A Coordination Model for Exploratory Multi-View Visualization

Nadia Boukhelifa, Jonathan C. Roberts, Peter J. Rodgers
Computing Laboratory, University of Kent
{n.boukhelifa, j.c.roberts, p.j.rodgers}@kent.ac.uk

Abstract

In this paper, we present a coordination model for exploratory multi-view visualization. We base our work on current research in exploratory visualization and other disciplines. Our model is based on sharing abstract objects such as the visualization parameters of the dataflow model to achieve coordinated exploratory tasks in multiple views. This model describes how current coordinations in exploratory visualization work and allows novel coordinations to be constructed.

Keywords: **Multiple views, exploratory visualization, reference model, coordination, coordination objects, coupling.**

1. Introduction

Multiple views are prevalent in many user interfaces. This is motivated by real-life needs for simultaneous display of multiform data, rapid information processing and the necessity to compare and contrast different aspects of the data. For example, interactive television allows the viewing of one program while the user browses additional and related information. Users are no longer passive. They are often actively involved in shaping multiple views of data through interactive exploration.

Multiple views come in different forms. They could be abstract views that facilitate information processing, hierarchical and time series views, which describe the history of exploration or side thumbnail views for experimentation and quick viewing.

In many of these cases coordinating multiple views is beneficial. For instance selecting a group of data items in one view in coordination with the selection of the same items in another can show new relationships such as distribution, grouping or subordination within these items, which might otherwise remain unseen.

However, most coordination realizations in current visualization systems are last-minute ad-hoc additions. The coordination rules are informal, which may be flexible, but the principles would be difficult to export to other visualizations. Conversely, embedded coordinations are the result of adopting an underlying model. For instance the Snap-together conceptual model for coordination [1] relies on the relational schemata, so joins between relations are links between visualizations. While not wrong in itself, this specialization limits the potential and richness of coordination operations.

Moreover, no visualization reference models explicitly consider coordination. Even if coordination is contemplated from the design point of view, it is usually only regarded as part of the communication protocol and is generally dealt with within that restricted domain. There is a need for a flexible model for coordination that will ensure easy embedding of coordination in such exploratory environments.

The aim of this work is to first develop a reference model for coordination that will allow visualization designers to formally specify existing and novel coordinations in multiple-view exploratory visualization; second develop some rudiments of coordination drawing on the findings of the interdisciplinary study of coordination and lastly provide some examples of the model in use. In this paper we present such a model and formalize aspects of coordination for exploratory visualization (EV).

This paper is divided into the following sections (2) Related work. (3) Discussion of the salient features of coordination in EV. (4) The new model. (5) An extended example of application. (6) a discussion about the relationship of the model to current coordinated visualization systems. Finally, (7) future work and conclusions.

2. Related work

Coordination is a subject that has been investigated by many separate disciplines but only recently have researchers such as Olson et al [2] appreciated the obvious advantages of interdisciplinary viewpoints.

2.1 Interdisciplinary view of coordination

There are many disciplines that use related ideas that could be cross-fertilized into coordinated EV. For example, a translation framework for ontology mappings of information services, integration and unification of data, constraints management in concurrent design projects [3], inter-modality in different brain functions, redundancy for reinforcement of information, time series and history, middleware provision for cooperative work and balance and harmony in interior design. For these files, entities such as objects, events, processes, functions, agents and ontologies all may be coordinated.

Each discipline studying coordination has established its own architectures, models and protocols; some are based on sharing memory and other resources, others on managing constraints or propagating data values or data parameters. Coordination in these fields is treated in different contexts.

Many researchers consider the field of coordination theory to be the study of the interaction between processes. For them, coordination is an interoperability problem because the coordination brings together heterogeneous and distributed system components.

The multiple ontologies research, in particular, provides additional representation formalism defining the mapping between different ontologies where translation and other techniques such as approximation are required. For example, Akahani et al [4] suggest an approximate ontology translation framework for coordinating heterogeneous information services. The inter-ontology mappings of data aim to achieve data integration by identifying semantically corresponding terms of different source ontologies [5] where the semantic correspondence refers to equality or similarity.

Ciancarini et al [6] examine a suitable middleware for coordinating distributed active document-centric applications. This middleware is a software layer that abstracts from the heterogeneous characteristics of different architectures, operating systems, programming languages and networks in distributed systems. The main responsibility of the coordination middleware is one of data communication. Furthermore, middleware coordination often allows multiple views to occur in client-server architecture.

Another approach to coordination is the observation pattern as an ontology and a formal framework, as proposed by Viroli et al [7]. They write “in general, *observation* occurs when a system *o [observer]* is interested in some information made available by a system *s [source]*. Typically, ... *s* is modeling some portion of the world *o* is interested in, and providing *o* with some knowledge about it, as well as some mechanisms to access it”. The request-reply strategy of interaction in this paradigm works as a Model-View-Controller (MVC) pattern.

Although much research on ontology translation has used informal mapping rules between those ontologies [4], the interdisciplinary study of coordination presents ideas that can be used for modeling multi-view exploratory visualization. There is much communality between concepts from a variety of fields and a strong case for transference of ideas.

2.2 Coordination for EV

Interactive visualization is important, enabling the user to change the viewing parameters in one realization. Certainly the user can subsequently perform the same operation a number of times in additional views; but there is an obvious benefit in simultaneously coordinating the operation for the multiple views.

