J Vis (2016) 19:543–559
DOI 10.1007/s12650-015-0329-3

REGULAR PAPER

Jie Liang · Mao Lin Huang · Quang Vinh Nguyen

Navigation in large hierarchical graph through chain-context views

Received: 16 August 2015 / Revised: 8 November 2015 / Accepted: 22 November 2015 / Published online: 17 February 2016
© The Visualization Society of Japan 2016

Abstract The most commonly used interaction techniques in space-filling visualization are drilling-down + semantic-zooming and focus + context methods. However, under these schemes, users often have insufficient knowledge about contextual information to guide them exploring through very large and deep hierarchical structures. This paper proposes an efficient interaction method called "chain-context view" (CCV) for the navigation in space-filling visualizations. Instead of displaying a no or one context views, we provide users with a progressive sequence of context views, which maximize the display area of contextual information. The rich contextual information provided in the exploration path could greatly increase the accuracy of user's decisions and reduce the "unsuccessful trips" and "unnecessary views" while locating the target object by browsing in deep levels of hierarchical structures with CCVs. The new method allows the users to trace each step of their interactions and make it easy to jump or return to any level of the hierarchy that they have previously visited. A usability study was conducted to evaluate the effectiveness of the CCV, by measuring the user performance and satisfaction on the navigation of deep levelled relational structures.

Keywords Information visualization · Interaction · Focus + context · Chain-context

1 Introduction

Graphs generated in real-world applications are often very large with thousands or even millions of items, such as research citation and collaboration networks. Space filling is considered to be a good approach for visualizing large hierarchical datasets with optimized visual presentation of global patterns of the overall data structure. It has a high capacity for the visual representation of very large hierarchical data with excellent space efficiency. This technique ensures space efficiency by dividing the display area into nested rectangles (Bederson 2001; Bederson et al. 2002; Bruls et al. 2000; Johnson and Shneiderman 1991), polygons (Balzer et al. 2005; Nguyen and Huang 2003; Wattenberg et al. 2006), ovals (Andrews et al. 1998;

J. Liang · M. L. Huang
Faculty of Engineering and IT, University of Technology, Sydney, Sydney, Australia

J. Liang (✉)
Key Laboratory of Machine Perception (Ministry of Education), School of EECS, Peking University, Beijing, China
E-mail: christy.jie@gmail.com

M. L. Huang (✉)
School of Computer Software, Tianjin University, Tianjin, China
E-mail: mao.huang@uts.edu.au

Q. V. Nguyen
MARCS Institute, School of Computing, Engineering and Mathematics, University of Western Sydney, Sydney, Australia

Stasko and Zhang 2000) or other shapes and assigning them as geometrical regions to subsets of the entire dataset for displaying. Space-filling techniques, especially Treemaps (Balzer et al. 2005; Bederson 2001; Bederson et al. 2002; Bruls et al. 2000; Johnson and Shneiderman 1991; van Wijk and van de Wetering 1999), have also shown high applicability and commercial value in many areas, such as finance analysis (Smartmoney Marketmap 2007), sport reporting (Jin and Banks 1997), image browsing (Bederson 2001), software and file system analysis (Baker and Eick 1995; Johnson and Shneiderman 1991) and food safety (Chen et al. 2015). Variation of enclosure approach has been extended to (Nguyen and Huang 2003; Liang et al. 2013). Some alternative techniques have also been proposed (Abello et al. 2004; Fekete et al. 2003; Nguyen and Huang 2005; Zhao et al. 2005) which combine the enclosure approach with node-link diagrams so both the space utilization and the explicit representation of the relational structures are archived.

Although the layouts generated in space-filling visualization are very efficient in terms of space utilization, the issues of "viewing-ability" and the ability to produce user-friendly interactive interfaces to guide users exploring data accurately are critical, especially for visualizing large and deep relational datasets (Fekete and Plaisant 2002). This is because that with space-filling visualization as well as other visualization techniques, it is hard to discern between nodes and edges, hierarchical levels, labels and other properties when a dataset of thousands of items is displayed concurrently (Herman et al. 2000). Therefore, an efficient and effective navigation scheme, combined with a visualization which provides users with rich contextual information, is essential when navigating large data structures. The navigation scheme should enable users to interactively adjust views to reach the final view of a sub-graph, allowing them to browse, and locate the desired object within the hierarchical levels of the datasets more effectively.

2 Related work

Navigation is an important phase in the interactive space-filling process. It involves moving from one view to another view, along the existing geometrical structure of the graph. When the size of a graph becomes large, it is desirable to allow users to have an easy access to contextual information through the navigational views. One of the most important issues involved in navigation is that users should always be able to review as much contextual information as possible from the navigational views. This allows users to maintain the perception of where they are and where they have moved from during their navigation. This also assists users to make more accurate decisions about where they should go next to reach the target view. At the target view, users can view the content and immediate relational structure of the target.

The most commonly used interactive navigation technique in space-filling visualization is drilling down + semantic zooming (Baker and Eick 1995; Bederson 2001; McGuffin et al. 2004; Smartmoney Marketmap 2007), a quick and simple navigation scheme that enlarges the display of a specific portion of the graph allowing users to view the detail of this substructure of the graph. It has been widely established by file and directory explorers provided in most current operating systems. This form of interaction is analogous to zooming into a region of interest with each step of the zoom operation being a substructure in the hierarchy (Fig. 1).

However, under this navigation scheme, users are usually provided with very limited knowledge about where to go and where they have been by extracting contextual information from only one view. Users could spend a significant amount of time browsing for specific items in hierarchies because of the lack of contextual information. With the drill-down approach the number of "unnecessary views" a user may take to reach the specific item could be significantly increased because in this approach, traversing each

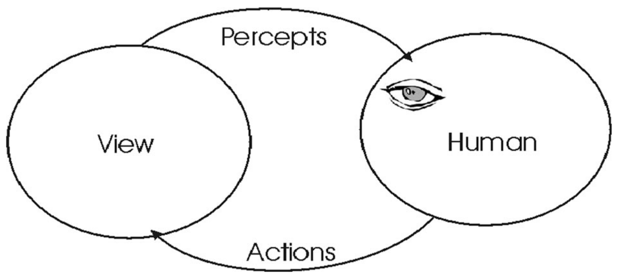

Fig. 1 Interaction process in drilling-down + semantic-zooming approach, in which the user can only perceive contextual information from one view

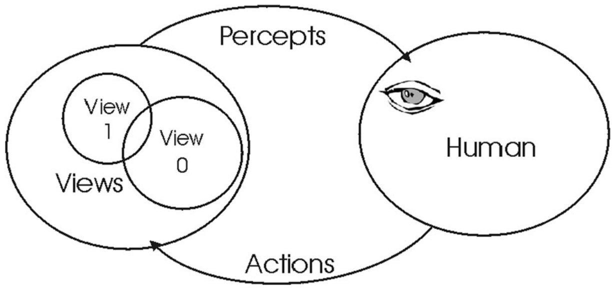

Fig. 2 Interaction process in focus + context approach, in which the user can percept contextual information from two views

successive layer requires abandoning the previous views. The lack of contextual information in navigation could negatively impact performance as a user has to reform or re-establish their mental map of the relational structure to determine where they have been and where they should to go to find specific items.

