Improving Visualisation by Capturing Domain Knowledge

Jonathan Meddes and Eric McKenzie

School of Computer Science, Division of Informatics,
The University of Edinburgh, James Clerk Maxwell Building,
The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ.
Tel: +44 131 650 5129 Fax: +44 131 667 7209
jmx|ram@dcs.ed.ac.uk

Abstract

An effective visualisation system depends on a user's ability to interpret a visual representation and make valid inferences. This paper first summarises the role of domain knowledge when interpreting a visualisation. Once the visual perception system has interpreted the visual representation, the user transforms the data into information by the introduction of domain knowledge; these are the rules or items of knowledge that are relevant to this visual representation allowing the user to make meaningful inferences.

In the remainder of the paper we concentrate on a visualisation architecture that encapsulates domain knowledge to improve user interpretation of a visual representation. We use an agent-based paradigm to provide a distributed model of computation which moves away from a heavyweight constrained based algorithm towards a lightweight distributed system that empower individual data items.

Finally, we present DIME (Distributed Information in a Multi-agent Environment), an implementation based on this approach. DIME is an ongoing research project that tightly integrates data storage, knowledge capture, and information visualisation in a 'visual environment'.

Keywords: visualisation, agent, domain knowledge, visualisation framework, user interface.

1 Introduction

Advances in the sophistication of storage devices have allowed application developers to store data that is more detailed, but to truly harness its potential users need access to this data. In many organisations traditional business processes have been replaced by information technology. Managers realise they can use this data and have the opportunity to have more control of their organisation. From a manager's perspective, this has the potential to be an *information revolution*, and can satisfy their need for enhanced control. Unfortunately, this revolution cannot take place until the information is presented in a readily accessible form.

An effective visualisation uses computer graphics to allow the user to understand data, but without utilising the users domain knowledge, information visualisation is not a realistic proposition. We present a model that captures the interaction between a visualisation system and the user in a *visualisation framework*. This identifies the user's domain knowledge as a crucial component of visualisation process allowing the user to transform data into information.

Using the visualisation framework as a reference, we describe an information visualisation architecture using agents. This captures the user's domain knowledge within the visualisation system encouraging knowledge dissemination between users in a system that integrates data storage and information visualisation. Finally, we introduce DIME (Distributed Information in a Multi-agent Environment), an ongoing research project and outline our plans for its future development.

2 Visualisation framework

The visualisation framework identifies the significant components of the visualisation process. That is, the task of creating a visual representation of a data series for the user to derive an understanding of the data. It is particularly important to understand the interface between the human and the visualisation system to accurately identify limitations exhibited by a visual representation.

The aim of adopting a visualisation framework is to identify the significant components of the visualisation process in the belief that "underlying the concept of visualisation is the idea that an observer can build a mental model, the visual attributes of which represent data attributes in a definable manner" [12].

The visualisation framework consists of two primary components, the *visualisation system* and the *user*. These are shown in figure 1 and described in the following sections. The interaction between these components determines how successful the user will be at understanding the data.

We are using a visual representation to convey the information contained in a database. It is undesirable for anomalies to be introduced into the users understanding of the data. Such anomalies reduce the effectiveness of a visual representation by adversely affecting the accuracy of information interpreted from the visual representation. A successful visualisation will construct a user interpreted data view that will have an *isomorphic* relation with the original data. If such a relation is present, the user will have an accurate understanding of the original data input into the visualisation pipeline. In a less successful visualisation, the user will have only a partial understanding of the data and only selected fragments of the user interpreted data view will be related to the original data. In extreme cases when the visual representation is incomprehensible to the user, the user interpreted data view may bear no relation to the original data.