Thus, coordination can be described (as detailed by Olson et al [2]) as “composing purposeful actions into larger purposeful wholes”, where “the additional information processing performed when multiple, and connected actors pursue goals that a single actor [or indeed the multiple actors working separately] pursuing the same goals would not perform”. We emphasize the point that the whole is greater than the sum of its components.

The essence of exploration in visualization is Visual Information Seeking (VIS) indeed as Ahlberg and Shneiderman [8] state: the emphasis is on “rapid filtering, ... progressive refinement of search parameters, continuous reformulation of goals and visual scanning to identify results”. Furthermore, to achieve effective coordination in exploratory visualization there are many dependencies between views that need to be managed appropriately.

A model for coordination improves understanding, and allows effective development and qualitative evaluation of systems that incorporate coordination. The field of visualization is full of overloaded terms and suffers from inconsistencies. Moreover, visualizations often are based on different models. Coordinating different visualizations requires a mechanism, which allows interoperability between these differing models. We need a model for coordination to define, test and compare coordination strategies.

2.3 Current EV coordination models

There are many visualization systems that utilize coordination. For example, tools like Xmdvtool [9], Spotfire [10] implement coincident brushing operations; others like VIP, LinkWinds [11] and Visage [12] include coordinated 3D views. Many systems include linked overview-detail views which is highly utilized in geovisualization, for example see [13]. This fits with North and Shneiderman [14] dual *selection* and *navigation* motivation for coordination. However, like Pattison and Philips [15] we believe in a wider view of coordination, which potentially may coordinate any aspect such as data preparation, averaging, clustering, moving window positions etc. Recently two models have been proposed: the Snap conceptual model [1] and the View Coordination Architecture [15].

2.3.1 Snap

The Snap conceptual model takes a data-centric approach to coordination. Relational database components are tightly coupled such that an interaction with one component results in changes to other components. Snap utilizes the concept of database design to promote better visual exploration. It provides a mechanism for constructing coordinations without the need for programming. In addition, new types of coordination are introduced such as the compound join and the multiple alternative joins [1].

Snap's user interactions are currently limited to 'select' and 'load', whereas exploratory visualization supports much more interactions, which could also be coupled for coordination outside the relational database scope.

As we shall see, Snap resembles our model in many ways; for instance its architecture is event-based and coordination is built from action associations. Snap also recognizes the need for a middleware party to ensure coordination operation and for a translation mechanism when dealing with heterogeneous information sources. However, our model handles coordination from a more general viewpoint and takes in consideration exploratory visualization needs for rich and varied user interactions. Furthermore, we are interested in modeling representation-oriented coordinations as well as data-centric coordinations.

2.3.2 View coordination architecture for IV

Pattison et al [15] present an architecture for the implementation of generic view coordination in the Model-View-Controller (MVC) pattern. The proposed framework separates between the specification and implementation of mapping between data model to view model.

Coordination is managed by a new component (called *coordination*). Bidirectional coordinations can be achieved through directional coordination between presentation components, view model components or specification components. The more components there are and the more links exist between them, the more complex the implementation and debugging becomes, especially when linking different components [15]. Thus, to encourage reuse, presentation, content and the coordination itself should be - as far as possible - disparate and independent.

Rather than concentrating on the implementation architecture our work focuses on a layered approach based on the dataflow model. Like Pattison we use a MVC fundamental design, however we utilize multiple components and different facets of coordination.

3. Facets of coordination

From the related work and broadness of the interdisciplinary viewpoint we see that coordination conjures some interesting challenges, such as relevance, design and visual depiction of each coordination as well as considering what and how to coordinate.

3.1 Coordination challenges & opportunities

First, due to the multiform nature of the multiple views, actions in one view cannot always be directly applied to other views. For example, it may be useful to coincidentally rotate two three-dimensional views, but if each uses a different mathematical projection then a translation needs to occur that converts user interactions in one view to a suitable format for the other. However, some coordinations that may be possible to achieve, may in fact not be useful; and yet others may be impossible to realize. However, at this abstract level it is possible to rely on the user to make such judgment of the usefulness of a particular coordination. Indeed, there is the whole question of how the system is implemented and whether the coordinations automatically occur or are created by user requests [16].

Second, there are design and user-interface questions that a designer may wish to pose. For example, if the multiple views represent a visual-history then is it feasible or relevant to coordinate between past variances of the exploration? Moreover, is everything coordinated or are aspects of a few windows coordinated (and if so who decides on what is coordinated — the user or the system?[17]) For instance, if may be beneficial to only coordinate views that are classified within the same group (the notion of Render Groups [18]).

Third, how does the system visually represent and notify to the user what is currently being coordinated (e.g. visual methods such as used by the spiral calendar [19], or by the implicit laying out of modules in Waltz [18]). Many issues in visualization such as synchronization, correlation of visual or non-visual information, occlusion, view explosion and multitasking could be more approachable through a coordination model.

3.2 Coordination in use - two examples

In order to develop some rudiments of coordination in EV we investigate the use of coordination in a current tool (LinkWinds) and how coordination may be thought of as analogous to program variables.

