A History Mechanism for Visual Data Mining

M. Kreuseler*
SD Industries GmbH,
89423 Gundelfingen,
Germany

T. Nocke†
Institute of Computer Science
University of Rostock
Rostock, Germany

H. Schumann‡
Institute of Computer Science
University of Rostock
Rostock, Germany

ABSTRACT

A major challenge of current visualization and visual data mining (VDM) frameworks is to support users in the orientation in complex visual mining scenarios. An important aspect to increase user support and user orientation is to use a history mechanism that, first of all, provides un- and redoing functionality. In this paper, we present a new approach to include such history functionality into a VDM framework. Therefore, we introduce the theoretical background, outline design and implementation aspects of a history management unit, and conclude with a discussion showing the usefulness of our history management in a VDM framework.

CR Categories: H.5.2 [Information interfaces and presentation]: User interfaces—Graphical user interfaces, interaction styles and user-centered design

Keywords: Visual data mining, Visualization, History, Undo/ Redo

1 INTRODUCTION

Visual Data Mining (VDM) has proven to be successful in exploring large, heterogeneous data sets. By combining automated and visual mining methods, a variety of exploration tasks can be supported to get a deeper insight into the data. Contrariwise, each exploration task is supported by different mining techniques, which require the processing of several steps in a special sequence.

Exploration denotes an undirected search for interesting features in a data set. Therefore, exploration typically includes different undo and redo steps to choose appropriate mining techniques as well as appropriate parameterizations to achieve the desired results. Moreover, it is often difficult to reuse earlier settings to roll back the same results. Thus, the analysis of huge data sets can be very time-consuming and confusing for a user.

To avoid several "trial and error" steps during the exploration process, Shneiderman introduced the *history task* in his *Task by Data Type Taxonomy* [18]. Following Shneiderman this task means to

> "keep a history of actions to support undo, replay, and progressive refinement."

Introduced already in 1996, Shneiderman's claim that information visualization systems should include functionality to preserve "the sequence of searches so that they can be combined or refined" is still not solved for most systems. This is an essential drawback especially in the context of visual data mining considering the increased number of visual and non-visual mining techniques in large frameworks.

Typically a user passes through similar tasks again and again, and only advanced users can handle complex tasks satisfyingly. Thus, users need support to handle large data sets effectively. This has to be considered when designing user interfaces for VDM frameworks. Especially, this includes the following aspects:

- simplifying the user interaction by undo/redo mechanisms,
- improving the user comprehension of the exploration process,
- improving the reusability of mining techniques, parameters and results, and
- supporting the selection of appropriate mining techniques for a certain data set.

In this paper we will design a history mechanism to achieve the goals discussed above. We have integrated this history functionality in our scalable VDM framework InfoVis (see Kreuseler et al. [13]). The paper is organized as follows. In section 2 we briefly describe related work. Section 3 gives a short overview about the general architecture of a framework for visual data mining including an effective history management. We introduce the theoretical basis of our history mechanism in section 4. Design and implementation aspects will be discussed in section 5. Finally, we show the applicability of our approach and discuss its challenges in section 6. We conclude in section 7 with a short summary and an outlook on further work.

2 RELATED WORK

History functionality has been established as an important feature of interactive software systems for a long time. For instance, Berlage [1] discusses the principal problems of selective undo of (isolated) actions in software systems. Moreover, he introduced the storage of, and the interaction with, actions in a history tree. Such undo/redo functionalities have been established in many fields of application, for example in computer-aided design (CAD) to substantially improve the construction process. The fast establishment of such undo functionalities in this background can be indicated by their spread from the year 1995 (in an overview paper for CAD systems undo functionality has not even been mentioned [4]) to 1997 (undo functionality is one of the evaluation criteria for CAD systems [5]).

Other examples for the usefulness of history functionality are the e-learning background and web guiding. The success of e-learning systems can be improved if former interactions of a user are used to adapt the learning process (see e.g. Kashihara et al. [11] and Plaisant et al. [16]). As an example for web guiding, Hightower et al. [7] introduce a graphical history-map of visited web pages in an internet browser to support user orientation, and proved their usefulness in user studies.

*e-mail: kreuseler@sd-industries.de
†e-mail: nocke@informatik.uni-rostock.de
‡e-mail: schumann@informatik.uni-rostock.de

IEEE Symposium on Information Visualization 2004
October 10-12, Austin, Texas, USA
0-7803-8779-1/04/$20.00 ©2004 IEEE

Similar functionality has been developed for visual data mining and visualization systems as well. Komlodi [12] analyzes the changes of history searches in information-retrieval systems. Derthick and Roth [3] use selective undo in combination with a time-line-based tree visualization for the data exploration system Visage. Events are marked with the current timestamp and ordered in a (temporal) hierarchy. Undo can be performed navigating along the time axis with a slider, and selective redo can be performed by dragging-and-dropping of event chains. Gayer and Slavik [6] introduce fluid simulator state trees for a complex combustion simulation scenario. This tree allows to store system and simulation changes caused by user interactions and replay resulting simulations in real-time.