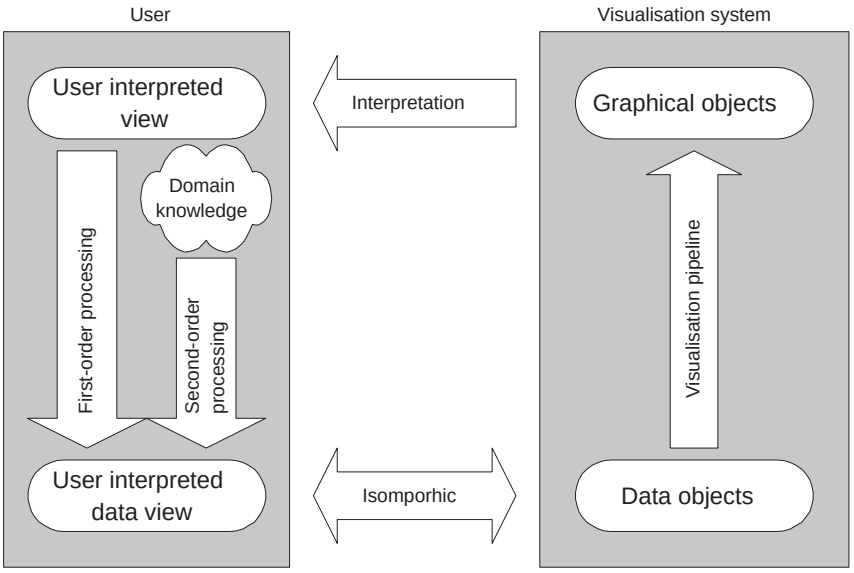

Figure 1: Visualisation framework

2.1 The visualisation system

The visualisation system uses a combination of software and hardware to provide the framework with an implementation of the *visualisation pipeline*. This pipeline is a crucial component of the visualisation system that is responsible for the flow of control from the input data source to its visual representation.

The pipeline commences with the *data objects* stage that represents the raw data as it is extracted from a data source. At this stage, the data is just a collection of values and relationships in a form that is not particularly useful to the remainder of the pipeline.

A *data classification scheme* provides a characterisation of the data source by identifying data items and their attributes, transforming them into a more concrete model for use by subsequent stages of the pipeline. In [11], the

authors describe the data classification scheme used in SAGE that is capable of describing the structure of relational data. This scheme is a somewhat more comprehensive to that used in APT [7], a similar automated presentation system which has a less complete set of characterisation constructs. Miss-classification or correct classification using an insufficiently expressive language is detrimental to the remaining stages of the pipeline, as creating a successful mapping between the data objects and the graphical objects depends on a complete understanding of the data.

Data classification is crucial to the success of a visualisation system. Simple data classification schemes, which while expressive, can present a naive encoding of the underlying semantics of the information. If combined with an automated presentation system this can lead to a visual representation that confuses visual readability with visual communication. This is not unlike the differences observed between novices and experts when constructing a visual representation. Novices are distracted by the syntax or surface features, experts are more concerned with the underlying semantic features and the available perceptual cues which can be embraced to communicate the data semantics [10].

Within the pipeline the data objects created by data classification are processed using a series of mapping rules. These rules identify features in the data to create an associated set of *graphical objects*. These objects are rendered by the visualisation system to create a visual representation for the data.

2.2 User

We represent the user in our framework using three components to capture the most relevant aspects to the human-computer interface. The user interprets a visual representation of data using their visual perception system; this creates in the user an *internal* representation of the *external* visual representation [13]. In the framework the internal representation is captured by the *user interpreted view* which can be considered as a simple raster encoding of the visual representation.

In terms of data content, the user interpreted view is meaningless and subsequent cognitive processes are required to extract details about the data. We separate the user processing of the interpreted view into two stages. *First-order processing* provides the high level analysis of the user interpreted view to create an abstract interpretation of the data; during *second-order processing* the users 'knowledge' is utilised to infer additional properties of the data. The processing transforms the user's interpreted view into a corresponding *user interpreted data view*, this is how the user understands the data encoded in the interpreted visual representation.

First-order processing

First-order processing allows the user to perceive basic structural properties about the data such as "data element A is associated with data element B". This is achieved by linking visual cues to aspects of the data [5]. This stage of the user interpretation is primarily concerned with processing basic perceptual elements.