To address the problem of lacking of contexture information, some space-filling techniques have used focus + context viewing to provide users with a detail view of a focused sub-graph as well as a global view of the overall graph to maintain user orientation. Typical techniques in focus + context space-filling visualization include Sunburst (Stasko and Zhang 2000), Information Slices (Andrews et al. 1998), Fisheye + Zooming (Shi et al. 2005), Layering + Transparency (Lieberman 1994; Nguyen and Huang 2004) and Multiple Views (Baldonado et al. 2000; North and Shneiderman 2000).

In comparison with the drilling-down + semantic-zooming technique, the focus + context is a better solution as it provides users with more contextual information by displaying two views simultaneously. This enables users to make more accurate decisions on where they should go next (Fig. 2).

Some current techniques, such as SpaceTree (Plaisant et al. 2002), Collapsible Cylindrical Trees (Dachselt et al. 2001), can achieve the focus + context interaction of trees by showing the path from the current focal node to the root, and siblings of each node along the path (Lee et al. 2006a). However, these techniques provide only the display of one level of structure in their context view. Although context views can be provided by using simple thumbnails, such as PadPrints (Hightower et al. 1998) and Microsoft Explorer (the folder explorer bar for historical context) and Microsoft PowerPoint (the thumbnails at the left side), these techniques, however, might not be quite efficient to display a long sequence of context views. This is because the size equality of the thumbnails makes it impossible to show concurrently a large number of views without reducing the thumbnails' sizes or using a scroll bar. However, the reduction in size can decrease the amount of information that is displayed, while the use of scroll bar can also hide some of the context information in which users need to scroll up or down to view further detail.

However, as real-world graphs are often very large with thousands or even millions of items, interactive exploration could easily move through more than ten hierarchical levels to reach the target view. The display of small numbers of views, one current view and a context view (or a small number of context views), could sometimes be insufficient for users to gain enough contextual information to make further decisions on where they should go next. Therefore, contextual information produced for navigation of data in traditional focus + context viewing techniques is still limited, especially for navigating through a large and complex relational data structure with many levels.

3 Proposed solutions

In this paper, we propose an efficient interaction method called chain-context view (CCV). The new solution should overcome these limitations and offers a history of user's actions while interacting with visualization system. Inspired by existing ideas, we aim to provide a new layout algorithm, which provide both focus view and context views with space optimization, and a new navigation scheme to guide users to explore focus of interests in large hierarchical structured data.

Chain-context view (CCV) for the navigation of large hierarchies (or clustered graphs) in space-filling visualizations. Instead of displaying a single or small number of context views as in traditional approaches, we provide users with the display of a progressive sequence of context views attempting to maximize the displayed contextual information. The display of rich context information produced in the exploration path could greatly increase the accuracy of user decisions and reduce the unsuccessful trips and unnecessary

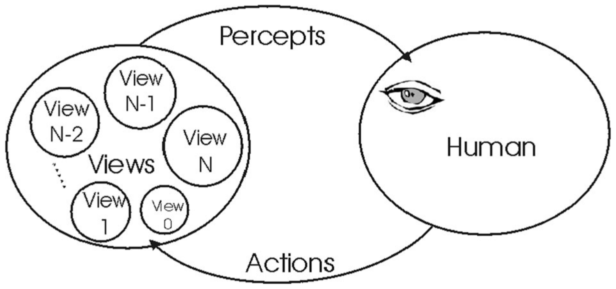

Fig. 3 Interaction processes in chain-context approach, in which the user can perceive rich contextual information from multiple views

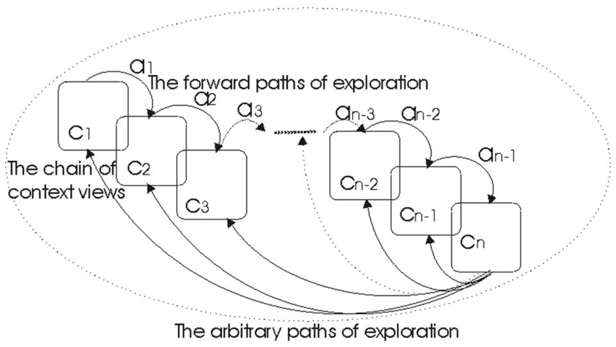
a1
The forward paths of exploration
a2
a3
an-3
an-2
an-1
C1
C2
C3
Cn-2
Cn-1
Cn
The chain of context views
The arbitrary paths of exploration

Fig. 4 Chain-context view enables user to make arbitrary paths of exploration to reach the target. Users can easily view, jump or return to any level of the hierarchy they have been

views during visual exploration of large hierarchies. This also allows users to trace each step of their interaction and makes it easy for them to jump or return to any level of the hierarchy they have already visited (Fig. 3).

Similar to Spiral Calendar (Card et al. 1994), we also archive the focus + context visualization by providing a sequence of all intermediate context views at each navigation step. The CCV maintains the user's orientation during exploration by providing multiple navigational views. This also enables users to explore large graphs by arbitrary paths while moving towards the target view (Fig. 4). In comparison, traditional approaches only allow sequential navigation; that is from a starting view move towards or backwards to the target view through a series of mouse clicks, the proposed navigation scheme is much more efficient.

Although theoretically, this technique is applicable to any layout algorithm, we applied it only to the space-filling visualization where space utilization is one of the key objectives. We used a modified EncCon, a space-filling layout algorithm (Nguyen and Huang 2005), in the prototype. The browsing proceeds by arbitrarily clicking on a selected node (or region) from either the main view or one of the context views.

The CCV only occupies a small portion of the display area, but it provides users with sufficient context information to guide their navigation through deep relational structures. The size and aspect ratio of these rectangular context views are optimized to ensure space efficiency. Visual cues are also employed to enhance the visualization in context windows. Additionally, Interaction techniques are applied to each interaction in order to maintain the user mental orientation of views.

The main contributions of this work are

- The solution maximizes screen space utilization, by optimizing context views within a small area, which provide multiple levels of structures.
- The solution provides both a focus view and an entire chain of navigation history, which guide users to browse and locate the target object within deep hierarchical structured data with minimal steps.

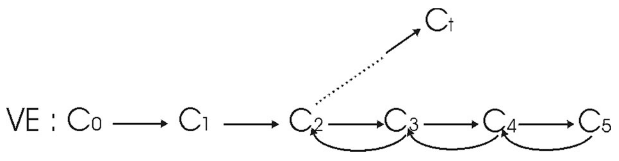

Fig. 5 Example of selecting visual exploration paths

4 The concept of chain-context view

4.1 The visual exploration model

Visual exploration (VE) is a process that conducted by the users, with the assistance of visualization, where a particular data item in a large visualization is located. It is then retrieved and its content and environment is inspected.