Roberts [17] introduces multiple views of a data set, extending the sequential visualization pipeline to a hierarchic structure. In alternative analysis chains (branches of the tree), differing by alternative but similar parameterizations, many parameters can be reused, applying "feature-sets" and "render groups". This improves the comparability of views (e.g. by using similar color maps) and relieves users to redo same parameterizations in similar analysis chains. Another approach to improve the reusability of a visualization design has been introduced by Humphrey [8], who uses a graphical notation that is easy to understand even to non visualization experts to layout and reuse graphic designs. Ma [14] introduces image graphs that enable users to directly interact with a set of preview visualization nodes that represent different parameterized volume visualization images. Edges in this graph represent changes in parameterizations from one image to the other. Thus, the user can better understand parameter change impacts, and the exploration process itself. These image graphs are a kind of history trees, storing and managing all images of current interest intuitively. Jankun-Kelly et al. [10] extend Ma's [14] approach by a visualization transformation and a visualization session model. These models serve as a basis for the visualization of process graphs and are stored in an XML file format. Furthermore, Jankun-Kelley and Ma [9] introduce a new Focus & Context approach to the visualization of larger process graphs.

Altogether, history, selective undo/redo and operator management functionality has been proven to be very helpful in data analysis. However, it has been only partly established in visualization and visual data mining systems. Thus, we will describe a more general approach of history management for visual data mining systems, that includes undo/redo functionality and opportunities to reuse and select suitable mining techniques for visual analysis tasks.

3 VDM FRAMEWORK ARCHITECTURE

Existing frameworks for visual data mining tasks include a variety of visual and non-visual mining functionality. The theoretical background for such systems was laid out by the data state reference model of Chi and Riedl [2] that introduces several data transforming operators.

Figure 1 shows the architecture of a general VDM framework with history management based on this data state reference model.

The architecture consists of a GUI that enables users to steer the analysis process and to select and parameterize different kinds of operators, including all three categories introduced by Chi and Riedl. Data and metadata can be imported. These metadata can be used to control the application and parametrization of operators. Moreover, internal data structures for operators and data are the base for an efficient operator execution. Furthermore, the operator management handles operator execution, operator states, operator dependencies and is the interface between the data structures

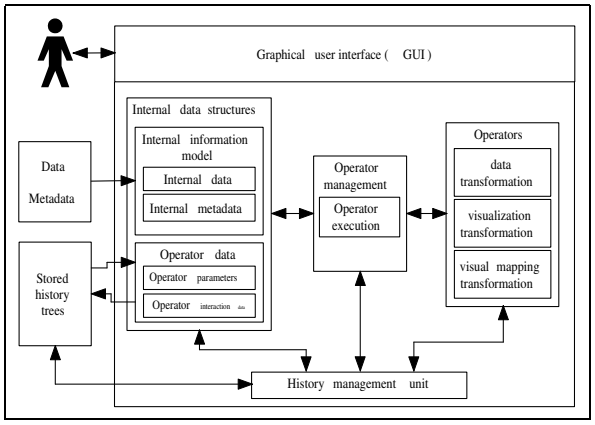

Figure 1: General layout of a visual data mining framework with an analysis process management unit

and the operator library. The history management unit (including undo/redo functionality) is connected with all the internal processes. Thus, operator states as well as their parameter values can be stored and reused. The user can interact with all these internal modules, including views on data structures, parameterization windows for the operators and views for the interaction with the operators (e.g. data flow diagram between the operators).

4 A HISTORY MECHANISM FOR VDM

In this section fundamental terms and definitions for the design of a unit encapsulating history functionality will be discussed.

Software system
A (software) system can be described as a 6-tuple $(A, I, O, z, \delta, \gamma)$, with A as the set of system attributes a_i $(i = 1 \ldots n)$, I the set of inputs, O the set of outputs, and with z as the current system state. Furthermore, $\delta : I \times Z \rightarrow Z$ is the state transition function (with Z as set of states), and $\gamma : I \times Z \rightarrow Y$ is the output function (with Y as the set of outputs). Then, the system state z is an element from the set $Z = D_1 \times D_2 \times \cdots \times D_n$, with D_i is the value range of the system attribute a_i. The function $\alpha : Z \times A \rightarrow D_i$ returns the attribute value av of the attribute a_i in the state z: $av = \alpha(z, a_i)$.

VDM basics
This general description has to be refined for our purposes. We want to describe a VDM framework as a 6-tuple $(A, I, O, z, \delta, \gamma)$ with the following semantics:

- $A = \{OP, OC, EV\}$ is the set of system attributes, with OP the set of operator states, OC the set of Operator connections and EV environment settings (e.g. user profiles).
- $I = \{UI, DI, MR\}$ is the set of inputs, with
  - UI as the set of user interactions (setting of operator and environmental parameters, of operator connection adjustments and graphical interaction),
  - DI as data input
  - MR as mining results (output data from previous steps).
- $O = \{MR\}$ is the set of system outputs that are mining results (which can be computed by any operator and are stored and displayed in any kind of windows).

