameter(new ModelParameterInputStyle.BooleanInput, IGModelDefinition) {
    ParameterName = "DimensionBarScaleVisibility",
    ParameterAlias = "Is dimension scale visible",
    ParameterType = "Boolean",
    IsMandatory = false
});
return parameters;
}
private void Initialization() {
    XElement Parameters = Definition.Element("Parameters");
    Title = Representation.GetDataInputValue(Parameters, "Title");
    DimensionBarCategory = Representation.GetDataInputValue(Parameters, "DimensionBarCategory");
    DimensionBarDescription = Representation.GetDataInputValue(Parameters, "DimensionBarDescription");
    DimensionBarScale = Representation.GetDataInputValue(Parameters, "DimensionBarScale");
    DimensionBarCategory = Representation.GetDataInputValue(Parameters, "DimensionBarCategory");
    DimensionBarDescriptionVisibility = Representation.GetDataInputValue(Parameters, "DimensionBarDescriptionVisibility");
    DimensionBarScaleVisibility = Representation.GetDataInputValue(Parameters, "DimensionBarScaleVisibility");
    DimensionBarDescriptionVisibility = Representation.GetDataInputValue(Parameters, "DimensionBarDescriptionVisibility");
    DimensionBarScaleVisibility = Representation.GetDataInputValue(Parameters, "DimensionBarScaleVisibility");
    ContentLength = DataInput.Rows[i]["ContentLength"];
    VisibilityVisible = Visibility.Visible; 
    VisibilityCollapsed = Visibility.Collapsed;
    visibility = String.IsNullOrEmpty(Representation.GetDataInputValue(Parameters, "VisibilityVisible"));
    visibility = String.IsNullOrEmpty(Representation.GetDataInputValue(Parameters, "VisibilityCollapsed");
    dimensionChart.ContentLength = ContentLength;
    dimensionChart.Title = Title;
    dimensionChart.DimensionBarCategory = DimensionBarCategory;
    dimensionChart.DimensionBarDescription = DimensionBarDescription;
    dimensionChart.DimensionBarScale = DimensionBarScale;
    dimensionChart.DescriptionVisibility = DimensionBarDescriptionVisibility;
    dimensionChart.ScaleVisibility = DimensionBarScaleVisibility;
    dimensionChart.CategoryVisibility = DimensionBarCategoryVisibility;
    dimensionChart.Orientation = barComponent.Orientation;
    if (i % 2 == 0)
        barComponent.Ori
    else
        barComponent.Orientation = DimensionBarOrientation.Upward;
    itmes.Add(componentList[j]);
}
for (int j = 0; j < componentList.Count; j++)
{
    DimensionBarComponent barComponent = new DimensionBarComponent();
    barComponent.DescriptionVisibility = String.IsNullOrEmpty(Representation.GetDataInputValue(Parameters, "DescriptionVisibility");
    barComponent.CategoryVisibility = String.IsNullOrEmpty(Representation.GetDataInputValue(Parameters, "CategoryVisibility");
    barComponent.ScaleVisibility = String.IsNullOrEmpty(Representation.GetDataInputValue(Parameters, "ScaleVisibility");
    barComponent.ContentLength = DataInput.Rows[i]["ContentLength"];
    barComponent.Title = DataInput.Rows[i]["Title"]; // Tag Cloud Model
    barComponent.Description = Representation.GetDataInputValue(Definitions, DataInput.Rows[i]["Description");
    componentList.Add(barComponent);
}
foreach (string row in DataInput.Rows)
{
    for (int j = 0; j < row.Count(); j++)
    }
```csharp
#region Variables
XElement Definition;
string Title, CloudItem, ItemMeasure;
double CloudWidth, CloudHeight;
DataTable DataInput;
Color ItemForeground, ItemBackground;
#endregion Variables

#region Constructors
public IR_CollectionCould(XElement IRModelDefinition) {
    Definition = IRModelDefinition;
    Initiation();
}
#endregion Constructors

#region Methods
public static ObservableCollection<IR_ModelParameter>
Parameters(XElement IRModelDefinition) {
    ObservableCollection<IR_ModelParameter> parameters = new;
    ObservableCollection<IR_ModelParameter> parameters;
    IR_ModelParameter new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "Title",
        ParameterAlias = "Title",
        ParameterType = "String",
        ParameterValue = "title",
        IsMandatory = false
    });
    parameters.Add(new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "CloudWidth",
        ParameterAlias = "Cloud width",
        ParameterType = "Double",
        ParameterValue = "400",
        IsMandatory = false
    });
    parameters.Add(new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "CloudHeight",
        ParameterAlias = "Cloud height",
        ParameterType = "Double",
        ParameterValue = "400",
        IsMandatory = false
    });
    parameters.Add(new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "ItemMeasure",
        ParameterAlias = "Cloud item size",
        ParameterType = "Double",
        IsMandatory = false
    });
    parameters.Add(new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "ItemForeground",
        ParameterAlias = "Item foreground",
        ParameterType = "String",
        IsMandatory = false
    });
    parameters.Add(new
    IR_ModelParameter(ModelParameterInputStyle.TextBox, IRModelDefinition) {
        ParameterName = "ItemBackground",
        ParameterAlias = "Item background",
        ParameterType = "String",
        IsMandatory = false
    });
    return parameters;
}
#endregion Methods

// Generation of the definition
new IR_ModelParameter(ModelParameterInputStyle.ColorInput, IRModelDefinition) {
    ParameterName = "ItemBackground",
    ParameterAlias = "Item background",
    IsMandatory = false
};
CollectionCloud cloud = new
CollectionCloud(cloudItems);
cloud.Title = Title;
cloud.CloudWidth = (CloudWidth < 200) ? 200 : CloudWidth;
cloud.CloudHeight = (CloudHeight < 200) ? 200 : CloudHeight;
return cloud;
```
public UIElement VisualPresentation()
{
    return VisualRepresentation();
}
#endregion IVisual

