An agent-based visualisation architecture

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. Advances in storage technology have led to a dramatic increase in the volume of data stored on computer systems. Using a generic visualisation framework, we have developed a data classification scheme that creates an object-orientated data model. This data model is used in an agent-based visualisation architecture to address the problems of data management and visual representations. As part of an ongoing research project we have implemented DIME (Distributed Information in a Multi-agent Environment), a visualisation system based on this approach. *

1 Introduction

Computer science is catching up with visualisation, and there is a substantial motivation for computer science to make the effort. We strive to make data more interpretable via a visual representation but visualisation did not start with computers. Bertin's seminal work with graphical representations [1] is still a widely acknowledged guide to good visualisation practice and predates the widespread introduction of bitmapped displays.

The increased storage capacity of computer systems has allowed application developers to retain more detailed data. Managers have realised that they can use this data to have more control of their organisation. From a managers 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 readily accessible.

An effective visualisation uses computer graphics that allows the user to understand data. This paper presents an architecture for a novel approach to visualisation and provides an extensible visualisation system with which to develop our ideas.

We have based our research around a generic visualisation framework in which we describe a data classification scheme used to create a data model that accurately classifies the data. The organisation of the data model directly influences the internal operation of our visualisation architecture which is developed using a technique allowing distributed communication orientated visualisation. Finally, we describe the implementation of the visualisation system.

* This research is supported by BT Laboratories.

2 Data classification

Data classification is the process of identifying data items, structures and relationships within a data series. The result is a data model that can be used by the remaining stages of the visualisation pipeline. Automated systems including SAGE [7] and ANDD [6] emphasised that automated visualisation can only be successful if the system understands the data. The accurate classification of data and relationships using a suitably expressive scheme is crucial to a visualisation system capable of creating effective visualisations. Anomalies introduced at this stage are propagated down the visualisation pipeline; this has the potential to introduce spurious visual cues into a visual representation.

Before we describe the main features of our data classification scheme, we shall identify the desirable properties it should exhibit. The data classification scheme attempts to create a model that captures the syntax and semantics of the data. The syntax of data is its raw features that define its structural properties, whereas the semantics of data is the implied meaning projected by the organisation of individual data items. The data classification scheme has two goals: (1) consistently encode the data using a set of well-defined data types so that subsequent stages of the pipeline receive the data model in a familiar format; and (2) capture relationships between data items and allow easy extensions to the model by introducing additional data items and relationships.

For illustration purposes, we present a subset of our full data classification scheme that is specifically intended for the classification of geographic and topological based data. An application that generates this type of data is the analysis of wide-area-network traffic; cities have a geographic location, network links have a topological structure and traffic can be measured using a quantitative value.

The data model created by our scheme uses an object-orientated paradigm; this is a convenient method of organising the data that simplifies extending the data classes. Every data item is represented in our data model by a data object that is an instantiation of a data class. The basic data classes that are available to the data model are *nominal*, *quantitative*, *spatial* and *relational*.

In a *nominal* data class, each member is a data object that represents a textual item, e.g., name.

A *quantitative* class contains members representing numerical values, e.g., age. Constrained subclasses provide support for more specialist values such as currency and percentage.

The *spatial* data class has members that represent a location in space. In its basic form the spatial data class is abstract and cannot be instantiated, but the creation of subclasses provides support for two-dimensional, three-dimensional and longitudinal/latitudinal co-ordinates.

The *relational* data class provides the data model with a versatile method of representing relationships. Instantiation of this class creates a data object containing references to its associated objects; the data class also contains references to all the classes of objects which are represented in its relations. Subclasses such as the *Complex Data Class* (CDC) provide increased specialisation by restricting the types of relations which can be made. The CDC defines a signature consisting of a collection of data classes. Each instantiation creates a *Complex*

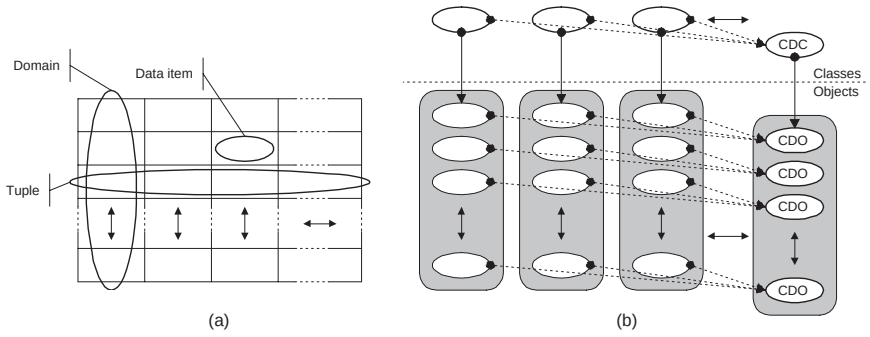

Fig. 1. (a) relational table and (b) its data model

Data Object (CDO) and must have exactly the same combination of data classes defining their relationships.