- $z_i = \{av_1, \ldots av_n\}$ is the current system state, and is defined by the value of all current relevant system attributes.
- $\delta : UI \times Z \rightarrow Z$ is the operator transformation function that transfers an operator (and its result windows) from one state to another triggered by user interactions, and as the case may be, as well transfers connected operators. An example for this is the recalculation of an visualization window driven by a mouse interaction. This may lead e.g. to a rotation of the displayed data.
- $\gamma : I \times Z \rightarrow O$ is the output function, with the following special cases:
  1. $\gamma_1 : DI \times Z \rightarrow MR$ - transferring raw input data to mining results, for instance the application of a cluster algorithm that clusters the raw data, or a visualization of the raw data
  2. $\gamma_2 : MR \times Z \rightarrow MR$ - transferring mining results from previous steps to further refined mining results.

The functions δ and γ can be understood as data mining operators that transfer input data – based on user interactions – to new operator states and to visual and non-visual outputs. Furthermore, these operator functions can be organized in chains that are automatically executed by a single user action.

Dependencies
In a VDM framework several state transitions δ and output functions γ are combined to generate certain operator states as well as output data. For the storage and the reconstruction of such chains for the purpose of history management, dependencies between these operators have to be considered. Let I_{op_i} be the set of inputs for an operator op_i and O_{op_i} be the set of outputs of the operator op_i. Then,

$$op_i \text{ is } \textbf{independent of } op_j \ (i \neq j) :\Longleftrightarrow O_{op_j} \cap I_{op_i} = \emptyset. \quad (1)$$

Vice versa,

$$op_i \text{ is } \textbf{dependent of } op_j \ (op_i \sim op_j) \ (i \neq j) :\Longleftrightarrow O_{op_j} \cap I_{op_i} \neq \emptyset. \quad (2)$$

Furthermore, an operator

$$op_i \text{ is } \textbf{competing to } op_j \ (op_i \,!\, op_j) \ (i \neq j) :\Longleftrightarrow O_{op_i} \cap O_{op_j} \neq \emptyset. \quad (3)$$

Figure 2 depicts an operator chain which transfers a given data input into a calculated mining result. This operator chain includes dependent and competing operators. Moreover, the figure includes the associated dependency structure. Two types of dependencies are considered: system inherent dependencies and user defined dependencies that describe the temporal order of the operations invoked by user interactions.

History Management
Operator chains express dependencies of transfer and output functions in general. However, to manage concrete user interactions for undo and redo purposes, we need an appropriate data structure which stores executed operators and their states. We call this structure a history tree. A history tree stores a whole (mining) exploration process on a data set owing to its dependency structure, starting at the initial state, branching on nodes, if alternative operator chains were executed. Beneath the possibilities to undo and redo operations based on this history tree, successful branches can be labelled and stored to the file system to recall them for later exploration and presentation purposes.

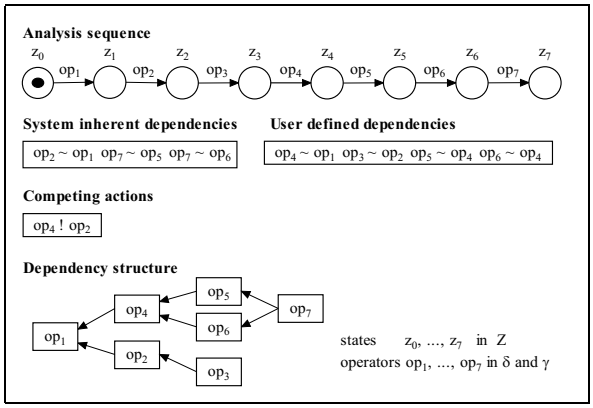

Figure 2: Example for an analysis sequence and its dependency structure

To realize this history management we define a VDM framework with history management as a 7-tuple $(A, I', O', H, z, \delta, \gamma)$, with H as the history tree, I' is the extended input set with $I' = I \cup H$, and O' is the extended output set with $O' = O \cup H$. This means that history information can be internally and externally stored and reloaded.

5 Design and Implementation of a History Management for VDM

After the discussion of the theoretical background of history management in a VDM system, this section will outline practical challenges in designing and realizing such a history management. There are 4 aspects to be considered in this context:

1. internal history management, including the internal management of the history tree and the operator state information,
2. the interface between the history management unit and the actions in the visual data mining system
3. the external storage of history trees in a file system and
4. the intuitive, interactive visual representation of the history tree.

5.1 History data structures and history management

Following Berlage [1], there are four possibilities to recall operator states:

- Storage of actions: all actions are stored. A recall of any state of the visual mining operators is done by returning to the starting state of the system and recalling the actions one by one.
- Storage of actions and inverse actions: additionally, the inverse actions are stored. This allows to undo actions directly. In some cases, the inverse actions are not trivial to be calculated.
- Storage of the system state: the entire state is stored. Recalling is done by exchanging the state of all operators. Disadvantage of the procedure are high storage costs.
- Storage of the system state change: all the internal data structures that have been changed by an action (that resulted from operator execution) are stored. This is not as storage expensive, but not trivial to implement.