Such basic perceptual elements and the process of linking them with dimensions of the data source are described as 'retinal techniques' by Bertin [2]. The use of perception has been the focus of much research and has created many guidelines, methods and frameworks that are intended to assist the visualisation creator construct visual representations. The elementary perceptual tasks identified for quantitative comparisons [3], and the encoding of data using colours [6][4] are examples of research which formalises the rule-of-thumb methods used by visualisation creators.

Second-order processing

Second order processing builds on first-order processing by enabling the user to identify from the perceptual cues, logical relationships between data items. The perceptual cues within the user-interpreted view identify patterns within the data. These patterns present the user with a high level understanding of the data encapsulated within the visual representation, but not the specific information that is its most valuable resource. The introduction of the user's domain knowledge introduces environmental influences into the interpretation of data; this *contextual information* is fundamental to transform data into information.

In many visual representations it is the *secondary notation* [10] which enables the user to extract the particular information they need. Secondary notation is not part of the elementary perceptual cues that are explicitly used to encode the data; rather, it invokes more subtle relationships between the data elements. An experienced visualisation creator uses secondary notation as a means of expressing additional semantics that might otherwise be omitted. This

(a) (b)

Figure 2: A visual representation of the 20 largest cities in the UK. The position of each city is mapped to its geographic position in the UK. (a) Each city is represented as an identical solid filled black circle. (b) Each city is represented as a solid filled black circle; its area is proportional to the population of the city.

can mean creating perceptual groupings which a logical rather than visual, creating an effective communication of the data which may not be the most aesthetically pleasing.

Domain knowledge allows the user to make logical groupings between data items that are not explicit in the visual representation. They use their knowledge of the semantics of the data contained within the visual representation to make additional inferences. To assist the user, *visual clues* can be encoded into the visual representation as hints for the user as to nature of the data. Such clues allow the user to place the data in context, invoking their domain knowledge in the interpretation process and leading to a more complete understanding of the data semantics and in turn the actual data. The deployment of visual clues should be viewed with caution as irreverence is as much a problem as relevance [1]; if a visual clue is visible it is assumed to be relevant to the visual representation and the data encoded within it. An example of visual clues is presented in the following section.

2.3 Assessing the role of domain knowledge

Domain knowledge plays a central role in the interpretation of a visual representation. To illustrate the application of domain knowledge on the interpretation by a user we describe the figures used in an associated experiment [9].

The experiment was conducted anomalously, figures 2 and 3 show the diagrams shown to the subject. Each figure was revealed to the subject one at a time for a period of exactly 30 seconds; the figures were exposed in the following order: figure 2(a), figure 2(b) and finally figure 3. At the end of the period the subject was asked to make a verbal interpretation of the visual representation and the data it represents.

All three figures were created from the same data source, the 20 cities with the largest population as recorded by the 1991 UK government census. In the figures, every city is represented by a glyph, the glyph used is a black circle. In the visual representation of the data, the glyphs are positioned in the correct geographic position as defined by the local coordinate system. In figure 2(a) the glyphs are a constant size as the only attribute mapped from the data source is geographic position. In addition to the geographic attribute, in figure 2(b) and figure 3 the glyphs have their size attribute mapped to the population of their associated city; the size of the glyph is defined as a linear relation between the population of the city and the area of the glyph.

Figure 2(a) is an example of a visual representation with clear perceptual cues. The glyphs are associated with some data set, but there are insufficient visual clues for a subject to induce their domain knowledge.

In figure 2(b), the glyph's size is proportional to the population of the city. Although the visual representation still has very few clues to invoke the subject's domain knowledge, the size of glyphs give a subject some clues about

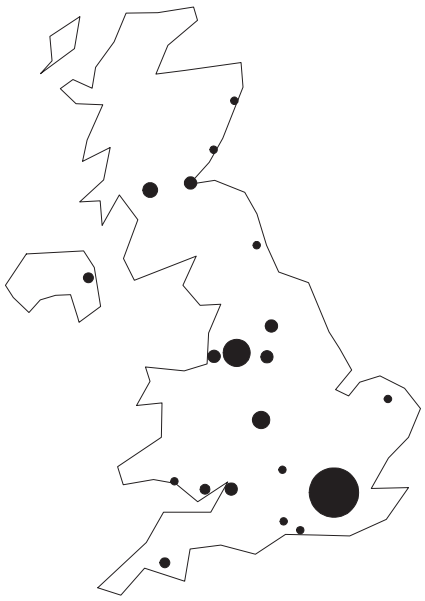

Figure 3: A visual representation of the 20 largest cities in the UK. Each city is represented as a solid filled black circle; its area is proportional to the population of the city. The position of the circle is mapped to the cities geographic position and overlaid on a map of the UK.