LinkWinds uses a data-linking paradigm for coordination, which is comparable to the spreadsheet concept where cells are related to each other using a formula and changing the formula in one cell recalculates the value of the linked cell [11]. The basic entities that are coordinated in LinkWinds are objects shown at the windowing level as either data, control or display objects. Objects of the same type sit in the same window making an object view. The general purpose of coordination is to detect possible relationships in data.

LinkWinds allows one-to-many links; for example a slider broadcasts messages to all objects it is linked to when its value is changed. The user performs linking as well as unlinking interactively. In addition, there exist some constraints on coordination. For instance, data must be put into empty windows and messages are passed only between objects that are already linked. There is also a message-passing protocol that handles inter and intra object messages. The effect coordination causes on the user interface is the emanating flow between linked objects.

Program variables may be thought as analogous to coordination; for example, variables may be used in multiple places and accessed by reference, they must be instantiated, and they each have a type (if they are of a wrong type then they may be *cast* - either by default or explicitly). There are also notions of global and local scope.

3.3 The rudiments of coordination

Taking aspects from this analogy, the LinkWinds example, aforementioned challenges and other coordination tools we categorize the rudiments.

Coordination entities:
This details what is actually being coordinated; such as aspects of the actual window, view, data, record, tuple, attribute, parameter, process, event, function, graphic or time.

Type:
The *type* of the coordination determines the method by which the entities are linked. For example, simple coordination (such as rotation or transformations) may be implemented using primitive types (float, integer etc) while others may be more complex data-structures. Translation (casting) may be required if the entities utilize different types. The types may also determine directionality of the links (unidirectional or bidirectional). For example, IRIS Explorer allows parameters to be coordinated but the events flow one way (as it disallows simultaneously connecting the reverse to inhibit circular event explosions taking place).

Chronology (lifetime and scheduling):
How long entities are coordinated is governed by its *lifetime*. (This is also known as the persistence of the coordination.) It may be coordinated permanently, for a given action, or determined by some scope. Moreover, the coordination may be synchronous, asynchronous, reactive, and proactive. For example, it may be useful to coincidentally rotate the view of a fast and a slow renderer; one solution is that the time-consuming one could update at a slower rate by taking every *n* events from a coordination event queue, or merely notified at every *n* seconds.

Scope:
Scope determines both the global/local connection and the lifetime of the links; a global scope would mean that any entity (wherever and whatever it is) could be connected; whereas some links may be restricted to be only used in a local area (e.g. the user may setup a group where by simply adding a new member to the group automatically coordinates it to each of the others in the group). Commonly known as a Render Group. Moreover, the scope also may restrict the *lifetime*.

Granularity of links:
Many entities may be connected together via various links. Granularity determines: the number of entities in one coordination {2..n}, number of views in one coordination {1..n}, number of links an entity contributes to coordination {0..n}.

Initialization:
Initialization determines who or how the coordination created. For example, it may be automatic (such as using a Render Group) or user specific, scheduled in some fashion etc. Moreover, the user may need to explicitly connect entity A to B, for every type by (say) connecting ports from module A to module B; alternatively, the user may only need to drag and drop the whole module into a Render Group to instantiate and link everything in module A to module B.

Updating:
Coordinated views require that the information is dynamically up-to-date. However, there may be conflicting uses especially if, for example, the multiple views represent a visual history. Commonly, in a dataflow paradigm, the down stream modules always reflect the information upstream. However, it may be prudent that sometimes some views become out-of synchronization: such that they reflect a previous time in history. The displays may be updated by various means, such as eager or greedy update, lazy update, or user initiated.

Realization (link realization, user control):
How is the expression of coordinated conveyed to the user? It may be that explicit lines are used (such as the Spiral Calendar [19]) or via some formal layout mechanism [18]. Moreover, how does the user control the information to be linked, do they use direct manipulation or indirect means via (say) dynamic sliders?

4. The Model

The challenge is to develop a model that addresses the coordination design issues mentioned above without bias towards a particular data, navigation or communication paradigm. Effectively, it should be flexible, adoptable, extensible and foster better visual exploration.

The model should allow visualization designers to formally specify existing and novel coordinations in multiple view exploratory visualization and so facilitate early testing of the proposed coordination designs before they are implemented by programmers or constructed visually by the user.

4.1. Abstract model for coordination

The model we define includes “coordination objects” that manage combinations of *entities* (e.g. parameters) that control aspects of the linked views. A single coordination object is associated with each separate coordination in the system. A view is said to be coordinated if it shares a common coordination object. All the coordination objects for a visualization system are held in a "coordination space" as shown in Figure 1. This is similar to the middleware layer component we mentioned in the interdisciplinary view on coordination, Section 2.1.

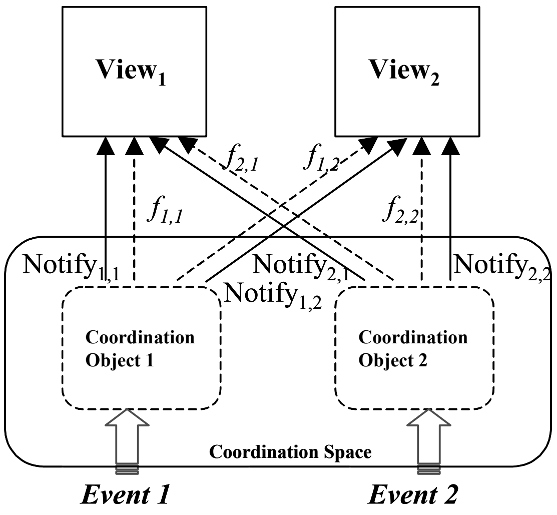


Figure 1. Abstract model for coordination in EV. This diagram shows two different coordinations between two views.