public UIElement VisualPresentation()
{
    return VisualRepresentation();
}
#endregion IVisual

#region IVisual
public UIElement VisualPresentation()
{
    return VisualRepresentation();
}
#endregion IVisual

public class IP_DeepZoom : IVisual
{
    #region Variables
    XElement Definition;
    #endregion Variables

    #region Constructors
    public IP_DeepZoom(XElement IPModelDefinition)
    {
        Definition = IPModelDefinition;
    }
    #endregion Constructors

    #region Methods
    public static void RenderVisualOutput(XElement Definition, DesignerCanvas TargetCanvas)
    {
        TargetCanvas.Children.Clear();
        // Render IPModels
        if (Definition.Element("IPModels").Elements("IPModel").Any())
        {
            foreach (XElement IPDefinition in Definition.Elements("IPModel"))
            {
                DesignerItem item = Design.NewDesignerItem();
                item.Content = new CaviIPModel(IPDefinition);
                // Add IPModel
                IXMLElement string s = IPDefinition.Element("Presentation").Element("X").Value;
                double X = (string.IsNullOrEmpty(s)) ? 10 : double.Parse(s);
                s = IPDefinition.Element("Presentation").Element("Y").Value;
                double Y = (string.IsNullOrEmpty(s)) ? 10 : double.Parse(s);
                TargetCanvas.AddDesignerItem(item, new Point(X, Y));
            }
            // Render child IPModels
            if (Definition.Element("IPModels").Elements("IPModel").Any())
            {
                foreach (XElement IPDefinition in Definition.Elements("IPModel"))
                {
                    DesignerItem item = Design.NewDesignerItem();
                    item.Content = new CaviIPModel(IPDefinition);
                    // Add IPModel
                    IXMLElement string s = IPDefinition.Element("Presentation").Element("X").Value;
                    double X = (string.IsNullOrEmpty(s)) ? 10 : double.Parse(s);
                    s = IPDefinition.Element("Presentation").Element("Y").Value;
                    double Y = (string.IsNullOrEmpty(s)) ? 10 : double.Parse(s);
                    TargetCanvas.AddDesignerItem(item, new Point(X, Y));
                }
            }
        }
        // Extract IPModels
        if (Definition.Element("IPModels").Elements("IPModel").Any())
        {
            Definition.Element("IPModels").Elements("IPModel").Add(new XElement(IPDefinition));
        }
        // Render IPModels
        if (Definition.Element("IPModels").Elements("IPModel").Any())
        {
            foreach (XElement IPDefinition in Definition.Elements("IPModel"))
            {
                layers.Add(new XElement(IPDefinition));
                if (layers.Count > 0)
                {
                    List<SemanticLayerOverview> lstLayerOverviews = new List<SemanticLayerOverview>();
                    List<UIElement> lstUIElements = new List<UIElement>();
                    {
                        // Do something with each layer
                    }
                }
            }
        }
    }
}
CavComponentType componentType = (CavComponentType)Enum.Parse(typeOf(CavComponentType), layer.Attribute("ComponentType").Value);
  if (componentType == CavComponentType.IRModel)
  {
    IPModel IRModel = new IPModel()
    IRModel.ModelName = layer.Attribute("IRModelName").Value;
    IRModel.Description = layer.Attribute("IRModelDescription").Value;
    lstLayerOverviews.Add(overview);
    lstVisualLayers.Add(IRModel);
  }
  else if (componentType == CavComponentType.IPModel)
  {
    IPModel IPModel = new IPModel()
    IPModel.ModelName = layer.Attribute("IPModelName").Value;
    IPModel.Description = layer Attribute("IPModelDescription").Value;
    lstLayerOverviews.Add(overview);
    lstVisualLayers.Add(IPModel);
  }