Our implementation is based on the first variant describing the actions in a history tree that is implemented as a common explicit tree. This is practicable for visual data mining, because for many operators no efficient inverse operation exists. Usually, it is easier to recalculate operators than to roll back the last action. Applying the first variant, the tree can be stored memory-efficiently both internally and externally (see fig. 1 and section 5.3). Moreover, we cache the last executed operator states and enable users to enrich all operators with additional information about the aquired knowledge. This information can be used to prevent users from time-consuming re-calculations. Altogether, we achieve a balanced compromise of memory efficiency on the one hand and usability, reusability and general applicability on the other hand. The nodes of the tree represent operators and the edges represent the dependency structures between the operators.

To support reuse, interpretation and evaluation of operators and of history tree branches for the exploration process, a variety of information are stored in the history tree nodes. Automatically, the input data of an operator, its parameterizations to describe the current state, its output data and information about the system state are stored. Furthermore, users can specify additional information about the interpretation of the mining result constructed by the operator. These additional information include the exploration tasks the operator execution supports and a textual specification of insights. The idea of using exploration tasks in this context is to evaluate if a certain chain of operator executions performed can be reused in a similar scenario, for instance on another yet similar data set. Then, the user specifies a task, and the history management reloads a certain chain of operators experienced on another data set. Furthermore, operators can be grouped to to single nodes: for instance a complex visualization operation contains a chain of elementary operators, such as a visualization transformation and visual mapping transformation operators.

In the special case that no inherent dependencies between two following operators are given, an additional edge is inserted to show the temporal (user-defined) dependency of these nodes. Moreover, suitable functions for the insertion and deletion of nodes and branches have been included. Among other things, this enables users to delete partial trees that are not relevant for further investigation.

5.2 Interface

The realization of a history management unit requires an interface to the visual data mining system that allows to protocol and recall user actions in the VDM system. This includes that the VDM system sends transfer information to the history management, and that the functionalities of the VDM system can be externally driven by the history management.

Our solution is the implementation of an interface class, including a registration mechanism for all the operator transformation functions that can be activated by user interactions. This class manages operator execution and operator parameterization functions. If a user interaction starts a function, several information can be acquired from this class concerning the invoked function. Furthermore, the play-back mechanism of the history management uses this class to call the operator functions related to the nodes of the history tree.

5.3 External storage of history and states

Many systems which are using history mechanisms such as CAD programs or word processors produce and save documents as results of the working process. This is different in VDM where insights of the data are usually revealed by applying mining functions in an iterative fashion. This includes re-applying functions with different parameters and changing input values. This means the result of data mining process is much more a sequence of mining steps than a final document. Moreover in many exploration scenarios, especially in the context of heterogenous or very large data sets, it might be useful to re-execute or to continue with a previously recorded mining process. In order to achieve this, we provide mechanisms to make recorded histories persistent. Therefore, we construct and store a history tree that makes a variety of information describing the exploration process available.

In addition to the internal history tree as described above, we also externally store attributes such as execution time and date of an operator. In order to manage and store the history tree we created a hierarchical file format which is based on XML. Our file format is generally applicable for a variety of environments. Since storage of history trees is not memory expensive and can be done very time-efficiently, we do not store the computed data structures explicitly, but store kind and parameters of the operators to achieve the data structures. Furthermore, our data format allows assigning several mining histories to a single data set. This is more efficient in terms of disk space and makes it easier to compare different mining processes of the same data.

5.4 GUI and History visualization

Usability and value of the history management depend on the capabilities of the user interface.

Our user interface supports two primary groups of operators - generating and displaying visual representations of the recorded history, and providing a set of tools for manipulating, (re)executing and evaluating operators of the recorded history.

The dynamic nature intrinsic to the work with interactive VDM systems causes permanent changes resp. steady growth of the current history. This has to be incorporated into the history visualization. Subsequently special requirements have to be fulfilled such as

- generating compact visual representations of the history in order to avoid overlap with the VDM system's primary display windows,
- updating the history display driven by events, and
- visualizing the history tree, i.e. how operations within the history relate to each other, displaying the most important properties of the current history's elements.

Our graphical interface displays the current history based on a node-link hierarchy tree layout. Here, visual representations of the hierarchy nodes depict operators or grouped operators. Since we assume the user of a classical data mining session is – in general – interested in the last few executed operators only, a classical tree layout is sufficient for this purpose (for further discussion see below). The nodes are represented by small glyphs that encode two types of information. First, a thumbnail picture displays which operator has been applied. In contrast to the works of Ma, Jankun-Kelly and Gertz [9][10][14] we use abstract icons to describe the operators instead of concrete rendered preview images. The idea behind this is that a metaphoric icon can often encode more information and be identified faster than a downscaled image. Second, we added small icons that encode the following information:

- type of operator(s) represented by the node: colored rectangles and triangles depict what kind of operators and what kind

of output functions had to be called by the VDM system in order to perform this particular mining or visualization operator represented by the hierarchy node. Types of operators are for instance *data transformation operators* (indicated by a white-blue triangle), *visualization transformation operators* (indicated by a blue-yellow triangles) and parameterizations (indicated by a dotted white rectangle),

- the number of operators (for grouped operator nodes).