The views that are being coordinated need to define a translation function (for instance f_1 and f_2 as shown in figure 1) from the shared coordination object to the linked view parameters.

The views must also register to be informed of a notify event when a coordination object is changed. If an event occurs, which might typically be a user initiated action in one of the linked views, it alters the coordination object, which sends a notify to all the linked views registered. Registration may depend on a given *scope*. Then, those views that were notified of a change consequently use the information provided by the coordination object via their translation function to generate the new view.

A single coordination object is considered to be present for each *type* of coordination in the system. So that if multiple views define coordination for both co-rotation and brushing, this will be represented by two separate coordination objects: a rotation coordination object for co-rotation and a selection coordination object for brushing.

In the previous example, brushing uses the same visualization parameter for both views, selecting in one view results in selecting in another view and the same applies to co-rotation. However, a coordination object may hold more than one visualization parameter. For instance, one action in view V_1 may be linked to two actions in view V_2. In this case, we have three parameters in the coordination object.

An advantage of this model is its dynamic nature, as views may be added and removed without other views that also access the same coordination object necessarily having knowledge of this activity. Importantly, views do not need to know about other views in the coordination.

4.2. A layered approach to coordination

The views themselves are a result of parameter changes to the visualization process; an interaction or exploration would generate a new view, likewise viewing the same data by a different display technique (multiform) would provide a new view. These different instances may be displayed in different windows, overlaid into the same window or in fact replace the current window (replication, overlay, replacement, respectively [17]).

Often in exploratory visualization, visual correlation is seen as the focus of coordination, which tends to limit the techniques to brushing and navigational slaving. However, coordination may be understood in a wider context and occur on any variable or data at any level within the whole visualization process [17].

Consider the dataflow model [20] (which is used in many systems such AVS, IRIS Explorer, Amira, Data Explorer DX to describe the whole visualization process); the data is Enhanced or enriched in some form, then Mapped into an Abstract Visualization Object (AVO) that can be Rendered into an image (Figure 2). Multiple views are generated by splitting the dataflow at any stage of the pipeline (generating a fan-out). Aspects of the replicated modules may be readily associated to engender coordination. In additional to the traditional dataflow model, aspects may also be coordinated at the viewport Transform (see Figure 2). Such coordinated Transform operations might include simultaneously rotating viewpoints or altering projections. Moreover, coordination may occur at Window level, for example, moving or closing windows concurrently. Incidentally, the same visualization process can also be described using the data state model as it is equivalent to the dataflow model [21].

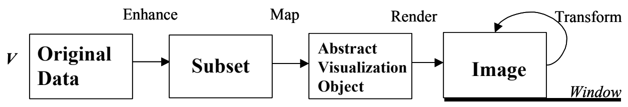


Figure 2. The data flow paradigm

One criticism directed at the data flow metaphor concerns the granularity of its processes. The dataflow paradigm describes a coarse model for the visualization process, for instance the entire graphics field is encapsulated in one process "rendering" [22]. In addition, a single process represents the mapping stage, which is the essence of information visualization