  // Spatial Semantic Zoom Model
  using System.Xml.Linq;
  using System.Windows;
  using System.Linq;
  using System.Collections.Generic;
  using System.Collections.ObjectModel;
  using System;

namespace CAVE_VS
{
  public class IP_SpatialSemanticZoom : IVisual
  {
    #region Variables
    XElement Definition;
    #endregion Variables

    #region Constructors
    public IP_SpatialSemanticZoom(XElement IPModelDefinition)
    {
      Definition = IPModelDefinition;
    }

    #endregion Constructors

    #region Methods
    public static ObservableCollection<IP_ModelParameter> Parameters()
    { return new ObservableCollection<IP_ModelParameter>();
    }

    public static ObservableCollection<IP_ModelParameter>()
    { parameters = new ObservableCollection<IP_ModelParameter>();
      parameters.Add(new IP_ModelParameter(ModelParameterInputStyle.TextInput)
      {
        ParameterName = "Longitude",
        ParameterAlias = "Longituden",
        ParameterType = "Double",
        IsMandatory = true
      });

      parameters.Add(new IP_ModelParameter(ModelParameterInputStyle.TextInput)
      {
        ParameterName = "Latitude",
        ParameterAlias = "Latitude",
        ParameterType = "Double",
        IsMandatory = true
      });

      parameters.Add(new IP_ModelParameter(ModelParameterInputStyle.TextInput)
      {
        ParameterName = "ZoomSequence",
        ParameterAlias = "Zoom sequence",
        ParameterType = "Int",
        IsMandatory = true
      });
    }

    #endregion Methods

    // Timeline Model
    using System.Xml.Linq;
    using System.Windows;
    using System.Linq;
    using System.Collections.Generic;
    using System;
    using System.Collections.ObjectModel;

    public UIElement VisualPresentation()
    {
      return VisualRepresentation();
    }

    // Extract IPModels
    if (Definition.Element("IPModels").Elements("IPModel").Any())
    { foreach (XElement IPDefinition in Definition.Element("IPModels").Elements("IPModel"))
      layers.Add(new XElement(IPDefinition));
    }

    // Extract IRModels
    if (Definition.Element("IRModels").Elements("IRModel").Any())
    { foreach (XElement IRDefinition in Definition.Element("IRModels").Elements("IRModel"))
      layers.Add(new XElement(IRDefinition));
    }

    if (layers.Count > 0)
    {
      SpatialSemanticLayerOverview overview = new SpatialSemanticLayerOverview()
      { LayerName = layer.Attribute("Name").Value,
        LayerDescription = layer.Element("Description").Value,
        latitude = lat,
        longitude = lon
      },
      lstLayerOverviews.Add(overview);
      lstVisualLayers.Add(Design.CavComponent(layer));
    }
  }
}
namespace CAVE_VS
{

class IP_Lifecycle : ISingleView
{
    #region Variables
    XElement Definition;
    #endregion

    #region Constructors
    public IP_Lifecycle(XElement IPModelDefinition)
    {
        Definition = IPModelDefinition;
    }
    #endregion

    #region Methods
    public static ObservableCollection<IP_ModelParameter> Parameters()
    {
        ObservableCollection<IP_ModelParameter> parameters = new ObservableCollection<IP_ModelParameter>();
        parameters.Add(
            new IP_ModelParameter(ModelParameterInputStyle.DateTimeInput)
            {
                ParameterName = "StartDate",
                ParameterAlias = "Start date",
                ParameterType = "string",
                ParameterValue = DateTime.Today.ToString()
            });
        parameters.Add(
            new IP_ModelParameter(ModelParameterInputStyle.DateTimeInput)
            {
                ParameterName = "EndDate",
                ParameterAlias = "End date",
                ParameterType = "string",
                ParameterValue = DateTime.Today.ToString()
            });
        return parameters;
    }

    public static TimelineEvent GetTimelineEvent(XElement Definition)
    {
        TimelineEvent e = new TimelineEvent()
        {
            ID = Definition.Attribute("Name").Value,
            Title = Definition.Attribute("Name").Value,
            DefaultValue = Presentation.GetParameterValue(Definition.Element("Presentation"), "EventColor"),
            RowOverride = -1,
            HeightOverride = -1.0,
            TopOverride = -1.0,
            StartDate = DateTime.Parse(Definition.Attribute("Presentation"), "Start Date"),
            EndDate = DateTime.Parse(Definition.Attribute("Presentation"), "End Date"),
            IsMandatory = (Presentation.GetParameterValue(Definition.Element("Presentation"), "Is MANDATORY") == "Yes")
        };