Using these basic data classes, a data model that accurately represents the input data is constructed using the data classification scheme. Figure 1 shows an external relational database table (a) translated into the classes and objects of the internal data model (b). In the table, every cell contains a data item, every item from a column is from a similar *domain* and the data items from a row constitute a tuple. In the data model, a data class captures the domain which is instantiated to describe its data items. A CDC is defined to represent the association between columns in the table. Instantiations create a CDO that captures the association between the individual data items of a tuple.

The data contained in the data model is merely a collection of data values and relations. In the absence of further detail describing the contents of the data model, it is meaningless as a self-contained information source. Data is transferred into information by the introduction of *context*. Typically, context is introduced by the user when they extract data from a model by introducing environmental influences from their *domain knowledge*. To move the data model from a simple collection of data values and relationships we introduce *domain functions* to provide context and replicate the environmental influences that dictate how a user changes data to information. In the data model, a domain function provides a basic service that can manipulate the data by mirroring the simple processing and reasoning a user performs when they interpret some data. For example, if we consider the *use* and *capacity* of a network *link*, a typical user of that data can deduce that the average *load* of that link is represented by *use*/*capacity*; a domain function would introduce new load objects to the data model.

3 Visualisation architecture

The visualisation framework allows us to develop visualisation ideas in an environment where we are not constrained by a specific visualisation architecture.

If we want to create tangible visual representations, we require a visualisation architecture to realise the visualisation system. In the remainder of this paper, we describe a visualisation architecture that demonstrates a novel approach to implementing the visualisation pipeline.

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 we have adopted is an agent based-system inspired by behaviours observed in biological systems.

3.1 Principles of an agent based architecture

The simplest way of illustrating the underlying principles behind an agent-based system is to use a biological example. If we consider a room with no windows or doors, the room is essentially a box. Although the room is sparse, it does contain some objects, a telephone, a chair, a blackboard and a stick of chalk.

The room is populated by five people; each with a personality that characterises their behaviour. For example, one could posses leadership qualities that leads them to command others, or have organisational skills that makes them request actions from others, or might pace around the room. The person exhibiting the latter behaviour is physically constrained by the boundaries of the room. If they do not rely on the walls to restrict their movement, a cognitive process must be present that not only makes them move but also controls their movement.

The inhabitants of the room can communicate using a commonly understood language. The person making the utterance could decide to communicate with another individual, a group of people (e.g., all males or some other commonly accepted group), or broadcast their utterance to everyone. This communication could transfer information, issue instructions (or orders) or ask and respond to questions. It would be fair to assume the people in the room would observe Grice's *maxims of communication* [4]. The adoption of these rules means that the people in the room communicate effectively and will not attempt to mislead one another, maliciously or otherwise.

A person's environment is defined by the room and the objects within it. The room is a static boundary that restricts the movement of people and objects. The chair is a passive object that may be used by the people in the room but offers the bear minimum of interaction. Similarly, the blackboard provides a service to the people in the room, but could also be used for storage or as a device to communicate with other people. The telephone is an interesting object; it provides a service allowing people to communicate outside their immediate environment and can affect the people within the environment when it receives an incoming call. In this case the telephone is no longer a passive object that provides a service, it can cause the people in the room to change their behaviour.

A simple example will provide an illustration of the types of communication that could take place in the room. It demonstrates the principle of co-operation that is essential for social groups to achieve a common goal. In this example, we want the names of the people in the room to be written on the blackboard. The

task is initiated by a telephone call that is answered by a person who knows how to use a receiver; the caller issues an unambiguous instruction to "write every persons name on the blackboard". The person who answered the call is now aware of the task that must be achieved and they take a controlling position in the environment. Luckily, the person who answers the call also has the ability to write on the blackboard and broadcasts a message to all the other people in the room asking them to reply with their name. One by one, the other people in the room reply to the question and as the answers are received they are written on the blackboard. Finally, the task is completed by adding their own name to the bottom of the list. This example demonstrates that some tasks can only be achieved by interaction and co-operation.