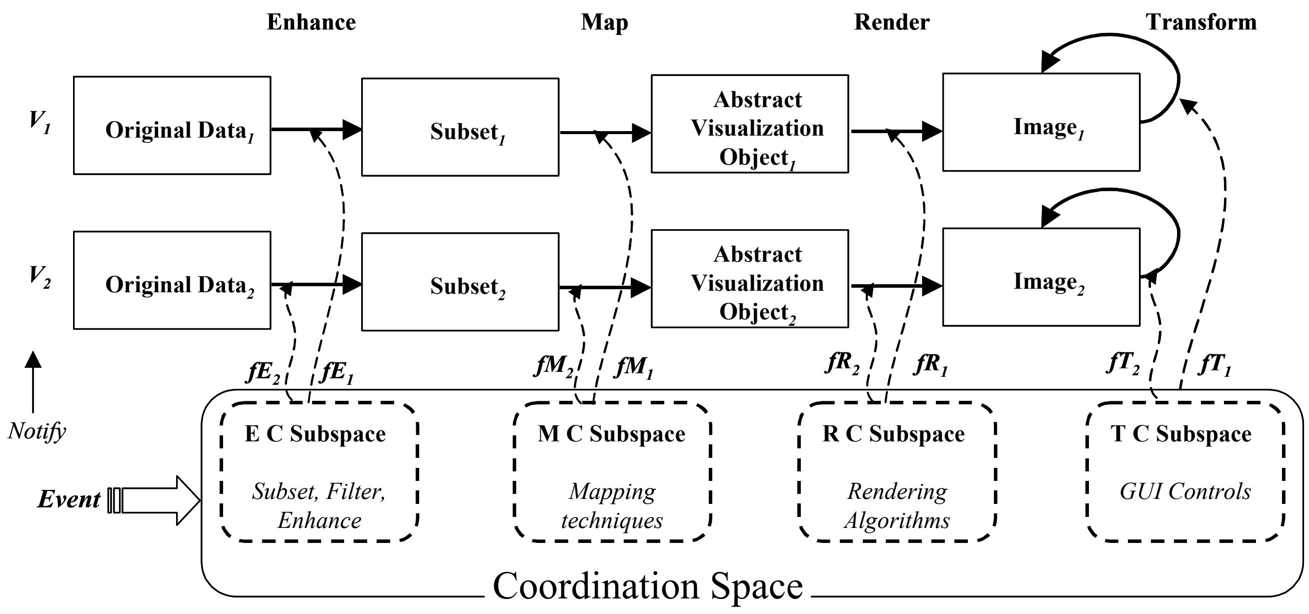


Figure 3 Coordination model for exploratory visualization showing subspaces

whereby information objects are mapped to visual objects. Moreover, the map and render stages of the data flow model are tightly associated in many application areas (such as 'information visualization') and thus often treated as one process.

Therefore, theoretically any process could be coordinated with anything (relevant translations applied) however; this may not be feasible or relevant. We propose coordination may occur at any process layer within the dataflow paradigm. However, we note that coordinations tend to occur within these levels.

4.3. Components of the model

Our model is divided into four components. They are the basic visualization processes and states, the coordination space, the events and the translation and notification functions.

4.3.1 *Basic visualization processes & states*

These are enhance, map, render and transform processes of the data flow pipeline as explained in section 4.2. Data is transformed in the data flow paradigm from the raw data to an image, firstly by enhancement to produce a derived data set. This is then mapped onto some geometry described by the Abstract Visualization Object (AVO). The final data state is the rendered image. These provide the coordination *entities.*

4.3.2 *Coordination space*

Each visualization system has one coordination space that holds various coordination objects. Coordination objects connect events and a number of linked coordinated views. In interactive exploratory visualization the event is often user initiated in one of the views.

One view may be associated with many coordination objects in a coordination space since a view may play various roles in different coordinations: it might be part of a focus-and-overview coordination, as well as a rotation coordination (see *granularity,* section 3.3).

The abstract parameters in a coordination object might be simple thresholds, coordinates of mouse clicks, or more complex notions such as modifications required to color maps or rendering algorithms (the concept of *type*, section 3.3). In the simple cases, the abstract parameters could easily be the actual parameters used by a view, such as a bounding box when filtering data in the Enhance section of the visualization pipeline. Even such simple parameters are not suitable in a raw form for all views, as some views may measure screen distances in different units, or from different origins, so translation functions may still need to be defined, for example to convert from measurements in inches to centimeters.

This demonstrates the need for a more neutral abstract format for storing these abstract parameters in some cases. Naturally, we might want to store the shared parameters in the same format as the event-generating view parameter format. Hence there will be no translation needed between the coordination space and that particular view. However, there might be a more suitable format for storing this parameter that suits more than one view, which is not the current format of the any view parameter, hence, in this case, all views would need to define a non-trivial translation function on the abstract parameter.

The abstract parameters stored in the coordination space can be grouped into four varieties of coordination sub-spaces (see Figure 3) since we describe the visualization process in terms of four processes: the enhance shared coordination space, the mapping coordination space, the rendering coordination space and the transform coordination space. However, in many situations the division between these sub-spaces is less rigid and coordination objects may include various parameters from each of the coordination sub-spaces.

4.3.3 *Events*

In this model, abstract parameters held in coordination objects are changed by events. Events can be generated by explicit user actions or automatically, by for example continuous analysis of the input data. Some user actions, such as selection with a mouse, are connected to a particular view, whereas others, such as keystrokes for altering parameters, are not.

Events modify abstract parameters, so they must be aware of the nature of the format and extent of the abstract parameters in the coordination objects. Where events are generated in particular views, the event-generating view will only be updated in the course of the event-notify cycle along with the other views.

As events associated with views are usually indirect in any case, and typically the cycle defined in this model is fairly immediate, this system seems an elegant manner of allowing multiple events, generated by any view linked to the coordination object, to modify the abstract parameters.

Whilst in this model we do not consider the time lag to be significant between the occurrence of the event and the notification event, this model should be adaptable to more critical real time visualization applications where large numbers of multiple events are occurring.

There may be many events associated with a particular coordination object, so allowing the modelling of various types of coordination. A simple master-slave relationship, where all the actions are in one view and which simply reflected in other linked views might have only one event inputting into the coordination object, from the master view. On the other hand, where the coordination allows input from any or all of a group of similar views, the number of events may be at least the number of linked views.

4.3.4 *Translation & notification functions*

A translation function takes the abstract parameters in the coordination object and converts them to view parameters which are used in the visualization process to produce the final image. Each view registered with the coordination object has one such function. The result of the translation function at the view level is the replacement of current view parameters affecting the operations in the data flow pipeline Often a translation function might affect only one operation, but it is perfectly valid for the function to coincidentally affect all of enhance, map, render and transform. The view integrator might define the translation function. It is also possible for a set of default registration methods to link a coordination object inside the coordination space to a certain class of views, which would then define a render group. The coordination object designer would define these defaults.