        CavComponentType componentType = (CavComponentType)Enum.Parse(typeof(CavComponentType), Definition.Attribute("ComponentType").Value);
        if (componentType == CavComponentType.IPModel)
        {
            if (Definition.Element("IPModels").Elements("IPModel").Any())
                foreach (XElement IRModelDefinition in Definition.Element("IRModels").Elements("IRModel"))
                {
                    XElement IRDefinition = IRModelDefinition;
                    if (IRDefinition.Element("IRModels").Elements("IPModel").Any())
                        foreach (XElement IPModel in IRDefinition.Element("IRModels").Elements("IPModel"))
                            events.Add(new XElement(IPDefinition));
                    events.Add(new XElement(IPDefinition));
                    timelines.Add(new TimelineViewer()
                    {
                        TimelineViewer()
                        {
                            DateTime StartDate = events.Min(e => DateTime.Parse(Presentation.GetParameterValue(e.Element("Presentation"), "Start Date")));
                            DateTime EndDate = events.Max(e => DateTime.Parse(Presentation.GetParameterValue(e.Element("Presentation"), "End Date")));
                            timeline.SetTimelineRange(StartDate, EndDate);
                            timeline.SetTimelineContent(events.Select(e => GetTimelineEvent()).ToList());
                            return timeline;
                        }
                    });
                }
            else
            {
                public UIElement VisualRepresentation()
                {
                    return VisualRepresentation();
                }
            }
        }

        // Process Model
        using System.Linq;
        using System.Windows;
        using System.Collections.ObjectModel;
        using System.Collections.Generic;
        using System;
        namespace CAVE_VS
        {
            public class IP_Lifecycle : ISingleView
            {
                #region Variables
                XElement Definition;
                #endregion

                #region Constructors
                public IP_Lifecycle(XElement IPModelDefinition)
                {
                    Definition = IPModelDefinition;
                }
                #endregion

                #region Methods
                public static ObservableCollection<IP_ModelParameter> Parameters()
                {
                    ObservableCollection<IP_ModelParameter> parameters = new ObservableCollection<IP_ModelParameter>());
                    parameters.Add(
                    }

```
new IP_ModelParameter(ModelParameterInputStyle.TextInput)
{
  ParameterName = "PhaseIndex",
  ParameterAlias = "Lifecycle phase index",
  ParameterType = "Int",
  ParameterValue = "1",
  IsMandatory = true
};
parameters.Add(new
new IP_ModelParameter(ModelParameterInputStyle.TextInput)
{
  ParameterName = "ComponentWidth",
  ParameterAlias = "Component width",
  ParameterType = "Double",
  ParameterValue = "400",
  IsMandatory = false
});
parameters.Add(new
new IP_ModelParameter(ModelParameterInputStyle.TextInput)
{
  ParameterName = "ComponentHeight",
  ParameterAlias = "Component height",
  ParameterType = "Double",
  ParameterValue = "400",
  IsMandatory = false
});
return parameters;
}

#region Methods
#region IVisual
public UIElement VisualRepresentation()
{
  List<XElement> phases = new List<XElement>();
  // Extract IRModels
  if (Definition.Element("IRModels").Elements("IRModel").Any())
  foreach (XElement IRDefinition in Definition.Element("IRModels").Elements("IRModel"))
  phases.Add(new XElement(IRDefinition));
  // Extract IPModels
  if (Definition.Element("IPModels").Elements("IPModel").Any())
  foreach (XElement IPDefinition in Definition.Element("IPModels").Elements("IPModel"))
  phases.Add(new XElement(IPDefinition));
  if (phases.Count > 0)
  {
    LifecycleViewer lifecycle = new
    foreach (XElement phase in phases.OrderBy(p =>
      int.Parse(Presentation.GetParameterValue(p.Element("Presentation"), "PhaseIndex")))))
    {
      CavComponentType componentType = (CavComponentType)Enum.Parse(typeof(CavComponentType),
        phase.Attribute("ComponentType").Value);
      if (componentType == CavComponentType.IRModel)
      {
        CavIRModel IRModel = new CavIRModel(phase)
        {
          Width = double.Parse(Presentation.GetParameterValue(phase.Element("Presentation"), "ComponentWidth")),
          Height = double.Parse(Presentation.GetParameterValue(phase.Element("Presentation"), "ComponentHeight"))
        };
        lifecycle.Lifecycle.AddPhase(IRModel);
      } else if (componentType == CavComponentType.IPModel)
      {
        CavIPModel IPModel = new CavIPModel(phase)
        {
          Width = double.Parse(Presentation.GetParameterValue(phase.Element("Presentation"), "ComponentWidth")),
          Height = double.Parse(Presentation.GetParameterValue(phase.Element("Presentation"), "ComponentHeight"))
        };
        lifecycle.Lifecycle.AddPhase(IPModel);
      }
    }
    return lifecycle;
  } else
  return null;
}
public UIElement VisualPresentation()
{
  return VisualRepresentation();
}
#endregion IVisual

F.3 Solver Pool

The Solver Pool stores solvers for information generation, representation and presentation. The key classes supporting the Solver Pool at the resource layer of the purposeful visualization system prototype are presented in Figure F.5.

![Class diagram of IG, IR and IP solvers](image)

Figure F.5 Class diagram of IG, IR and IP solvers

The essential code snippets implementing the above classes are shown as follows:

**IG Solvers**