This distributed computation, co-operation and communication which is evident in the biological example above demonstrates exactly the principles which underpin an agent-based system. In such a system, every person and object is represented as an agent. Agents in the environment have a similar existence to the people in the room; they are autonomous entities but are capable of communication. Through communication, agents can arrange co-operation to achieve common goals. Agents can co-exist in an environment that has a similar constraining effect to the room. The environment also provides a transport mechanism for inter-agent communication; this is analogous to the longitudinal sound waves that are transmitted through the air of the room.

We have used this distributed model of computation to create DIME (Distributed Information in a Multi-agent Environment), a visualisation system that moves away from a heavyweight constraint based optimisation algorithm towards a lightweight distributed system that empowers individual data items. The agent environment is populated directly from the data model created by the data classification scheme. The object-orientated nature of the data model easily translates into the agent environment by allowing *every* data object and class to be represented as an agent. Using this approach provides a convenient method of organising the data within the visualisation system; it also allows the responsibility for creating a visual representation of the data to be devolved to the individual data items and their associates.

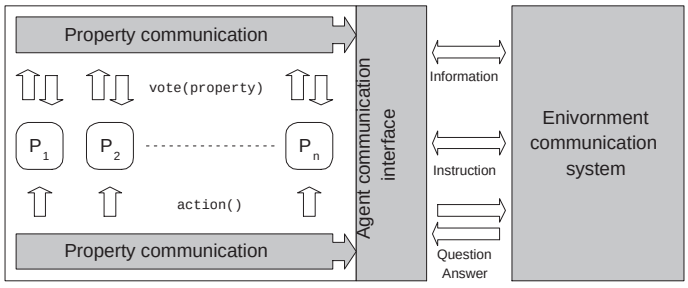

Fig. 2. Architecture of an individual DIME agent with properties (P_1 to Pn) and the communication interface with the agent environment.

3.2 DIME agent environment

Using an agent-based philosophy, DIME supports agents operating within an agent environment. We first describe the construction of an agent before describing the operation of multiple agents within the environment; the discussion of our system is presented with a minimum of implementation detail.

DIME agent: Figure 2 shows the structure of an agent which is characterised by the properties it holds (P_1 to P_n). Typically an agent is assigned numerous properties and the interaction between the properties dictates its overall behaviour. Properties can be broadly cast into two types, (1) a value property stores details about the agent, and (2) a behaviour property directly describes how the agent behaves within the environment.

The internal operation of an agent is controlled by its *property communication interface*. This is responsible for controlling the individual properties of an agent, organising the updates of each property via a voting mechanism, inter-property communication, and providing access to the *agent communication interface* for inter-agent communication. When an agent is introduced into the environment, it is activated to perform its tasks. Once activated, an agent uses a round-robin algorithm allowing each property to optionally perform some action. The combined effect of these actions allows agents to complete their goals. The action of each property is defined within its action method which is invoked each time it is the property's turn to do some work. This method allows the property to make changes to the character (or state) of the agent by updating or adding properties. Properties in control of the agent are co-operative and operate in a fair and reasonable manner; this is particularly important when a property needs to update the values of other properties in the agent. As a safeguard to an agent being dominated by a minority of properties, we use a voting mechanism in an attempt to reach a consensus on the evolution of the agent. The voting mechanism allows all other properties to cast a vote on their support for the agent to adopt the new property value. The support a property can express for new values ranges from definitely reject to definitely accept, with a neutral position being taken by a property with no interest in the vote. If the outcome of the vote is marginal, a random function makes the casting vote.

The operation of an agent is clearly illustrated by a simple example; we limit our consideration to a single agent defined to randomly move around a limited area and only consider its directly relevant properties. This behaviour is expressed using three properties, (1) the *position* property stores the agents current position in its environment; (2) the *boundary* property represents the limits of the agents movement; and (3) the *random movement* property defines the agents behaviour in its environment. Only the random movement property has an action method, the other two characterise the agent but have a passive role in its behaviour. When the random movement property wishes to move the agent, it uses the inter-property communication interface to request the current position of the agent; this is the value of the position property. It can then make a change to the position to represent a random movement in the environment. Before this change can be adopted, a vote must take place involving the position

and the boundary properties. The position property has no preference in the outcome of the vote and returns a neutral verdict. The boundary property must investigate the proposed change in detail. If the proposed position is outside the boundary, it rejects the change; otherwise, it votes to accept the change and the property communication interface will permit the new value to be adopted.