The goal is to provide tools for users to efficiently explore a large graph $G = (V, E)$ with a shortest path (minimized number of view transformations) to reach his expected view of a particular item. The visual exploration of the graph G consists of a human, a sequence of context views $C = (C_0, C_1, \ldots, C_n)$ and a sequence of actions, e.g. mouse clicks, $A = (a_0, a_2, \ldots, a_{n-1})$. We assume that a context view $C_i = (G_i, D_i)$ consists of a sub-graph $G_i = (V_i, E_i)$ of G and a drawing D_i of G_i consisting of a location for each node $v \in V$ and a route for each edge $e \in E$. It is also assumed that the exploration of G proceeds by making a sequence of possible user actions (e.g. mouse clicks), which transform the state of the context view (see an example at Fig. 4). A "successful trip" is a VE in which the final view C_n is equal to the target view C_t.

4.2 Unsuccessful trips and unnecessary views

We aim to navigate large graphs efficiently with minimal steps. This can be achieved through the reduction in "unsuccessful trips" and "unnecessary views". During the navigation of a graph, an "unsuccessful trip" occurs when a user reaches the final view C_n of a VE, and he confirms that C_n is not the expected target view C_t. He/she also believes that it is unlikely that he/she will be able to reach the target view C_t by taking further actions along the current path. Therefore, the user withdraws the VE. In other words, an "unsuccessful trip" is a withdrawn VE.

In visual exploration of a graph, we aim to efficiently interact with the visualization to reach the target view C_t. In particular, we always attempt to find the shortest path from the current view C_i, where we are now, to the target view C_t. Suppose that the shortest path between C_i and C_t consists a sequence of intermediate views, $\{C_{i+1}, C_{i+2}, C_{i+3}, \ldots, C_{t-1}\}$. Thus, an "unnecessary view" is defined as a view C that is not in the shortest path and satisfies: $C \notin \{C_{i+1}, C_{i+2}, C_{i+3}, \ldots, C_{t-1}\}$. The efficiency of visual interactions can be measured by the number of "unnecessary views" contained in successful trips. For example, a user has reached view C_5 and confirmed that it is not the target view he/she is expecting. Thus, he/she has to adjust the path to find out the target, see Fig. 5.

If the visualization is under the drilling-down + semantic-zooming or focus + context interaction scheme, a user would have to then return back along the path $(C_4, C_3, C_2, \ldots, C_{t-1})$ to reach the target view C_t because at a time, he could only perceive information from one or two views. However, if the visualization is under the CCV scheme, then he could be able to find the shortest path, which consists of views $(C_2, \ldots, C_t)$ to reach the target. This is because C_2 is displayed as part of the CCV. Since views C_3 and C_4 are not in the shortest path between C_5 and C_t, so we could call them "unnecessary views" for browsing from C_5 to C_t. Overall, in comparison with three interaction methods, CCV approach tends to be a more efficient solution, in terms of preventing unsuccessful trips and unnecessary views by displaying more contextual information in the visualization.

5 Technical specifications

5.1 Screen partitioning

The entire display area is initially divided into two regions including a small area in the left-hand side for displaying context views and a large area in the right-hand side for the main view (or focus view). The main

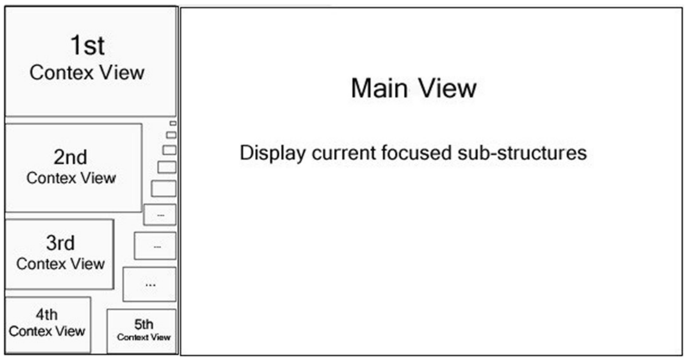

Fig. 6 Partitioning of views in the visualization system

view displays a current focus sub-graph G_i or the complete graph G at the beginning of a VE (Fig. 6), and it occupies the main portion of the entire display and is responsible for showing users as much detail of the structure as possible. This allows them to inspect particular data items and analyse the structure of G_i for taking further actions.

A chain of context views is displayed inside a small rectangular region with efficient partitioning. It enables users to perceive additional contextual information that could increase the accuracy of decision when taking further actions. The display of an intermediate chain of context views during the navigation also enables users to choose an arbitrary path for reaching the target quickly and efficiently. While the CCV is responsible to display clearly as much contextual information as possible, it should not occupy the major portion of display space. In the implementation with a typical widescreen of 1280 × 800 pixels, the width of the entire chain-context region is defaulted with 300 pixels. Some details of the substructures displayed in the context views will be filtered through visual abstraction. Only the main structures remain for guiding navigation. The abstracted context views aim to maintain users' orientation of where they are, where they came from and where they have been during visual exploration.

5.2 The layout of chain-context view

The layout algorithm in the CCV calculates the size and aspect ratio between rectangular boxes displaying all context view boxes with optimal layouts in the region. In the representation, each context view displays a simplified abstract structure of a sub-graph at each navigation point. Although the number of context views in the chain is theoretically unlimited, we limit the maximum number of the context views to be displayed to 20. This is because most of the hierarchies we used in the experiments have fewer than 20 levels.

The current main view C_{n+1} (the focus view) is displayed at the main region which is already defined. We are only concerned with the calculation of boxes $\{B_0, B_1,\ldots, B_n\}$ for displaying context views $\{C_0, C_1,\ldots, C_n\}$ in the small display region in which the context views are allocated in a reversed direction from the smallest box to the largest, i.e. $\{C_0 \rightarrow B_n, C_1 \rightarrow B_{n-1},\ldots, C_n \rightarrow B_0\}$. Suppose that n is the number of context view boxes $\{B_0, B_1,\ldots, B_n\}$ to be allocated, we need to place them in the context display area which is drawn as a thin rectangle in vertical direction. These generated context view boxes $\{B_0, B_1,\ldots, B_n\}$ must satisfy the following conditions:

The width–height ratio $\{R_0, R_1,\ldots, R_n\}$ of all context view boxes must be the same as the width–height ratio of the main view R. The width $\{W_0, W_1,\ldots, W_n\}$ of all context view boxes is uniformly decreased. Thus, the width W_{i+1} of B_{i+1} is defined by:

$$W_{i+1} = cW_i \tag{1}$$

where c is a constant and the value must within $0 < c < 1$, which indicates the magnitude of the difference between context views. The default value in the implementation is $c = 0.95$.

Suppose that W and H are the width and height of the entire context display region, the width of the first context view box W_{c0} is initially defined as: $W_0 = W$. Therefore, we can calculate the widths $\{W_1, W_2, \ldots, W_n\}$ for all context view boxes using Eq. 1.