```csharp
using System;
using System.Xml.Linq;
using System.Linq;
using System.Data;
using System.Data.SqlClient;
namespace CAVE_VS
{
    public static class Generation
    {
        // Check if an XML definition contains a data source
        public static bool HasValidInput(XElement definition)
        {
            if (definition != null)
            {
                if (definition.Attribute("ComponentType").Value == "DataSource" ||
                    definition.Descendants("DataSource").Any())
                {
                    return hasDataSource = true;
                }
            }
        }
    }
}
```

```csharp
// Extract a data source from an XML definition
public static XElement GetDataSource(XElement definition)
{
    if (definition == null)
        return null;
    if (definition.Attribute("ComponentType").Value == "DataSource")
        return definition;
    else if (definition.Descendants("DataSource").Any())
        return definition.Descendants("DataSource").First();
    else
        return null;
}
```

```csharp
// Generate the data of an IGModel
// To be expanded to enable accessing Access and Oracle databases
public static DataTable IGModelData(XElement IGModelDefinition)
{
    string tableName = IGModelDefinition.Attribute("Name").Value;
    DataTable data = new DataTable(tableName);
    // Code to generate data...
}
```

---

248
string conn = IGModelDefinition.Element("DataSource").Element("ConnectionString").Value;
SqlConnection sqlCon = new SqlConnection(conn.Replace("Provider=SQLCLIEDB;", ""));
try {
  if (IGModelDefinition.Element("Parameters").Elements("Parameter").Count() > 0)
    {
      string sqlCommand = IGModelDefinition.Element("CommandText").Value;
      sqlCon.Open();
      // Initialize model content using (SqlDataAdapter da = new SqlDataAdapter(sqlCommand, sqlCon))
      { 
        da.Fill(data);
        foreach (DataColumn col in data.Columns)
        {
          col.ColumnName = GetStandardName(col.ColumnName);
        }
        (HasParameterFilter(IGModelDefinition, data))
        FilterDataTable(IGModelDefinition, data);
      }
    }
  catch (Exception ex)
    {
    sqlCon.Close();
  }
  System.Windows.MessageBox.Show(ex.Message);
} return data;
// Check if an IGModelDefinition includes filters
public static bool HasParameterFilter(XElement IGModelDefinition, DataTable Table)
{
  bool hasParameterFilter = false;
  if (Table.Columns.Count > IGModelDefinition.Element("Parameters").Elements("ParameterValue").Count())
    { 
      hasParameterFilter = true;
      else if (IGModelDefinition.Element("Parameters").Descendants("ParameterValue").Any())
      { 
        hasParameterFilter = true;
      return hasParameterFilter;
    }
  private static void FilterDataTable(XElement IGModelDefinition, DataTable Table)
    {
      for (int i = (Table.Columns.Count - 1); i >= 0; i--)
        {
          bool isParAdded = IGModelDefinition.Element("Parameters").Elements("Parameter").Any(p => p.Element("ParameterName").Value == Table.Columns[i].ColumnName);
            if (!isParAdded) 
            Table.Columns.RemoveAt(i);
          string filters = "";
          // Apply parameter filters 
          foreach (XElement par in IGModelDefinition.Element("Parameters").Elements("Parameter"))
          {
            if (par.Elements("ParameterValue").Any())
              {
                string filter = par.Element("ParameterName").Value + " = " + par.Element("ParameterValue").Value + " ";
                if (filters != "")
                  filters = filters + ", AND " + filter;
                else
                  filters = filter;
            }
          }
        Table = Table.Select(filters).CopyToDataTable();
      }
    }
  public static class Representation
  {
    public static Color HexToColor(string hex)
      { 
        return String.IsNullOrEmpty(hex) ? 
        (Color)ColorConverter.ConvertFromString("#ff15428b") : 
        (Color)ColorConverter.ConvertFromString(hex);
      }
      public static string ColorToHex(Color color, bool alphaChannel)
        { 
          return String.Format("#{0}{1}{2}{3}", color.A.ToString("X2"), color.R.ToString("X2"), color.G.ToString("X2"), color.B.ToString("X2"));
        }
    public static string BoolToString(bool value)
      { 
        return value ? "Yes" : "No";
      }
      // Check if an XML definition contains an IGModel public static bool HasValidInput(XElement definition)
    {
      bool hasIGModel = false;
      if (definition.Attribute("ComponentType").Value == "IGModel")
        { 
          if (definition.Attribute("ComponentType").Value == "IQModel")
            return true;
          return false;
        }
      return true;
    // Extract an IGModel from an XML definition public static XElement GetInput(XElement definition)
    {