the data source. In particular, the three largest cities, London, Manchester and Birmingham, are strong clues to the data source. In addition, the cities in the extremities of the UK, Aberdeen in the North of Scotland, Exeter in South-West England, Norwich in the East of England and Belfast in Northern Ireland, give an approximate geographic shape to the visual representation and present the subject with visual clues to place data source in context.

Finally, figure 3 presents the glyphs in the *context* of an outline map of the UK. In this figure the subjects are prompted into associating the glyphs with their geographical domain knowledge, the map is a strong visual clue to the application. The visual clues to the data source are similar to the pervious figure, figure 2(b), with the strongest clue being the glyph associated with London; its area reflects the population of the UK capital being more than three times that of the second largest city.

The visual clues in figure 3 give the glyphs a strong association with cities, and where the subject has the appropriate domain knowledge of a cities population, the area is an additional strong visual clue. Interestingly, those subjects who were familiar with the source of funding for this work suggested the size of the glyph was connected to the number of telephone lines. This is not an entirely improbably interpretation of this visual representation and demonstrates the miss-application of domain knowledge. All subjects were in agreement that the glyphs in the figure 3 were geographically positioned.

In the following section, we describe the mechanisms provided in our visualisation architecture to incorporate domain knowledge.

3 Incorporating domain knowledge into a visualisation architecture

Data in the visualisation pipeline flows in one direction, from the database to its visual representation. Often, the visualisation pipeline transforms the data using a monolithic algorithm that detaches the visual representation from the originating data.

A visualisation architecture must address (1) how data is stored or represented within the system, (2) how visual representations of the data are created, and (3) how the operational requirements of (1) and (2) are co-ordinated.

The visualisation architecture [8] we have adopted is an agent-based system [14] inspired by behaviours observed in biological systems.

The architecture is specifically intended to integrate data storage with its visual representation. In the following section, we briefly describe the principles of agent architectures before going on to describe integrated data storage and visualisation. Finally, we describe the capture of domain knowledge and evolution within the agent environment.

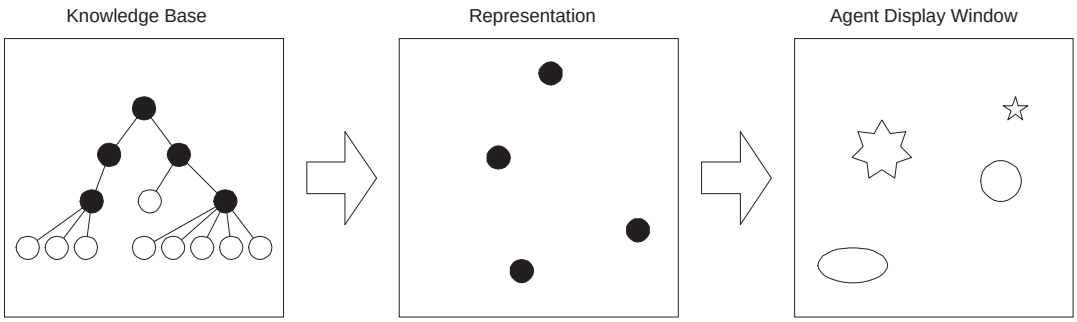

Figure 4: Abstract view of a visualisation architecture using agents showing the three components of the architecture. The knowledge base is used as a repository for the structure and semantics of the data. Representation agents use the knowledge base to create a visual representation of items within it. Finally, the agent display window is a specialist agent assigned the task of rendering the representations.