The width–height ratio R of the main view can be easily calculated. The value of R could be varied depending on the size of the current adjustable visualization window. In the prototype system, the

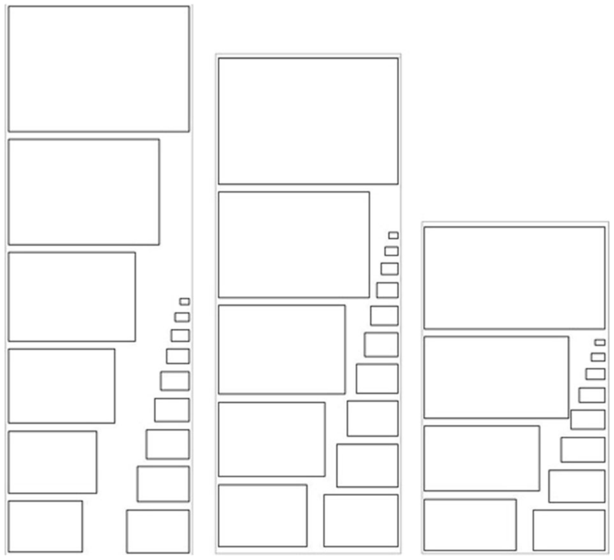

Fig. 7 Example of the layout implementing the partitioning algorithm with three different width–height ratios R1, R2 and R3

partitioning and calculation of context boxes are repeated accordingly when the value R is changed, i.e. corresponding to the window-resize event.

Suppose that at a state, the width–height ratio of the main view is R, and C_i is a context view, and the width W_i of a context view box is defined by Eq. 1, we can then easily calculate the $\mathrm{H_i}$ by formula:

$$H_i = \frac{W_i}{R}. \tag{2}$$

The first context view is located at the top-left corner of the context display region. The partitioning process first follows the top-down direction where context views are aligned along the left border. Then, when the partitioning process reaches the bottom of the region, the direction is reversed to bottom-up, where context views are aligned along the right border (Fig. 6). The number k of the context views that can be fitted into the context display region in the top-down direction is defined by formula:

$$\sum_{i=0}^{i=k} (H_i + d) \leq H < \sum_{i=0}^{i=k+1} (H_i + d) \tag{3}$$

where d is the default vertical distance between two context boxes. The height H_i of a context view box is linearly calculated based on the number of previous context view boxes using Eqs. 1 and 2. We can now easily calculate the widths $\{W_1, W_2,\ldots, W_k\}$ and heights $\{H_1, H_2,\ldots, H_k\}$ for all context view boxes in the top-down direction. The calculation of width $\{W_{k+1}, W_{k+2},\ldots, W_n\}$ and heights $\{H_{k+1}, H_{k+2},\ldots, H_n\}$ for remaining context view boxes aligned along the right border is done in the bottom-up manner. Therefore, the width W_{k+1} and height H_{k+i} of a context view box is also recursively defined using Eqs. 1 and 2.

The above partitioning process will repeat with a small magnitude's reduction in the constant c until the partitioning result satisfies the space-efficiency criteria. In order to ensure no overlaps among the boxes, the size of the context view boxes in the bottom-up direction will be decreased if an overlap occurs between two boxes at either the left-hand side or the right-hand side. In addition, we will ignore the context view boxes if they are overlapped with the first context view. Figure 7 shows three examples of implementing our partitioning algorithm with three different width–height ratios R_1, R_2 and R_3.

5.3 The chain of context views

In the visual exploration of a large hierarchy, the user interactively visits and inspects a sequence of sub-graphs $\{G_1, G_2, \ldots, G_n\}$ along the hierarchical structure one by one, moving from the parent G_i to the child G_{i+1}. The currently visited graph G_i is the sub-graph of the last visited graph G_{i-1}. Thus, we have $G_i \in G_{i-1}$. A chain-context consists of a number of context views, and each context view C_i displays a sub-

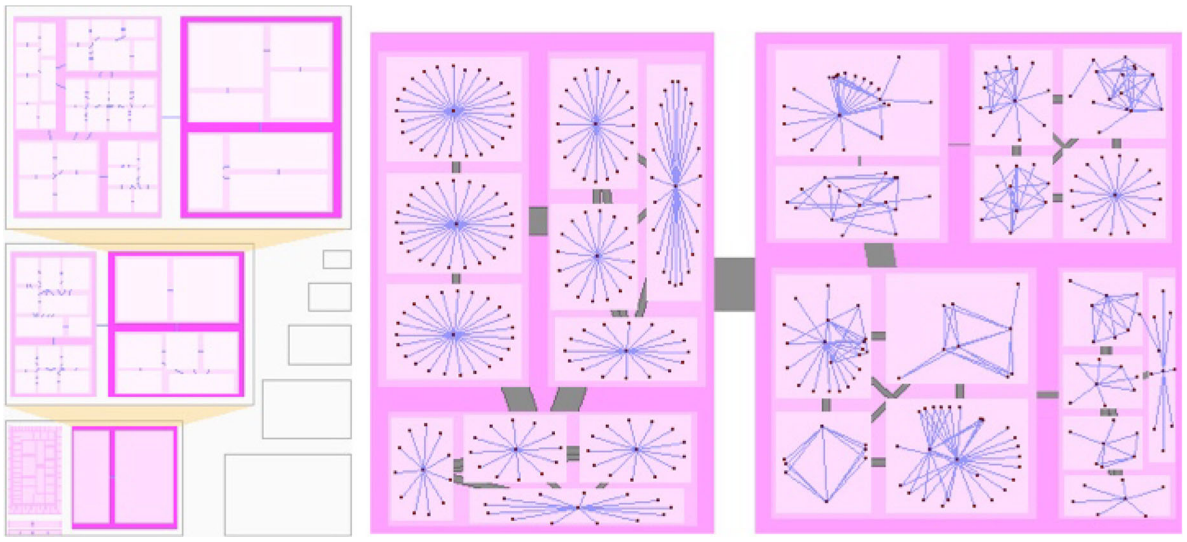

Fig. 8 Display of the chain-context view in modified EncCon visualization. It contains three levels of context views

graph G_i that was focused in the history of one particular navigation point. Therefore, the interrelationships among these context views $\{C_1, C_2,\ldots, C_n\}$ are parent–child relationships among the corresponding sub-graphs $\{G_1, G_2,\ldots, G_n\}$.

The context views $\{C_1, C_2,\ldots, C_n\}$ are displayed in the boxes $\{B_0, B_1,\ldots, B_n\}$. These are chained graphically by transparent lighting trapezoids. We use these transparent trapezoids to visually represent the interconnection between each pair of the neighbouring boxes B_i and B_{i+1} (Fig. 8). In the context chain, each box B_i is projected following the lighting trapezoid to its highlighted sub-region, in box B_{i+1}. This makes it much easier for users to identify the location of sub-graph G_i in its parent graph G_{i+1}.