      if (definition == null)
        return null;
      else if (definition.Attribute("ComponentType").Value == "IGModel")
        return defination;
      else if (definition.Attribute("ComponentType").Value == "IRModel")
        return defination;
      else if (definition.Descendants("IGModel").First())
        return defination;
      else
        return null;
    }
    public static string GetParameterValue(XElement IGModelParameters, string ParameterName)
    {
      hasIGModel = true;
      return hasIGModel;
    }
    public static bool IsIGModelRequired(CavComponentType IRModelType)
      { 
        if (IRModelType == CavComponentType.IR_Image)
          return false;
        return true;
      }
    // Extract an IGModel from an XML definition public static XElement GetInput(XElement definition)
    {
      if (definition == null)
        return null;
      else if (definition.Attribute("ComponentType").Value == "IGModel")
        return defination;
      else if (definition.Attribute("ComponentType").Value == "IRModel")
        return defination;
      else if (definition.Descendants("IGModel").Any())
        return defination;
      else
        return null;
    }
  }
  using System;
  using System.Windows.Media;
  using System.Xml.Linq;
  using System.Linq;
  using System.Data;
  namespace CAV_V5
  {
    public static class Representation
    {
      public static Color HexToColor(string hex)
        { 
          return String.IsNullOrEmpty(hex) ? 
          (Color)ColorConverter.ConvertFromString("#ff15428b") : 
          (Color)ColorConverter.ConvertFromString(hex);
        }
        public static string ColorToHex(Color color, bool alphaChannel)
          { 
            return String.Format("#{0}{1}{2}{3}", color.A.ToString("X2"), color.R.ToString("X2"), color.G.ToString("X2"), color.B.ToString("X2"));
          }
        public static string BoolToString(bool value)
          { 
            return value ? "Yes" : "No";
          }
          // Check if an XML definition contains an IGModel public static bool HasValidInput(XElement definition)
        { 
          bool hasIGModel = false;
          if (definition.Attribute("ComponentType").Any())
            { 
              if (definition.Attribute("ComponentType").Value == "IGModel")
                || (definition.Attribute("ComponentType").Value == "IQModel")
                return true;
              return false;
            }
          return true;
        }
        // Extract an IGModel from an XML definition public static XElement GetInput(XElement definition)
        {
          if (definition == null)
            return null;
          else if (definition.Attribute("ComponentType").Value == "IGModel")
            return defination;
          else if (definition.Attribute("ComponentType").Value == "IRModel")
            return defination;
          else if (definition.Descendants("IGModel").Any())
            return defination;
          else
            return null;
        }
        public static string GetParameterValue(XElement IGModelParameters, string ParameterName)
        {
          // Extract an IGModel from an XML definition public static XElement GetInput(XElement definition)
        { 
          if (definition == null)
            return null;
          else if (definition.Attribute("ComponentType").Value == "IGModel")
            return defination;
          else if (definition.Attribute("ComponentType").Value == "IRModel")
            return defination;
          else if (definition.Descendants("IGModel").Any())
            return defination;
          else
            return null;
        }
    }
  }

IR Solvers
```csharp
public static XElement GetParameterContext(XElement IRModelParameters, string ParameterName)
{
    XElement par = IRModelParameters.Elements("Parameter").First(p => p.Element("ParameterName").Value == ParameterName);
    string parValue = par.Element("ParameterValue").Value;
    if (par.Elements("ParameterContext").Any())
    {
        XElement parContext = par.Element("ParameterContext");
        if (parContext.Element("ParameterInputStyle").Value != ModelParameterInputStyle.ParameterInput.toString())
            parValue = parContext.Element("ParameterValue").Value;
        return parValue;
    }
    return null;
}

public static bool IsValidOpacity(string opacity)
{
    double dblOpacity;
    if (String.IsNullOrEmpty(opacity))
        return false;
    else if (double.TryParse(opacity, out dblOpacity))
        return (dblOpacity > 0) ? true : false;
    else
        return false;
}

public static bool IsValidSize(string size)
{
    double dblSize;
    if (String.IsNullOrEmpty(size))
        return false;
    else if (double.TryParse(size, out dblSize))
        return (dblSize > 0) ? true : false;
    else
        return false;
}