Operator nodes are connected by edges that refer to user interactions that triggered new VDM operations, i.e. consecutive user interactions within the visual mining process are linked by an edge within the tree representation. There are two types of edges:

- user-defined (temporal) dependencies are represented by dashed lines,
- system-defined dependencies are represented by continuous lines.

Figure 3 illustrates the visual representation of a recorded history of a complex mining process.

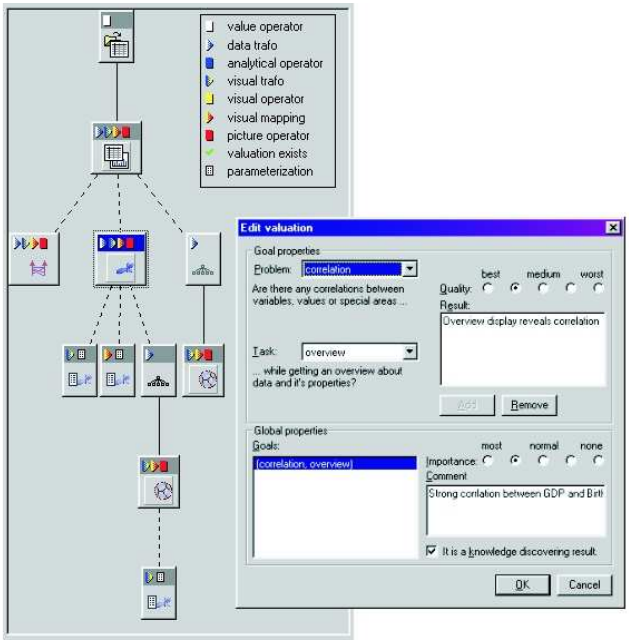

Figure 3: Visualization of a mining history based on a hierarchy tree in combination with an editor window for assigning special user information to hierarchy nodes.

The hierarchy display provides a basis for analyzing and navigating the recorded history. Furthermore, our visual interface implements a set of interaction tools. Each node of the displayed history can be selected. A variety of operations can be applied on selected nodes, such as activating, deactivating, changing, undoing or redoing the action behind that node. A complete branch of the history which represents a chain of operators can be replayed, i.e. all mining operations which have been recorded to that history branch are re-executed. Alternatively, one can choose a particular node in order to replay only small parts of the recorded history. Thus the entire mining process or single parts of it can be repeated.

The visual interface is linked via a Brushing mechanism to the VDM system, i.e. as soon as a history node is activated or selected the corresponding window within the VDM system is highlighted as well. We found that especially this feature makes working with a history tree resp. with redo/undo mechanisms a lot easier.

As already mentioned above, we assume that the user of a classical data mining session is – in general – interested in the last few operators only, a classical tree layout is sufficient, and – even for non-visualization experts – easy understandable. Thus, using the scrolling mechanisms of the window, the last operators assure that the most important part of the history tree stays within the focus. Another case is if the user wants to work with externally stored history trees, and for example therefore wants to recall certain history states. In this case,

- if history trees get larger, and
- if the user wants to get general orientation and/or
- jump into arbitrary operator nodes,

he can resize the history window to screen size. For our current scenarios this strategy was sufficient (until a number of approximately 50 operator nodes). For the visualization of even larger history trees other paradigms such as Focus & Context or information hiding could be applied, for instance using the hierarchy visualization technique MagicEyeView. A further solution to handle larger history trees is to extend the operator grouping mechanism by nesting groups of operators in multiple abstraction layers, and thus reducing the number of operator nodes to be displayed.

Moreover, our history contains an editor for adding further information to each history tree node (see fig. 3). This editor was especially designed for gathering data mining results, i.e. for recording the insights into the data, which have been obtained from related operators in the VDM system. In order to make the data input more efficient, the editor offers a set of predefined exploration tasks common in visual data mining which can be selected and assigned to nodes. Furthermore, priorities, rankings and supplying textual information for individual user notes for operator nodes can be specified.

The recorded information, which may represent the main results of the data mining process, can be used for gathering deeper insights into the data. Therefore we provide a tool that allows searching the recorded history tree based on the user supplied additional information, i.e. the user can search, for instance, for all those history nodes that represent operators related to a particular exploration task. Other examples might be finding all history nodes with a priority above a given threshold or with specific characteristics such as a certain type of operator. Furthermore we plan to extend our searching tool to compare different history trees of different mining processes.

6 Discussion

After these considerations we want to introduce an example for the application of the history management in a VDM framework, and discuss its advantages as well as challenges for our future work.