5.4 Display and visual cues

Visual cues are used in the design of visual components to amplify the perception of contextual information displayed at each context view and the interrelations between the CCV. We use a modified EncCon space-filling algorithm (Nguyen and Huang 2005) to draw large hierarchies (or clustered graphs) in the visualization system. We use thin edges to show the relationships among nodes and thick edges to show the abstract relationships among clusters (Fig. 8). The display of context views is designed as follows:

Each CCV initially displays the top three levels of intermediate context structures in an abstract and simplified manner. This display aims to provide users with rich historic contextual information for navigation, while it only uses a small amount of the computational resources.

Colours are employed to assist users to easily perceive the hierarchical structure. In our prototype, the backgrounds of local regions (rectangular boxes) of non-leaf nodes are painted with the same colour tone but at different brightness. The colour brightness of a node is dependent on its hierarchical distance (or level) relatively to the focus sub-graph displayed in the current context view. The colour-tone's transition is from a light-tone colour to the white colour in our implementation. The colour scheme can be easily adjusted to suit with a particular user preference.

Other visual cues, including emphasized background colours and semi-transparency, are also used to highlight the historical focus sub-graphs and their relations among the intermediate context views. Technically, the historical focus sub-graphs displayed in the context chain are highlighted using a darker colour, i.e. we use the pink colour to display highlighted sub-graphs and the light-pink colour for the display of other sub-graphs display of a context view (Fig. 8). The coherence of the interconnections between context views is enhanced by using of the semi-transparent trapezoids connecting from a focused sub-graph in an upper level context view to a corresponding next context view box.

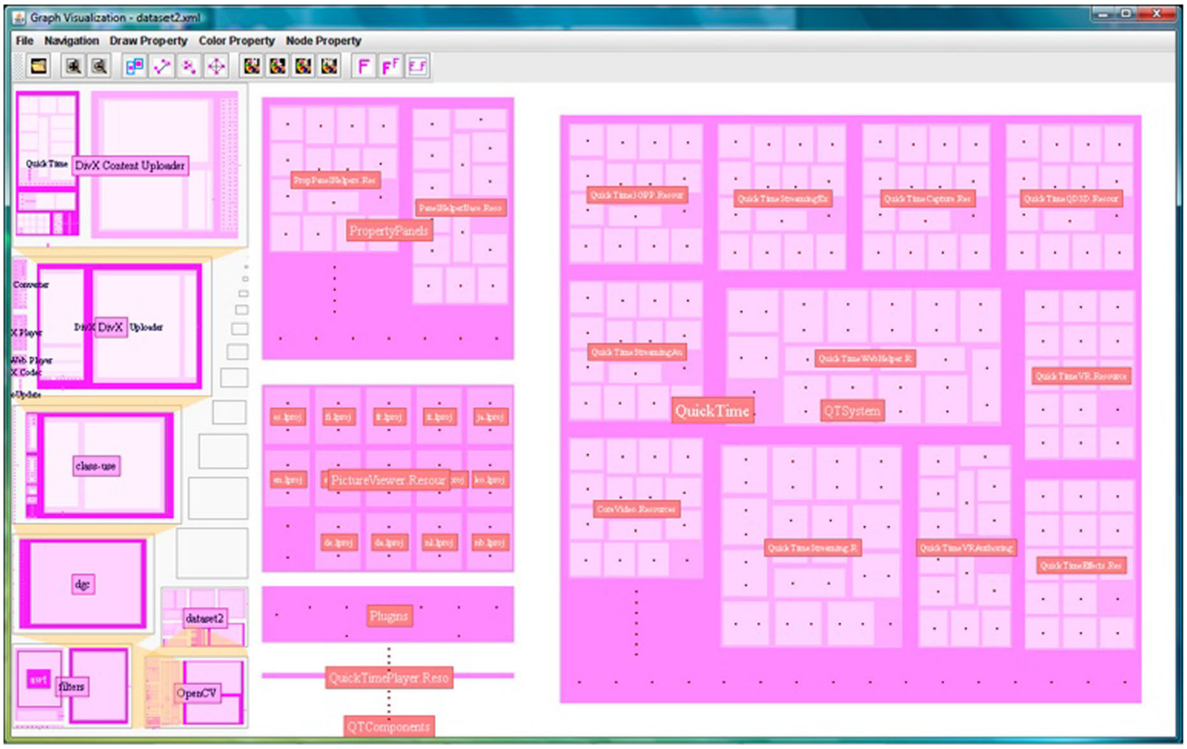

Fig. 9 Example of chain-context view visualization of a large file system with seven context views and a main focus view

5.5 Interactive view transformation

The interactive navigation in the visualization system is achieved by arbitrary mouse clicking on either the main view or any of context views. We use semantic zooming to enlarge the display of a particular sub-graph at a navigational point. In short, by selecting a sub-graph in the context view, it will expand and occupy the entire main-view display region. The context views in the chain will then be relocated. The previous main view is now added as the first context view to the head of the chain (Figs. 9, 10). All transactions are achieved in accommodation of multiple Interaction schemes to preserve the users' mental maps of views.

The navigation is taken by a mouse clicking onto a sub-graph G_i or its local region located in a box B_j which is defined in Sect. 5.2. The direction of an interaction can be either drilled down, i.e. a forward path of exploration, or arbitrarily rolled up, i.e. the arbitrary paths of exploration (Fig. 4). When a mouse clicks onto a box B_{i+1}, the corresponding focus view will transform to the last context view C_{n+1} display in B_1 and the selected sub-graph G_{i+1} will become the focus graph and the corresponding view C_{i+1} will transform to the main view. Each context view C_k in the chain will be transformed to C_{k+1}, and its location will be moved from B_j to B_{j-1}. The navigational action can also be taken on any of context view in order to recall the context information for making arbitrary steps to quickly reach the target view.

Animated drilling-down view transformations—this process occurs when a sub-graph G_{n+1} of the current focus graph G_n is selected in the main view. First, the views $\{C_0, C_1,\ldots, C_{n-1}\}$ in the context chain are gradually rescaled and moved to the next smaller context view boxes $\{B_n, B_{n-1},\ldots, B_1\}$, respectively. Concurrently, the size of the current focus view C_n is smoothly reduced and it is then moved to the first context view box B_1. The selected sub-graph G_{n+1} is then expanded gradually to the size of main view and transformed into the new focus graph occupying the entire the main-view box B_m (Fig. 11). During the above view transformations, four types of Interactions, include movement interaction, expansion interaction, rescaling interaction and fade-out interaction, are employed to reduce the cognition overheads in identifying the change of views. The following is the pseudocode for drilling-down navigation corresponding to the transformations of context views $C = \{C_0, C_1,\ldots, C_n, C_{n+1}\}$. Note that C_{n+1} contains graph G_{n+1} which is a selected sub-graph in C_n.