Organisation

The central feature of an agent architecture is the knowledge base. It can not only store the syntactic features of data, but also a semantic description. The semantic network comprises of concepts and features; a *concept* is used to describe similar elements from the source, the data elements are characterised by the features of a concept. Concepts are arranged in the semantic network with hierarchal structure, more specialist concepts approach the leaves of the tree. Items from the data source are represented within the knowledge base by an instantiation of the appropriate concept from the semantic network.

Agents can be separated into two basic types, (1) class agents and (2) data agents. Although every agent is equal and has equivalent functionality, this distinction assists the organisation of agents within their environment. A class agent introduces a dimension of grouping into the agents' organisation. The class structure is comprised of a hierarchical arrangement of class agents, each of which can be sub-classed to provide a more specialist grouping of like agents.

Figure 4 shows an abstract view of the agent architecture used for visualisation. Within the knowledge base describing the source data, concepts of the data are described using class agents and features of the concepts are described using their properties. Instances of the concepts are created using data agents.

Whereas a class agent provides the principle organisational features of the environment, a data agent (usually) contains an item of data that we wish to represent within the system. For example, a data agent may be introduced into the system that represents a numerical value; to do this the data agent will be assigned a *quantitative* property encapsulating the value.

Class agents can take a pro-active role in the life of its associated agents. For instance, an agent could be assigned a property that is responsible for calculating a total of all the numerical values represented by its associate agents. In this case, the agent would have a data value representing an abstract view of the data held by its associates.

3.1 Using agents for visualisation

Within the environment extensions allow agents to create their own visual representation. These extensions take the form of properties that take into account a suitable visual representation for the data. The philosophy behind using agents for visualisation stems from the devolving power to individual data agents. At the lowest level, individual

data agents are responsible for deciding how they are represented within the visual representation, and similarly, class agents are responsible for how groups of agents are represented.

Within our architecture, a single specialist agent is assigned the task of rendering the overall visual representation constructed by the agents in the environment, this agent provides the *agent display window*. The glyphs used by agents to represent the data can have varying degrees of complexity. For example, a glyph representing a single agent holding a numerical value might take the form of a coloured box with colour saturation indicating the agent's value. Alternatively, a class agent representing numerous data agents might use a more complex glyph; it could use colour saturation, size and may use numerous abstract visualisation objects to create a complex visual representation of the data.

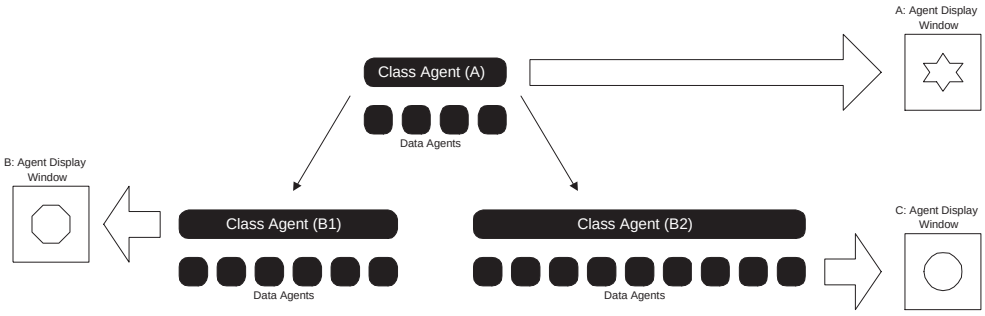

Figure 5: Agents provide different representations of themselves in the agent display window

Figure 5 shows a class hierarchy of three agent classes. Classes *B1* and *B2* are subclasses of class A, and every class agent has several items of data encapsulated in its data agents. Three agent display windows (A, B and C) show *potential* representations of the data from the class hierarchy. Agent display window C shows a glyph representing a single data item; the same agent that represents the data creates the glyph. Agent display window B shows a more complex glyph created by class agent *B1*; this glyph is a visual representation of all the data agents of the class. Agent display window A shows a complex glyph created by class A; it is a visual representation of the data encapsulated by all the data agents in classes A, *B1* and *B2*. In each element of the agent display window, it is the responsibility of the agent creating the glyph to decide on the most suitable visual representation for the data it represents. For example, when a combination of glyphs violates good visualisation practice, it is the responsibility of the agents to initiate a vote to resolve the conflict.