Figure 4 shows a history tree (left) of a data mining process and the related visualization windows (right) of the VDM system. This example illustrates the usefulness of a visual representation of a recorded history for the VDM process. A data set is selected (top node). This data is displayed by a technique called the data table view (2nd node). The data table view provides a general overview (compact line mode in the context area), and detailed information about values of the country Botswana. After exploring the table, the user employs a hierarchical cluster algorithm (3rd row) to analyze the general structure of the countries. As a first display technique, the hierarchical technique MagicEyeView is launched to vi-

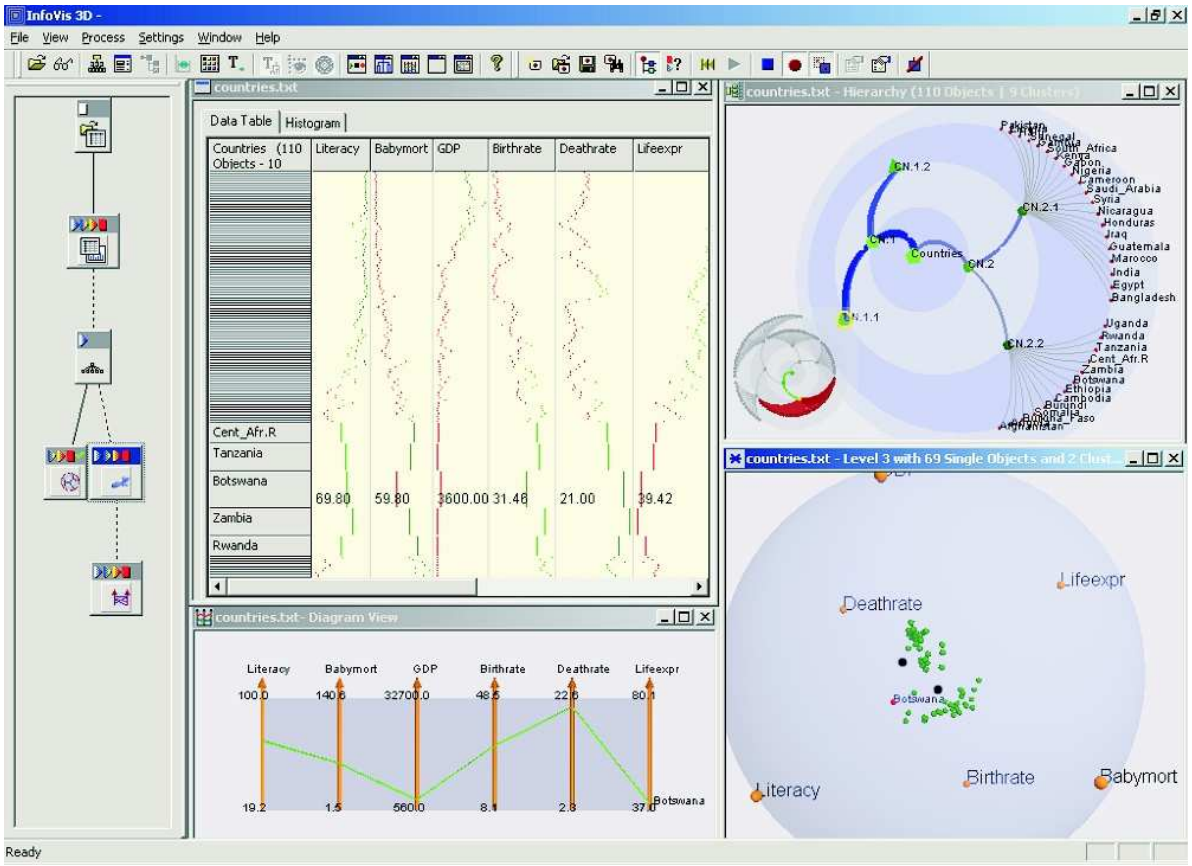

Figure 4: Screen shot from the VDM System InfoVis (a demographic data set of the countries of the world) with History tree (left), DataTable display in a table lense mode (center), tree visualization (MagicEyeView) of result of a hierarchical cluster algorithm (top-left), parallel coordinate display with the selected polyline of Botswana (center-bottom), and a ShapeVis display (bottom-right) showing all the countries from the captured countries from the MagicEyeView

sualize the cluster results (1st node in the 4th row). Particular clusters which contain developing countries are focused in the display. Afterwards, the user decides to get further insight into the structure of these clusters. Therefore another visualization technique called ShapeVis is applied (2nd node in the 4th row). ShapeVis displays details about those countries within the clusters of interest with green spheres while the remaining, less interesting countries are aggregates and represented by 2 black spheres. Continuing our example, the user selects the country of Botswana that has different attributes compared to all other countries within the cluster. Afterwards, to get deeper insight of this aspect, a parallel coordinate view is launched (5th row). Now the data of the country Botswana can be analyzed in different views - the parallel coordinate view and the TableView. As a result the user obtains the information that Botswana has both relatively low birth and death rates in comparison with similar countries.