// Default data output
// Private static XElement DataOutput(XElement IRModelDefinition)
//
// Get data scenario that is not child scenario
public static XElement GetChildScenario(XElement IRModelDefinition, string ScenarioName)
{
    XElement sce = IRModelDefinition.Elements("VisualScenario").FirstOrDefault(p => p.Element("ScenarioName").Value == ScenarioName);
    return sce;
}

#region VisualOutput
// Render visual output based on IRModelDefinition
public static void VisualOutput(XElement IRModelDefinition)
{
    CavComponentType IRModelType = (CavComponentType)Enum.Parse(typeof(CavComponentType), IRModelDefinition.Attribute("IRModelType").Value);
    switch (IRModelType)
    {
    case CavComponentType.IR_DimensionChart: visualOutput = new IR_DimensionChart(IRModelDefinition); break;
    case CavComponentType.IR_FlowLines: visualOutput = new IR_FlowLines(IRModelDefinition); break;
    case CavComponentType.IR_PushpinSet: visualOutput = new IR_PushpinSet(IRModelDefinition); break;
    case CavComponentType.IR_PolygonSet: visualOutput = new IR_PolygonSet(IRModelDefinition); break;
    case CavComponentType.IR_QuestionList: visualOutput = new IR_QuestionList(IRModelDefinition); break;
    case CavComponentType.IR_SmallMultiples: visualOutput = new IR_SmallMultiples(IRModelDefinition); break;
    case CavComponentType.IR_StatisticalChart: visualOutput = new IR_StatisticalChart(IRModelDefinition); break;
    case CavComponentType.IR_TagCloud: visualOutput = new IR_TagCloud(IRModelDefinition); break;
    case CavComponentType.IR_Media: visualOutput = new IR_Media(IRModelDefinition); break;
    }
    return visualOutput;
}
```

using System.Collections.Generic;
using System.Linq;
using System;
using System.Xml.Linq;
namespace CAVE_V5

IP Solvers

using System.Collections.Generic;
using System.Linq;
using System;
using System.Xml.Linq;
namespace CAVE_V5

IP Model Types

using System.Collections.Generic;
using System.Linq;
using System;
using System.Xml.Linq;
namespace CAVE_V5
{  if (IPModelType ==  
  CavComponentType.IP_DeepZoom)  
    return ModelCategory.IPModelTypeI;  
  else  
    return ModelCategory.IPModelTypeII;  
}  
// Check if an XML definition contains valid input components for creating an IPModel  
// Such components could be IRModels and/or IPModels  public static bool HasValidInput(XElement definition)  
  {  
    bool hasValidInput = false;  
    if (definition != null)  
    {  
      if (definition.Attributes("ComponentType").Any())  
        {  
          if (definition.Attribute("ComponentType").Value == "IRModel" || definition.Attribute("ComponentType").Value == "IPModel")  
            hasValidInput = true;  
          else if  
            (definition.Descendants("IPModel").Any() || definition.Descendants("IRModel").Any())  
              hasValidInput = true;  
        }  
      return hasValidInput;  
    }  
    // Extract input components from an XML definition  
    public static List<XElement> GetInput(XElement definition)  
    {  
      if (definition == null)  
        return null;  
      else  
      {  
        string ComponentType = definition.Attribute("ComponentType").Value;  
        if (ComponentType == "IPModel" || ComponentType == "IRModel")  
          {  
            List<XElement> lst = new List<XElement>();  
            lst.AddRange(definition.Descendants("IPModel").Distinct().ToList());  
            lst.AddRange(definition.Descendants("IRModel").Distinct());  
            return lst;  
          }  
        else  
          return null;  
      }  
    }  
    // Render visual output based on IPModel  
    public static IVisual VisualOutput(XElement definition)  
    {  
      CavComponentType IPModelType =  
        (CavComponentType)XDocument.Parse(typeof(IPModelType).CreateInstance()).Element("IPDefinition").Attribute("IPModelType").Value;  
      IVisual visualOutput = null;  
      switch (IPModelType)  
      {  
        case CavComponentType.IP_DeepZoom:  
          visualOutput = new  
          IP_DeepZoom(definition);  
          break;  
        case CavComponentType.IP_SemanticZoom:  
          visualOutput = new  
          IP_SemanticZoom(definition);  
          break;  
        case CavComponentType.IP_SpatialSemanticZoom:  
          visualOutput = new  
          IP_SpatialSemanticZoom(definition);  
          break;  
        case CavComponentType.IP_Lifecycle:  
          visualOutput = new  
          IP_Lifecycle(definition);  
          break;  
        case CavComponentType.IP_Department:  
          visualOutput = new  
          IP_Department(definition);  
          break;  
        default:  
          visualOutput = null;  
          break;  
      }  
      return visualOutput;  
    }  
    #region DataOutput  
    // Extract IGModels of all IRModels in use  
    public static XElement DataOutput(XElement IPDefinition)  
    {  
      XElement xml = new XElement("DataOutput");  
      if (IPDefinition.Element("IRModels").Descendants("IRModel").Any())  
        foreach (XElement IRDefinition in IPDefinition.Element("IRModels").Descendants("IRModel").Distinct())  
          xml.Add(new XElement(IRDefinition.Element("DataOutput").Descendants("IRDefinition").Any()));  
      else  
        foreach (XElement IRDefinition in IPDefinition.Element("IRModels").Descendants("IRModel").Distinct())  
          xml.Add(new XElement(IRDefinition.Element("DataOutput").Descendants("IRDefinition").Any()));  
      return xml;  
    }  
    #endregion  
    #endregion  
    #endregion  
}
F.4 Scenario Pool

The Scenario Pool stores information generation scenarios, information representation scenarios, information presentation scenarios, and visual scenarios. The key class supporting the Scenario Pool at the resource layer of the purposeful visualization system prototype is presented in Figure F.6.

![Class diagram of IG, IR, IP and visual scenarios](image)

The essential code snippets implementing the above class are shown as follows:

```csharp
using System.Windows.Controls;
using System.Windows;
using System.Xml.Linq;
using System;
using System.Linq;
namespace CAVE_VS
{
    public partial class CavVisualScenario : UserControl, ICavComponent, IVisual, IContext
    {
        #region Properties
        public string ComponentName
        {
            get { return this.lblVisualScenarioName.Text; }
            set { this.lblVisualScenarioName.Text = value; }
        }
        public string ComponentSettingFile
        {
            get; set;
        }
        public CavComponentType ComponentType
        {
            get { return CavComponentType.VisualScenario; }
        }
        public XElement Input
        {
            get; set;
        }
        public XElement Output
        {
            get; set;
        }
        public UIElement Preview
        {
            get { return this.vbPreview; }
        }
        public bool IsInstantiated()
        { return this.Output != null; }
        public void ShowComponentSetting()
        {
            try
            {
                VisualScenarioDesigner scenarioDesigner = new VisualScenarioDesigner();
                scenarioDesigner.txtScenarioName.Text = this.ComponentName;
                scenarioDesigner.txtScenarioDescription.Text = this.Description;
                //Initialize context definition
            }
            catch (Exception)
            {
                //Handle exception
            }
        }
    }
}
```
if (this.Context.ContextDefinition != null)
    scenarioDesigner.Context = this.Context;
if (IsInstantiated())
    scenarioDesigner.RenderVisualOutput(this.Output);
    scenarioDesigner.Closing += (s, a) =>
    {
        this.ComponentName = scenarioDesigner.txtScenarioName.Text;
        this.Description = scenarioDesigner.txtScenarioDescription.Text;
        this.Output = scenarioDesigner.XML();
        // Update preview
        ShowDetailView();
        this.vbPreview.Child = VisualRepresentation();
    }
    scenarioDesigner.Show();

    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
public void ShowOverview()
{
    this-previewContainer.Visibility = Visibility.Collapsed;
    public void ShowOverview()
    {
        this.previewContainer.Visibility = Visibility.Visible;
    }
}
public UIElement VisualRepresentation()
{
    if (this.Output != null)
    {
        DesignerCanvas canvas = new DesignerCanvas();
        IP_DeepZoom.RenderVisualOutput(this.Output, canvas);
    }
    else
    return null;
    }
public UIElement VisualPresentation()
{
    return this.VisualRepresentation();
}
#endregion IVisual
#region IVisual
public void ApplyContext(XElement ContextDefinition)
{
    if (this.Output != null)
    {
        this.Output = Representation.ContextualizeModelDefinition(ContextDefinition, this.Output);
        this.ShowDetailView();
        this.vbPreview.Child = this.visualRepresentation();
    }
    }
#endregion IContext
#region Constructors
// Initiate a visual scenario based on its definition
public CavVisualScenario(XElement VSDefinition)
{
    InitializeComponent();
    this.ComponentName = VSDefinition.Attribute("Name").Value;
    // Initialize context
    if (VSDefinition.Elements("Contexts").Any())
        this.Context = new Context(VSDefinition.Element("Contexts"));
    else
        this.Context = new Context();
    this.Input = this.Output = VSDefinition;
    ShowDetailView();
    this.vbPreview.Child = this.VisualPresentation();
}
public CavVisualScenario()
{
    InitializeComponent();
    this.ComponentName = "Visual Scenario";
    // Initialize context
    this.Context = new Context();
}
#endregion Constructors
REFERENCES


NZ Chapter of the ACM's Special interest Group on Human-Computer interaction (pp. 93-100). ACM, New York, USA.