3.2 Evolution of knowledge within an agent environment

A fundamental feature of an agent system is the evolution of agents as they react to environmental influences. A change in the value or state of a property usually ultimately affects the behaviour exhibited by the agent. The reaction to changes in an agent's internal characteristics also influences the behaviour of other agents as they react to changes in their environments; in turn other agents react to these changes and so on. This chain reaction ensures that evolution is not localised to an individual agent or group of agents; instead, evolution propagates throughout the agent community leading to a constantly evolving environment.

In the visualisation framework, we identified the important role of domain knowledge when a user transforms data into information. Within our agent environment, we replicate domain knowledge by assigning specialist properties to an agent. These properties perform simple processing tasks that mimic the domain knowledge of the user.

Agents provide convenient facilities to introduce domain knowledge into the agent environment. Throughout the lifetime of the agent, it is aware of the knowledge it possesses and attempts to influence the environment to reflect its knowledge. Agents create relationships and introduce additional agents into the environment to reflect a belief of the agent. As time progresses, a network of relationships and agents are introduced to reflect the domain knowledge encapsulated within the environment. The same principles can be used to expand the collective 'knowledge' by assigning an agent the task of seeking data from outside their immediate environment. This task further extends the dissemination of knowledge.

Using this technique, agents can acquire data from a variety of external sources. Specific agents are assigned the task of investigating an external data source for relevant data; this is a characteristic of a specialist property. Once the agent has located some relevant data, it can populate the environment with additional agents to reflect the results of its investigation. The agent continuously monitors the external data for changes and reflects these changes within its own environment.

Usually, an agent only communicates with other agents in its environment. In common with all other characteristics of an agent, the ability to acquire data from an external source is captured by a property held by the agent. Such a property provides the agent with a portal connection to an external source.

Creating a network in this way provides the opportunity for identifying 'creative relationships' to be identified as agents use their domain knowledge. Evolution when driven by domain knowledge extends the system beyond a facility for data storage into a *knowledge repository*.

4 DIME visualisation system

The agent architecture described in the previous section has provided inspiration for DIME, Distributed Information in a Multi-agent Environment. DIME is an ongoing research project investigating the use of an agent-based architecture for information visualisation and storage. The DIME visualisation system is implemented in Java using the Java 3D API.

The agent population is described using the Data Specification Language (DSL). The DSL provides DIME with a description of how the agents will be organised within the environment. Once parsed, the agents can be activated; this allows agents to perform their tasks.

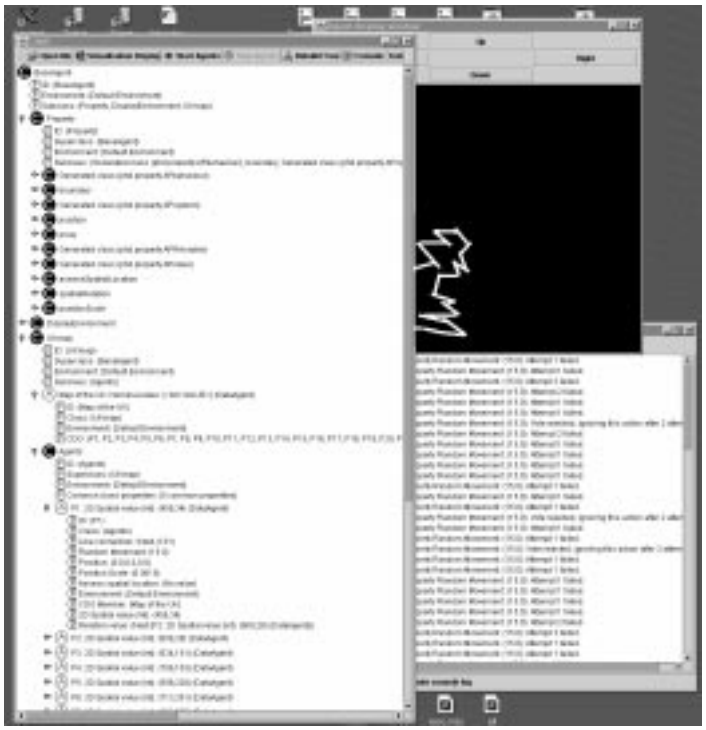

Figure 6: Screen-shot of the DIME visualisation system

Figure 6 shows a screen-shot of the DIME visualisation system. The window on the left is the agent browser that slightly obscures the agent display window in the top right of the screen; it shows a partially constructed map of the UK that is being created by the collaboration of many agents. Finally, the console window shows messages from agents. The *agent browser* allows the user to browse the class agent hierarchy; class agents can be expanded