As well as defining a translation function, a linked view must be registered to receive notify events, that indicate when the coordination object has changed, indicating that the view's image needs to be updated. As there may be concurrent coordination objects defined over one view parameter, the notify indicates which coordination object has changed, and so which translation function must be accessed. The view must define a notify handler, that is triggered by the notify event. At its simplest the notify handler merely forces a regeneration of the image by resending the current data through the dataflow pipeline. Where temporal issues exist, for example with time consuming visualizations, so that perhaps several notify events occur during the production of one view, then the notify handler in the view must include a scheduling algorithm to deal with the queue that develops in the best way for that particular view, deciding whether to restart the visualization process or discard some notify events.

4.3.5 *Other elements in the coordination object*

As mentioned above, it may be desirable to include some notion of default registration for a render group of views so that a plug and play approach to linking views to the coordination object is possible. There are other possible components in sophisticated coordination objects. It may be possible to place constraints on both the type and number of views that can connect to an object. For instance, if one view wants to register to coordinate with another specific view over a particular coordination object, a constraint can be added to that coordination object as to only give those views permission for access and change. A further constraint on linking views might be related to its lifetime, so restricting linking to limited periods, or timing out views after a certain amount of time.

5. Using the model to define types of coordination in EV

Let *eve* be an exploratory visualization environment and V_1, V_2 are views within *eve*. V_1 and V_2 are described by the E.M.R.T composition where each of the components, Enhance, Map, Render and Transform has a coordination sub-space associated with it.

CASE I Disconnected Views
V_1 and V_2 are disconnected if they do not share any coordination object between them.

CASE II Linked Views
V_1 and V_2 are connected if they both access at least one common coordination object.

Two views can be *tightly coupled* if they have coordination objects that cover each of the sub-coordination spaces. In Figure 3, the tightly coupled views V_1 and V_2 share all possible coordination sub-spaces.

5.1 Example (enhancement coordination)

Given two views V_1 and V_2, a user interacts with V_1; this causes the firing an event, which affects E, M, R or T of V_1. For instance, the event might be an enhancement to the original data set, such that only values greater than 50 are included in the enhanced data set (Figure 4). If the user wants this event to play a role in coordination, the new filtering parameter will be stored in the coordination object E as default storage. Here is what happens at the view level and at the coordination space:

CASE I Disconnected views

The enhancement will only affect V_1 and the enhanced data will be mapped then rendered as specified by its own visualization pipeline.

CASE II Linked views

V_2 data set will also be enhanced given that the two views share the same filtering parameter. In the case of the identity coordination; no translation needed so both views use information in the coordination space as it is, a notify message is sent to both views enhance processes to use the new filter value. In the case of non-trivial translation functions, this value gets translated to view format. If there is no sharing at the remaining stages of the visualization process, each view will finally render accordingly.

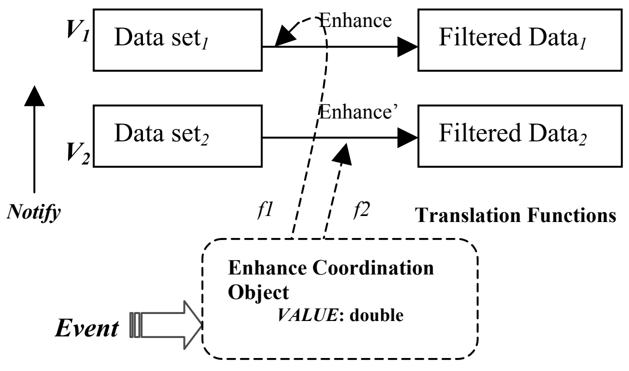


Figure 4 Schematic showing an example for coordinating an Enhance method.

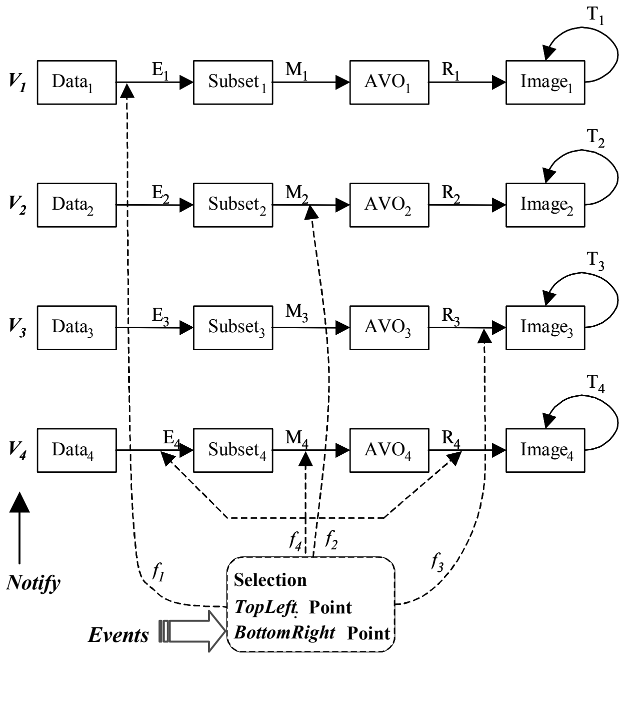
With this model it is possible to discuss the detail of coordinations. For example, where only the Enhance section of the data flow pipeline is altered in all linked views, we have type E coordination as shown in Figure 4. More complex combinations are possible, such as views where multiple operations are altered, giving R,E,TR coordination, where Render and Translate are affected in one group of view, Enhance only in a second and Render only in a third group of views. In a tightly coupled coordination we have coordination with the type EMRT.