Algorithim 1

```
procedure drill-down (CtextViews [] C, Graph [] G, Sub-Graph $G_{n+1}$)
begin
   add the new context view into C;
   add the sub-graph $G_{n+1}$ into G;
   associate the last view $C_{n+1}$ in C to the selected sub-graph $G_{n+1}$
   redraw the all context views in C;
   {$G_0$, $G_1$, ..., $G_n$} is allocated at the corresponding {$C_0$, $C_1$, ..., $C_n$}
end {drill-down}
```

Animated rolling-up view transformations—this process occurs when a sub-graph G_{i+1} of G_i displayed at one of the context view boxes B_j is selected. First, the display C_n of the current focus graph G_n in the main-view box B_m fades out gradually into the background. Concurrently, the chain of context views $\{C_0, C_1, \ldots, C_i\}$ rescales and moves smoothly to the new smaller boxes $\{B_i, B_{i-1}, \ldots, B_0\}$, respectively (Fig. 11). The sub-region of selected graph G_{i+1} in B_j is then transformed to the main view and expanded smoothly to the entire main-view area. Similar to the drilling-down process, every view transformation is accommodated with four types of interactions to reduce the cognitive overheads in tracking changes. The rolling-up view transformation also allows users to interactively select and enlarge a context view in which more information can be displayed. This property aims to provide a quick analysis at context views without showing these views at the focus panel. The following is the pseudocode for arbitrary rolling-up navigation corresponding to the transformation of context views $C = \{C_1, C_2, \ldots, C_n, C_{n+1}\}$. Note that C_{n+1} contains graph G_{n+1} which is a selected sub-graph in C_n.

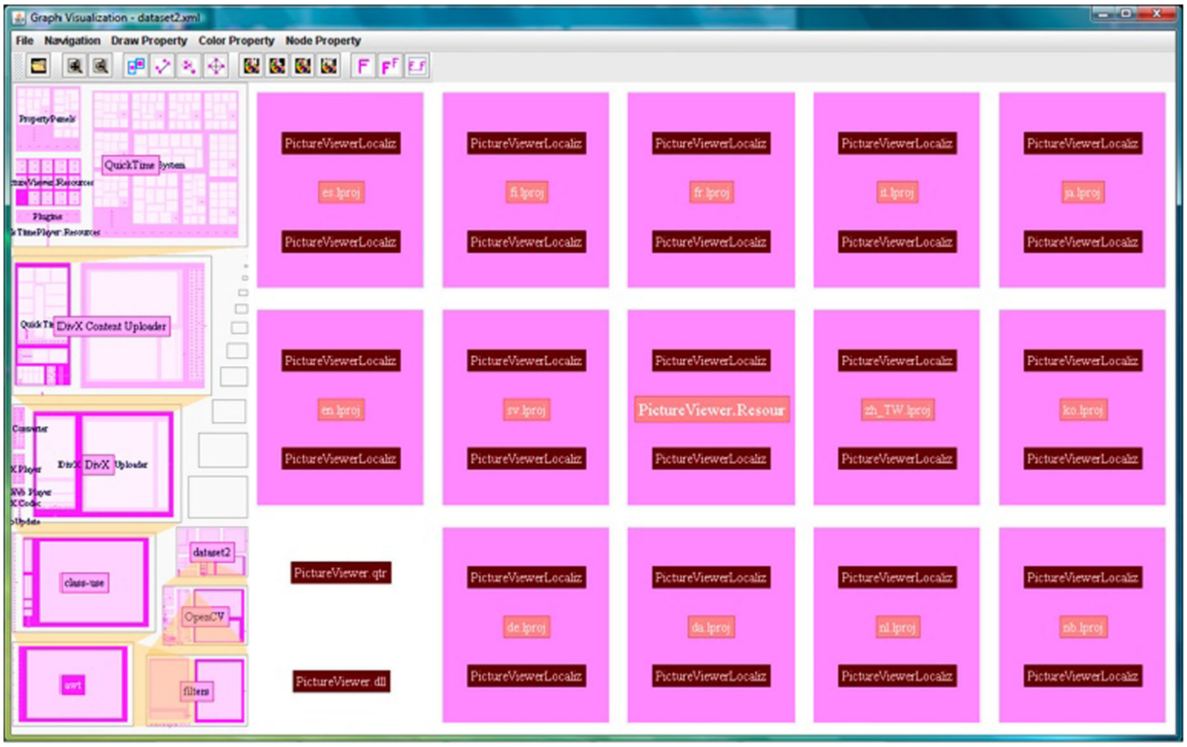

Fig. 10 Example of the view transformation when a sub-graph located at the *left-hand side* of focused graph in main view of Fig. 10 is selected. The selected sub-graph is gradually enlarged, and it occupies the main view

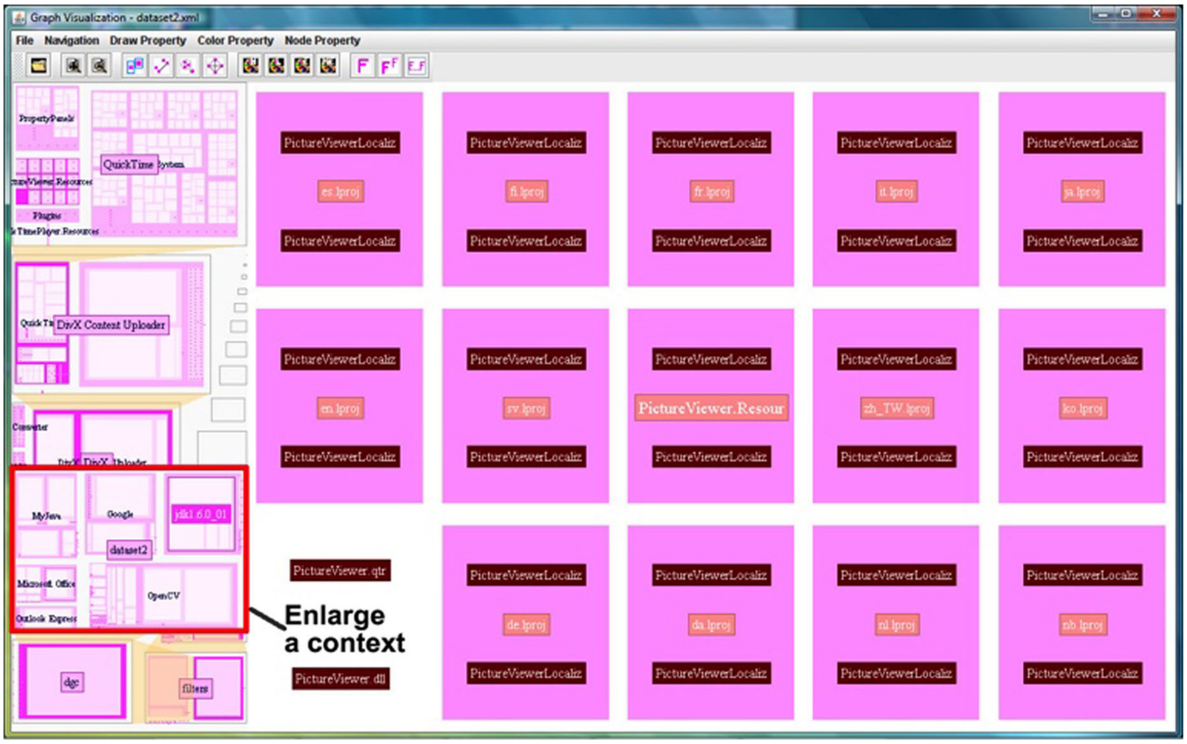

Fig. 11 Example of enlarge and navigate from a context view, i.e. structure of "dataset 2"