DIME environment: Agents do not directly have access to other agents internal properties and all contact must be channelled through the agent communication interface. This interface provides the necessary functionality to direct messages to individual agents, groups of agents (e.g., all the agents in a class or associated by a relationship agent) or all agents within the environment. Using this communication interface, agents can pass properties to one another using the following styles of communication, (1) *instruction* messages pass instructions to an agent which must be followed; (2) *information* messages inform other agents of a property value; and (3) *question* messages request information from an agent which are responded to using an *answer* message. Messages are routed by the environment communication system and stored until required by the agents.

The agent environment supports the introduction of domain knowledge via domain functions. These specialist agents have two roles, (1) to search for combinations of data agents that can be used to create new agents; and (2) to co-ordinate groups of agents. In the latter role, domain function agents act as a proxy for other agents in the environment. In this guise, the agents for whom it is acting as a proxy are assigned a *proxy* property; this refers all action and voting to the proxy agent. This mechanism is available to an agent that co-ordinates groups of agents when a substantial influence on their conduct is required.

The structure of the data model provides a natural organisation of the data within the visualisation system. Agents have a more complete understanding of the data than any other entity and can create visual representations of themselves using a special agent providing an *agent display window*. Agents that wish to be represented must posses the *visible* property; this is responsible for communicating visual information about the agent to the agent display window and encapsulates all the visual information about how to render the agent. This information is received from other properties and can be influenced directly or indirectly by other agents in the environment. Specialist properties can provide the agent with additional knowledge of a suitable visual representation for the data it represents. For example, a Gestalt property gives an agent knowledge about Gestalt principles [5] of appropriate organisation. Another property could provide the agent with knowledge of colour perception [2]. Using such properties, the agents have an improved knowledge of a suitable representation for the data.

Agents can collaborate to represent themselves as one visual item in the agent display window. The data model provides several inherent organisational features that are retained within the agent environment. The class agents and their associated data members provide two extreme levels of abstraction. At the most detailed level, agents can provide a visual representation in the agent display window or alternatively the data agents can provide a high level abstraction of the data. A CDO provides an orthogonal dimension of abstraction where

related data items provide an associated concept. In a data model created by the data classification scheme the highest level of abstraction is a complex data class. A CDC agent is an abstract view derived from a table representing all the class agents, data agents and CDO's. The autonomous but collaborative nature of agents allows them to negotiate suitable visual representations for the natural structures present in the data they represent.

4 Current work and future plans

The agent visualisation architecture has formed the basis for DIME, an ongoing research project implemented using Java and the Java 3D API which has allowed the rapid development of prototype agents and provides a stable platform to introduce additional functionality. DIME provides an interface to the agent environment allowing a user to browse the agent class hierarchy and expand individual classes to study its data agent instantiations. Properties of agents can be investigated and where necessary the characteristics of an agent can be changed by removing, adding or changing its properties. Visible agents are rendered in the agent environment window by the agent rendering system.

At present visualisations in DIME use graphical style sheets (GSS) [3]. A GSS specifies the properties that should be assigned to the agents for an effective visual representation of the data they represent. A future enhancement will introduce dynamic property allocation by introducing user profile agents that characterise a user. Such agents are assigned the task of identifying data that is relevant to the user; the agents representing this data are given greater prominence in the visual representation.

A long term goal for the development of DIME is to create a system capable of acting as a data repository. Our ambition is for the data management to be controlled by agents encouraging a long-term data management strategy rather than a system for short-term visualisation.

References

1. Bertin, J.: Graphics and graphic information processing. Walter de Gruyter, Berlin, 1981.
2. Healey, C.: Choosing effective colours for data visualisation. IEEE Visualisation 1996.
3. Felciano, R., Altman, R.: Graphical Style Sheets: Towards Reusable Representations of Biomedical Graphics. Conference on Human Factors in Computer Systems 1998 (to appear).
4. Grice, H.: Logic and Conversation. In Cole P and Morgan J eds. Syntax and Semantics: Speech Acts. **3** (1975) 41-58. New York: Academic Press.
5. Koffka, K.: Principles of Gestalt Psychology. London New York: Kegan Paul, Trench, Trubner, : Harcourt, Brace. 1935.
6. Marks, J.: A Formal Specification Scheme for Network Diagrams That facilitates Automated Design. Journal of Visual Languages and Computing. **2** (1991) 395-414
7. Roth, S., Mattis, J.: Data Characterization For Intelligent Graphics Presentation. CHI'90 Proceedings.