5.2 Example (selection coordination)

We illustrate our model using visualizations of geographical map data. Diagrammatically our coordination objects and linked views are shown in Figure 5.

The event handled by the Selection coordination object is that of the user selecting a rectangular area of a map, an event that could be generated by any of the views.

The effect in V_1 is to filter the data to show only the selected rectangle whereas V_2 changes the color map to highlight particular objects, such as road junctions inside the rectangle. The result in V_3 is to modify the rendering inside the rectangle, increasing the level of detail to show smaller roads. Finally, the consequence for V_4 is to perform a combination of all these actions, cropping the data, highlighting certain objects and increasing the level of detail by altering the rendering.



V1 Data1 E1 Subset1 M1 AVO1 R1 Image1 T1
V2 Data2 E2 Subset2 M2 AVO2 R2 Image2 T2
V3 Data3 E3 Subset3 M3 AVO3 R3 Image3 T3
V4 Data4 E4 Subset4 M4 AVO4 R4 Image4 T4
Notify
f1
f4 f2
f3
Selection
TopLeft Point
BottomRight Point
Events

Figure 5 Schematic showing selection coordination.

Furthermore, f_1 is a simple identity function, as the Selection object holds parameters indicating which area of the map to crop, which is the information required by the Enhance flow of V_1. f_2 is slightly more complex, as it is alters part of the color mapping, so this function indicates that a subsection of the color map will be replaced by the mapping indicating selection. f_3 will pass the selection area to the renderer of V_3, with a flag indicating the desired level of detail. f_4 combines all the effects of f_1, f_2, and f_3 in V_4. This type of coordination is then E,M,R,EMR.

6. Discussion

Our model fits well with current visualization systems that implement coordination. For instance, Amira is a modular and object oriented software system for scientific visualization. It allows the simultaneous display of multiple data sets in different views or in a common view. Amira's components are modules and data objects, each of which has a set of parameters, which can be modified using a parameter editor in an interaction area of the application. Views are coordinated if they share some parameters displayed to the user in the object pool view. The user specifies which views to coordinate.

Similarly, IRIS Explorer users interactively create their application by linking modules; each module has some associated set of parameters, which describe its behavior. The control panel editor creates, modifies, and links module control panels and a parameter function editor creates relationships between parameters in linked modules [23]. The parameter value in the downstream module is then expressed as a function (*P-Func*) of the upstream parameter values (the translation function in our model).

Moreover, the authors of this paper developed a visualization system using Java in which coordination is implemented in the Object Oriented paradigm [24]. The system displays web search results in multiform views, where each search results returned by the search engine is mapped to a glyph. There is an interface Class for these abstract objects (glyphs), which acts as the middleware layer that facilitates coordination between glyphs. Glyphs which implement this interface share data and methods and hence can be coordinated over these parameters. In this application, users discover coordination in the midst of exploration.

Many visualization systems such as IRIS Explorer, AVS and Amira provide capabilities for user-oriented design; users could choose modules, edit parameters and link components. This facilitates coordination design. We note that these systems use the dataflow model to build applications for scientific visualization.

7. Future work & conclusions

In this paper we described how coordination objects in exploratory visualization are built from simple user interactions. Our model handles any combination of data sets and any number of linked views. The issues involved in coordinating different data sets and visualization methods are dealt with at the translation function stage, where abstract parameters are converted to meaningful parameters in the particular view.

Interactions are variations on the basic visualization functions: enhance, map, render and transform. If a view interaction is to play a role in coordination, it changes the coordination space, which is then reflected upon the linked views that use that space.

Our model is based on sharing objects (parameters) that control the rendered view and not the sharing of the data that is being visualized. Indeed, a view parameter can be part of more than one coordination object.

We notify all linked views upon change. Hence, we use the eager notify mechanism for our model (but not necessarily an eager update). However, we do not allow channels for storage and we are not concerned with establishing a protocol for notification in this paper.

Coordination as described by this paper is the mechanism through which views interact together to achieve purposeful goals that could not otherwise be achieved efficiently by these individual views working uncooperatively. Our model borrows ideas from recent research in visualization and other disciplines

Our future work includes implementing an example system that closely maps to the model and so demonstrates the flexibility, novelty of possible coordinations and practicality of the approach.

More research is required in the area of coordination design to provide rules and guidelines. In addition, comparative studies regarding users ability to work with independent compared to working with cooperative multiple views are still under-investigated. For example, how many coordinated events can one user keep track of during visual exploration? (Some work has been done, such as by North and Shneiderman [25]). Moreover, further research is required in the area of multitasking for multi-view exploratory visualization.

Moreover, new types of coordination can be introduced. We could have default coordinations based on default system or user settings. Furthermore, we can have recommended types of coordination if the system learns about user interactions, goals and existing coordinations.

There exist various commercial visualization environments, which implement coordination based on sharing parameters between visualization modules. We aim to enrich the shared coordination space to include not just the visualization parameters but abstract objects such as constraints, complex methods and time.