Algorithm 2

```
procedure roll-up (CtextViews [] C, Graph [] G, Sub-Graph Gi, Sub-Graph Gi+l)
begin
  decrease the size of context views in C by n – I;
  decrease the size of sub-graph in G by by n – I;
  add the new context view into C;
  add the sub-graph Gi+1 into G;
  associate the last view Ci+1 in C to the selected sub-graph Gi+l
  redraw the all context views C
end {roll-up}
```

Figure 9 shows a screen dump of CCV visualization of a large file system. It contains seven context views displaying on the left-hand side and a main view displaying a focus sub-graph on the right-hand side. The current file structure is visualized, such as there are six directories in which the directory on the right-hand side has a large portion of files and directories. Figure 10 shows the outcome of a view transformation after the user clicks on the second top left-hand side directory in the main view from Fig. 9. Then the clicked directory is expanded and transformed to the focus graph where it occupies the entire display area of main-view box. The expended sub-graph is now shown much more details. The previous seven context views are scaled down and shifted into the smaller view regions. The previous main view is now reduced in size and displayed in the first context view box at the top-left corner panel. Figure 11 is another example of enlarging a context for showing further detail in which a substructure is selected for different navigational moves.

5.6 Interaction design

We aim to create smooth view transformations to preserve users' mental map of view changes and maintain their orientation of navigation. To achieve this, all view transformations done in our prototype are accommodated by multiple Interactions. The Interaction process lasts about 1–1.5 s depending on the complexity of a particular view transformation. Several Interaction techniques are applied concurrently to all types of view transactions. It is a real challenge to implement animated view transformations with low computational cost when exploring a large hierarchy. This is because that the high computational cost in transforming a sequence of views C_0, C_1, C_2,…, C_n, C_{n+1} could disrupt a smooth animation. Therefore, to reduce the computational cost without affect much on preserving the user's mental map of view changes, we only apply Interactions at the high levels of the hierarchical graphs. The Interaction provided in the system includes smooth moving, smooth expansion and/or rescaling and fade-out animation. These Interaction schemes are: *movement interaction*—enhance users' perception of moves of visual structures by using "slow in and slow out" animation rather than linear interpolations, in order to provide users with a more pleasant feel at the beginning and the end of a view transformation. In "slow in and slow out", the action starts slowly and then gradually moves faster at the middle state and then slows down gradually at the end of the Interaction process. This process naturally follows the human perception process, that is, to provide users with enough time allowing them to gradually withdraw the old mental map and gradually re-form the new mental map of views.

Expansion interaction—applied for the semantic expansion of a small display region into a large display area. Since the computation for calculating the new layout at each Interaction loop could disrupt the smooth movement (affect to the speed of movement interaction), we only compute the final layout of the sub-graph and linearly change the size and position of the sub-graph according to their initial values. The intermediate size and position of a sub-graph are then calculated accordingly.

Rescaling interaction—applied for the smooth rescaling of a view to a new size. The layout of this view does not change in this Interaction.

Fade-out interaction—the display of the main view fades out smoothly into the background. In our implementation, the fade-out animation is a sequence of changes in colours, from the foreground colours to the background colours, in order to prevent the sudden loss of the mental map of views when we move to other views.

5.7 Fisheye browsing of chain-context view

The latest development is that we produced a fisheye interaction technique, as an alternative approach, for users to browse through the CCV. The user is now enabled to enlarge the size of any context view in the chain dynamically to review the detail. In contrast to the static approach in which the size of view boxes is decreased in order of B_0, B_1,…, B_n, the user could now select and enlarge any view box B_i to explore the detail of the view at any context level. A focus view box is first defined as a current mouse pointer, and then users can scroll down and up to change the focus point, and the position and size of all context views in the chain are then adjusted accordingly. The size of view boxes is gradually getting smaller when distance of a view box is further away from the focus point. The technique utilizes the display space at both left and right sides for displaying the full chain of context view (Fig. 12).

Different display properties based on the point of interest are also applied to visualize the CCV during the fisheye browsing. In general, the focus context is displayed with full detail and the less detail for context whose position is far away from the focus context (Fig. 12).

6 Usability evaluations

We conducted a within-subjects usability study with three experiments to assess the effectiveness of the CCV. The study aimed to measure user performance on browsing and locating specific task while navigating deep and large relational datasets with this new viewing technique. To compare with the traditional methods, users were required to complete four types of tasks by using three navigation methods: chain-context view(CVV), focus-context view(FCV) and no-context view(NCV). Our hypothesis was CCV which provides more contextual information, facilitates users more efficiently and effectively for searching and navigating within the large data structures, than other two methods.

6.1 Control group

This usability study invited fifteen subjects including nine males and six females, from various knowledge backgrounds, IT and mathematics and engineering and accounting, etc. They were students, and professionals who deal with data analysis every day, who were familiar with the concept of file and directory structures, but none had any previous experience using the interactive navigation methods and were not