The example shows the advantages of a history management: the user can continue the exploration process by investigating other clusters of countries and roll back to former results, for instance to the specifics of the country Botswana whenever he wants. Furthermore, he can directly select one of the output windows by clicking the associated node in the history tree. This is of high benefit considering the usually numerous, overlapping output windows (in figure 4 we avoided overlapping for better understanding our example; however, typical VDM systems produce a lot of overlapping windows).

Figure 5 shows the editor window to enrich a node of the history tree with additional information. Here, the knowledge gathered and the tasks fulfilled with a certain mining operator can be described and stored. On the one hand the dialog allows to specify goals and tasks that have been performed by the exploration of the MagicEyeView output. In this case these are a general overview about the cluster structure of the data set and details-on-demand about the underlying categories. Furthermore the user can annotate and evaluate how well these goals are fulfilled by the technique. On the other hand, the user can store the knowledge gathered by the output window and its importance for the general exploration process. In this case information about two general clusters of the countries are extracted: two small cluster of developing countries, and 2 bigger clusters of the other countries.

To summarize, the example described above depicts three features of the history management approach: the intuitive orientation in the exploration process, the opportunity to simply and quickly recall history states, and the storage and reuse of evaluated exploration history for the current data set as well as for similar data sets in a similar exploration context.

In working with a history management in a VDM framework, there are still challenges remaining. First of all, there are problems arising from the size of a history tree. If each atomic user interaction is represented by a node, the history tree soon becomes complex and complicated to handle. As described above, for instance Focus & Context mechanisms can be integrated to solve this problem. Our

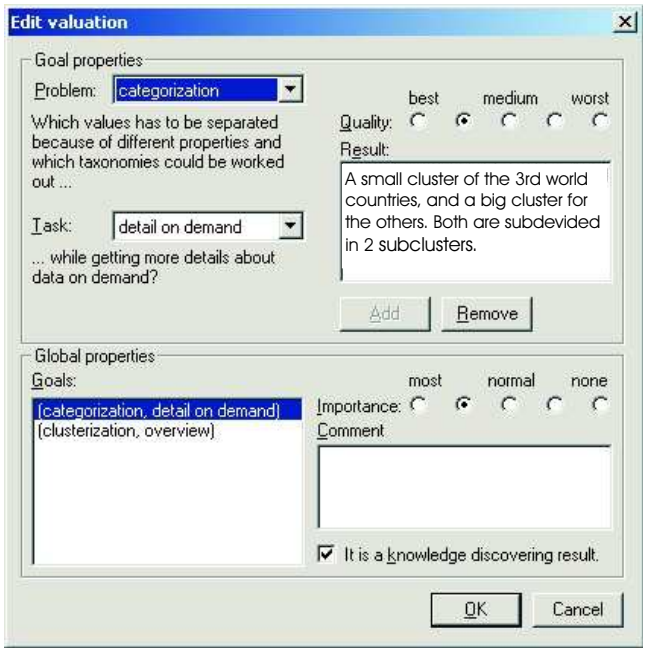

Figure 5: Screen shot of the mining operator evaluation dialog from the VDM System InfoVis (for the MagicEyeView history node)

approach was to accumulate operators to functional groups. This has several advantages:

1. The complexity of the history display can be reduced, and the user can acquire details-on-demand. This includes the development of new visualization techniques to display and navigate the history.
2. In many cases a single mining operator is not sufficient to support complex exploration tasks. A user may want to mark a set of operators as suitable for a complex goal with the intention to reuse the whole group of operators in later explorations on similar data sets.

However, a combination of grouping and Focus & Context techniques is beneficial. Currently, we are developing another VDM-System with a history mechanism: the Visana system for visual data mining in modelling and simulation environments (see Nocke et al. [15]) that integrates mining operators on a high abstraction level (a kind of grouping), hiding their internal structure. In this VDM framework we are currently designing a mechanism for

- user-driven selection of an operator module – covering a certain number of separate operators – in a data flow chart,
- the automated insertion of operators by a history management, and
- semi-automatic selection support mechanism in dependency on the exploration context (VDM design).

In the first case, the user can drag-and-drop operators to the graph such as in module based visualization systems (e.g. OpenDX [19]). This requires some knowledge about dependency and operator input and output data types. In the second case, the history management inserts operators into the flow chart, based on a user-driven selection of an operator from the main window menu, not requiring any knowledge on the operator dependency structure. This structure is only present by enabling and disabling window menu items. In the third case, the user will specify goals to be performed on the current input or mining result in any position of the mining process, and by a semi-automatic selection support mechanism a set of new operators and operator transformations will we added to achieve this goal.

7 Conclusions

In this paper, we investigated the integration of a history management into a VDM framework. Therefore, we described the theoretical basics of VDM and history management, including a VDM definition enriched by history functionality and a discussion of operator dependencies. Based on these theoretical issues, we described our approach to design and implement a history management system for VDM systems, and outlined how we integrated the history management into the InfoVis framework. This involved internal and external storage of history trees, the definition of an interface between the history management unit and the VDM functions, and displaying and editing the history tree itself. Then we outlined the advantages of using a history tree based on an example.