to inspect individual data agents. The agent browser provides an interface to the agent's properties that define its character and suggests an intuitive metaphor with the social groups we are surrounded with in everyday life.

5 Conclusion and future work

Domain knowledge plays a crucial role in the users successful understanding of information contained in a visual representation. Without domain knowledge, the user cannot give context to the data and the visual representation is flawed.

In an attempt to better understand the process of information visualisation, we have developed a framework that provides a focus for the investigation of domain knowledge and the benefits it brings to information visualisation. The framework captures the significant elements of the visualisation process with the ultimate goal being an isomorphic relation between the original data (from the database) and the users overall understanding of the data.

The framework has served as a driving force for the development of a visualisation architecture using an agent paradigm allowing for a distributed approach to visualisation. A fundamental feature of this architecture is the long-term capture of the users domain knowledge in the character of agents. This aids the dissemination of knowledge between users and provides an agent environment that serves as a knowledge repository.

Distributed Information in a Multi-agent Environment (DIME) is a visualisation system using an agent-based architecture. In this system, data storage is closely coupled with information visualisation. DIME provides an interface to the agent environment that is specifically intended as a management information system tool. Users of these systems are typically senior members of an organisation who are technically competent and have the ability to manage people. Such organisational skills can be utilised to organise data into groups and assign tasks to clusters of data. This provides an interesting avenue of interaction that leads to the wider acceptance of management information systems and the introduction of creative and social interaction within information visualisation.

Acknowledgements

This research is supported by BT Laboratories, UK.

References

[1] Anzai, Y.: Learning and diagram-drawing and graphical reading skills: analysis and design. Presented to Workshop on Graphical Representation, Reasoning, and Communication. AI-ED '93. August 1993, Edinburgh, UK.

[2] Bertin, J.: Semiology of graphics. University of Wisconsin Press, London, England, 1983.

[3] Cleveland, W., McGill, R.: Graphical perception: Theory, experimentation and application th the development of graphical methods. Journal of the American Statistical Association, 1984, Vol. 79, pp. 531-554.

[4] Healey, C.: Choosing effective colors for data visualisation. IEEE Visualisation 1996.

[5] Keller, P., Keller, M.: Visual Cues. IEEE Computer Society Press. 1993

[6] Levkowitz, H.: Color scales for image data. IEEE Computer Graphics and Applications. January 1992, Vol. 12, pp. 72-80.

[7] Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Transactions on Graphics. Vol. 5, No. 2, April 1986 110-141.

[8] Meddes, J., McKenzie, E.: An agent-based visualisation architecture. Third International Conference, VISUAL '99 Amsterdam, The Netherlands, June 1999 Proceedings.

[9] Meddes, J.: Domain Knowledge Experiment Protocol and Results. Informatics Technical Report Series, University of Edinburgh, 2000.

[10] Petre, M.: Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming. Communication of the ACM. June 1995, Vol. 38, No. 6.

[11] Roth, S., Mattis, J.: Data Characterization For Intelligent Graphics Presentation. CHI'90 Proceedings.

[12] Robertson, P.K.: A methodology for choosing data representations. IEEE Computer Graphics and Applications. Vol. 11, Issue 3, May 1991, pp. 56-67.

[13] Scaife, M., Rogers, Y.: External cognition: how do graphical representations work? Int. Journal of human-Computer Studies (1996) 45, pp. 185-213.

[14] Tecuci, G.: Building Intelligent Agents: an apprenticeship multistrategy learning theory, methodology, tool and case studies. Academic Press, 1998, London.