Acknowledgements

This work has been supported by EPSRC (grant reference: GR/R59502/01). CVEV Project homepage http://www.cvev.org

References

[1]. C. North, N. Conklin, K. Indukuri and V. Saini, "Visualization Schemas and a Web-based Architecture for Custom Multiple-view Visualization of Multiple-Table Databases", Information Visualization Journal, Palgrave, pp 211-228, December 2002.

[2]. G. M. Olson, T. W. Malone, J. B. Smith, "Coordination Theory and Collaboration Technology", Lawrence Erlbaum Assoc. 2001.

[3]. L. Gupta, J. F. Chionglo, M. S. Fox, “A Constraint Based Model of Coordination in Concurrent Design Projects”, Project Coordination Workshop of the IEEE Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative enterprises (WET ICE 96), 1996

[4]. Jun-ichi Akahani, Kaoru Hiramatsu, Kiyoshi Kogure, “Coordinating Heterogeneous Information Services Based on Approximate Ontology Translation”, Challenges in Open Agent Systems, at AAMAS'02 2002.

[5]. H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner, “Ontology-Based Integration of Information – A Survey of Existing Approaches”, Proceedings of the Workshop Ontologies and Information Sharing, IJCAI, pp. 108-117, 2002.

[6]. P. Ciancarini, R. Tolksdorf, F. Zambonelli, "Coordination Middleware for XML-Centric Applications", Proceedings of the 16th ACM Symposium on Applied Computing, Madrid, March 2002

[7]. M. Viroli, G. Moro, and A. Omicini. “On Observation as a coordination paradigm: an ontology and a formal framework”. In ACM Symposium on Applied Computing, Proceedings of 16th International Conference (SAC01), pp 166-175, Las Vegas (NV), USA, March 2001.

[8]. C. Ahlberg, B. Shneiderman, “Visual Information Seeking: Tight Coupling of Dynamic Filters with Starfield Display”, CHI’94, pp 313-317.

[9]. Matthew Ward. “XmdvTool: Integrating multiple methods for visualizing multivariate data”. In Proceedings Visualization ’94, pp 326-333. IEEE Computer Society Press. 1994.

[10]. C. Ahlberg. “Spotfire: An information Exploration Environment”. SIGMOD Record. 24(4): 25-29, December 1996.

[11]. A. Jacobson, A. Berkin, M. Orton, “LinkWinds: Interactive Scientific Data Analysis and Visualization”, Communications of the ACM 37, pp. 43-52, April 1994.

[12]. S. F. Roth, et al 1996. Visage: “A user interface environment for exploring information”. In Proceedings Information Visualization, pp 3-12. San Francisco, IEEE.

[13]. G. L. Andrienko and N. V. Andrienko. “Interactive Maps for Visual Data Exploration”. International Journal of Geographical Information Science, 13(4): 355-374. 1999.

[14]. C. North and B. Shneiderman. “A Taxonomy of Multiple Window Coordinations”. University of Maryland Computer Science Dept. Technical Report #CS-TR-3854. 1997

[15]. T. Pattison, M. Philips, “View Coordination Architecture for Information Visualization”, Australian Symposium on Information Visualisation, (invis.au). ACS volume 9, pp 165-171, 2001.

[16]. J. C. Roberts. “On Encouraging Coupled Views for Visualization Exploration”. Visual Data Exploration and Analysis VI, Proceedings of SPIE, volume 3643, pages 14-24. January 1999.

[17]. J. C. Roberts, “Issues of Dataflow and View Presentation in Multiple View Visualization”, In CISST Annual Conference, CSREA Press, pp 177-183, 2001.

[18]. J. C. Roberts. “Waltz - an exploratory visualization tool for volume data, using multiform abstract displays”, Visual Data Exploration and Analysis V, SPIE, volume 3298, pp 112-122. 1998.

[19]. J. Mackinlay, G. Robertson, R. DeLine, “Developing Calendar Visualizers for the Information Visualizer”. Proceedings of UIST'94, ACM Symposium on User Interface Software and Technology, pp 109-118, 1994.

[20]. R. Haber, and D. McNabb, “Visualization idioms: A conceptual model for scientific visualization systems”. In Nielson, G., Shriver, B., and Rosenblum, L., editors, Visualization in Scientific Computing. IEEE Computer Society Press, pp 74-93. 1990.

[21]. E. H. Chi, “A Taxonomy of Visualization Techniques using the Data State Reference Model”, Proceedings of InfoVis, IEEE Computer Society. pp 69-76. 2000.

[22]. A. M. Duclos, M. Grave, “Reference Models and Formal Specifications for Scientific Visualization”, Scientific Visualization Advanced Software Techniques, Ellis Horwood Workshops, pp 3-14, 1993.

[23]. J. Walton, “Data Visualization with IRIS Explorer What's New?” Tech. Rep. TR10/96 (NP3070), Numerical Algorithms Group Ltd., 1996.

[24]. J. C. Roberts, N. Boukhelifa and P. Rodgers, “Multiform Glyph Based Search Result Visualization”. Proceeding Information Visualization, IEEE, pp 549-554. July 2002.

[25]. C. North, B. Shneiderman. “Snap-Together Visualization: can users construct and operate coordinated visualizations?” International Journal of Human-Computer Studies 53(5), pp715-739. Academic Press. 2000.