Finally, further research has to be done to explore the application, interpretation and reusability of applied operators stored in a history tree. This includes further improvement of the history tree visualization:

- separating successful history branches from unsuccessful subtrees: this can be done e.g. by color coding the branches,
- applying more advanced visualization techniques for larger history trees, for instance Focus & Context displays (see e.g. [9]).

Moreover, we must investigate the reusability of operator chains for similar exploration and presentation tasks. This requires the distinction of strongly data-set-dependent (e.g. attribute selections) and data-set-independent (e.g. highlighting of outliers) branches of the history tree, to decide which operators can be completely reused, which have to reparameterized and which are not transferable from one exploration scenario to a similar scenario.

Furthermore, the possible extension from history trees to directed history graphs has to be considered if the dependency structure gets more complicated, especially if the execution of an operator depends on more than one input. Then, this history graph becomes a structure similar to data flow graphs used in module-based visualization systems such as OpenDX (see e.g. [19]).

Acknowledgements

The authors thank all the students working on our VDM-framework for their involvement. Our special thank goes to Arne Klaassen for implementing the history mechanism and integrating it into our framework. Furthermore, we thank Georg Fuchs for proofreading our English.

References

[1] T. Berlage. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Transactions on Computer-Human Interaction*, 1(3):269–294, 1994.

[2] E.H. Chi and J.T. Riedl. An Operator Interaction Framework for Visualization Systems. *Proceedings of the Symposium on Information Visualization '98*, 1998.

[3] M. Derthick and S. F. Roth. Enhancing Data Exploration with a Branching History of User Operations. *Knowledge Based Systems*, 14(1-2):65–74, March 2001.

[4] G. Fandel, P. Francois, and K.-M. Gubitz. CAD market study (in german: CAD Marktstudie). AIP-Institut Hagen, 1995.

[5] M. Furrer. Market overview CAM systems I & II (in german: Markuebersicht CAM-Systeme). Technical Report 16, CAD-CAM-Report, 1997.

[6] M. Gayer and P. Slavk. Pre-Calculated Fluid Simulator States Tree. *In: Twelfth IASTED International Conference on Applied Simulation and Modelling. Anaheim : Acta Press*, pages 610–615, 2003.

[7] R.R. Hightower, L.T. Ring, J.I. Helfman, B.B. Benderson, and J.D. Hollan. Graphical Multiscale Web Histories: A Study of PadPrints. *Benderson, Shneiderman (eds.): The Craft of Information Visualization, Readings and Reflections*, pages 220–227, 2002.

[8] M.C. Humphrey. Creating Reusable Visualizations with the Relational Visualization Notation. *Proceedings of the IEEE Visualization'00, Salt Lake City, Utah, USA*, pages 53–60, 2000.

[9] T.J. Jankun-Kelly and K.-L. Ma. MoireGraphs: Radial Focus+Context Visualization and Interaction for Graphs with Visual Nodes. *In Proceedings of the IEEE Information Visualization 2003 Conference, Seatle, USA*, October 2003.

[10] T.J. Jankun-Kelly, K.-L. Ma, and M. Gertz. A Model for the Visualization Exploration Process. *In Proceedings of the 13th IEEE Visualization 2002 Conference, R. Moorhead, M. Gross, K. I. Joy (eds.)*, pages 323–330, October 2002.

[11] A. Kashihara, Y. Satake, and J. Toyoda. A History Visualization for Learning-by-Exploration in Hypermedia on WWW. *Proceedings of WebNet 98. Orlando. Florida.*, pages 497–502, 1998.

[12] Anita Komlodi. Search history for user support in information-seeking interfaces. *Extended Abstracts of ACM CHI 2000: Human Factors in Computing Systems Conference. The Hague, The Netherlands*, pages 75–76, 2000.

[13] M. Kreuseler and H. Schumann. A Flexible Approach for Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1), January-March 2002.

[14] K.-L. Ma. Image Graphs - A novel Approach to Visual Data Exploration. *Proceedings of the IEEE Visualization'99, San Francisco, CA, USA*, pages 81–88, 1999.

[15] T. Nocke, U. Boehm, H. Schumann, and M. Flechsig. Information Visualization Supportung Modelling and Evaluation Tasks for Climate Models. *In: Proceedings of the Winter Simulation Conference, WSC'03, New Orleans, USA*, December 2003.

[16] C. Plaisant, A. Rose, G. Rubloff, R. Salter, and B. Shneiderman. The design of history mechanisms and their use in collaborative educational simulations. *Proceedings of the Computer Support for Collaborative Learning, CSCL'99*, pages 348–359, 1999.

[17] J. C. Roberts. On encouraging multiple views for visualization. In *Proceedings Visualization'98*, pages 8–15, 1998.

[18] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. Technical Report UMCP-CSD CS-TR-3665, College Park, Maryland 20742, U.S.A., 1996.

[19] D. Thompson, J. Braun, and R. Ford. *OpenDX, Paths to Visualization*. Visualization and Imagery Solutions, Inc., first edition edition, 2001.