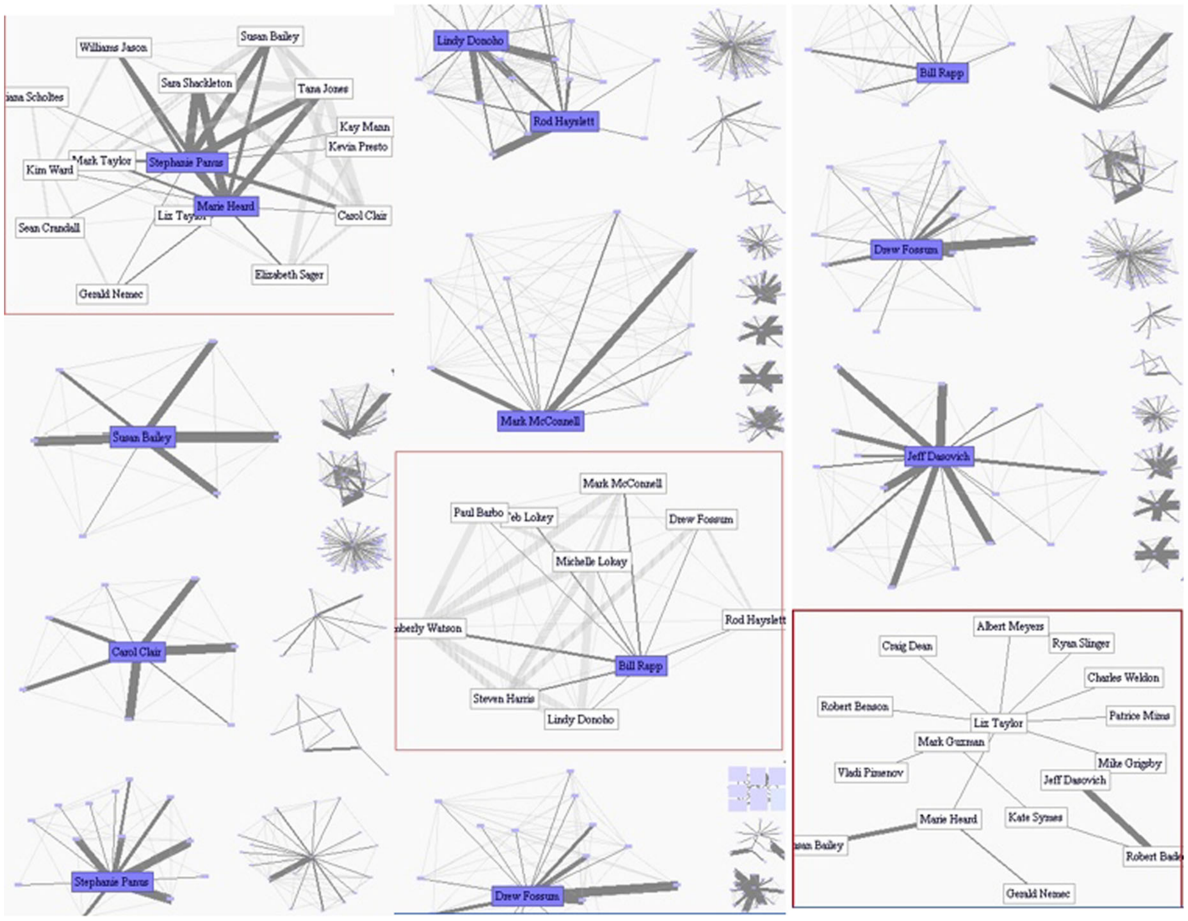

Fig. 12 Example of fisheye browsing of chain-context view when **a** the focus is at the *top box*, **b** the focus is at a *random point*, and **c** the focus is at the *bottom box*

familiar with space-filling concepts. To stimulate their motivation to participate into questionnaires, subjects were given prizes according to their performance ranking.

6.2 Experiment design and procedures

Three control experiments were carried out in sequence with a rest gap time. Each experiment consists of three separate questionnaires using different datasets. We generated datasets by controlling depth of the hierarchy, balance of the tree, number of leaf nodes and number of total nodes. The data selection is based on the number of hierarchical levels, while ensuring similarity of total size and the semantic complexity. Nine datasets were carefully selected to ensure similarity in terms of the number of levels to be traversed and semantic complexity. There were real file system datasets with around 12,000 files and 1200 folders with 13 levels. In each experiment, all subjects were required to conduct three questionnaires, by using one navigation techniques in random order.

The user tasks have been defined as browsing and locating tasks, referring to the categorized tasks in task taxonomy for graph visualization (Lee et al. 2006b). Each questionnaire contained four types of tasks: (1) Task 1: searching a node at a specific level of the structure, without given path; (2) Task 2: locating a node with a given path; (3) Task 3: returning to the visited node in Task 1; and (4) Task 4: listing file directory of the node. Each questionnaires contain eight questions, two questions for each task (Table 1).

Experiments were conducted on a standard 14-inch-wide screen monitor, with resolution of 1280 $\times$ 800 pixels. Subjects were seated 80 cm from the monitor. The sequence of evaluated techniques was counter-balanced, and the sequence of the questions per technique was given according to the increase in difficulty degree and complexity of data. Before experiments, subjects were required to take practices tasks on the test

Table 1 Sample questions in the questionnaires

Type of tasks	Sample of questions
Task 1: search target without given path	Find an object named "NODE 1" at level 3 of hierarchy
Task 2: locate target with given path	Return to main directory dataset1 and find "NODE 3" with the indicated pathway as [dataset1 → WinZip → jdk1.6.0_01 → demo → plugin → jfc → Java2D → src → java2d → demos → Fonts → NODE 3]
Task 3: return to pre-visited node	Please fill out three parent directory of NODE 4 _ → _ → _ → NODE 4
Task 4: list the path	Please return to "NODE 1"

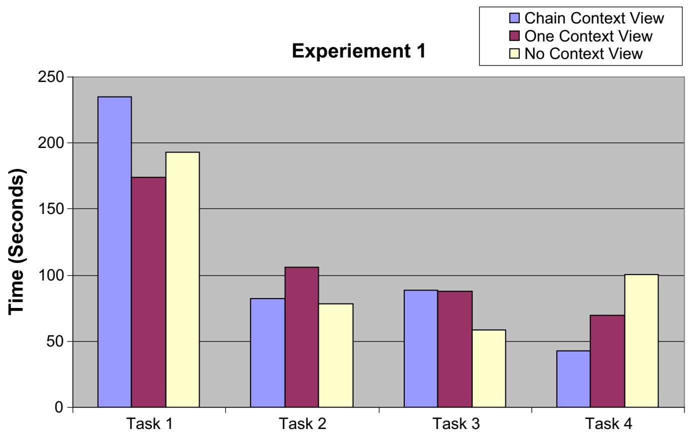

Fig. 13 Mean response times in experiment 1

data, until each subject was familiarized with tasks and file visualization system. During the questionnaires, the user reported where the targets were located to the experimenter, who noted it. Task completion time corresponded to the duration was measured. The reading time and writing time are not counted in completion time. Subjects were allowed to give up or stop the trial, if the time exceeded 5 min in a certain question. After each condition, the users were required to assess their subjective satisfaction with the technique for each type of tasks in experiments. Upon completion of the experiment, they offered feedback and rate their overall preference in a semi-structured interview. In total, evaluation involves 1080 trails (15 subjects, 3 experiments, 3 questionnaires, 8 questions) and 45 interviews (15 subjects, 3 experiments).

6.3 Performance results and user confidence

Efficiency of visual interactions should be measured by the exact numbers of "unnecessary views" in "unsuccessful trips". However, it would be unrealistic to measure the number of unnecessary trips during the experiments. We used the responding times to measure the performance. Surprisingly, the outcome of experiment 1 showed that the completion time in NCV and FCV for Task 1 to Task 3 was overall shorter than CCV (Fig. 13). CCV only performed better in Task 4, with the advantage of full historical context information. In the interviews, subjects pointed out that unfamiliarity with the visualization and the navigation schemes influence their performance.

Figures 14 and 15 provide an overview of user performance in three experiments. Experiment 2 showed that the efficiency of all three techniques improved (Fig. 14). Based on overall statistic analysis, the performances of CCV were improved by 25 %, much more than other two techniques which only average 5.5 % (FCV is slightly better than NCV). It implies that the user performance improves much faster in than other two techniques. In experiment 3, subjects with CCV ($\mu = 269.50$, $\sigma = 154.86$) completed all tasks significantly faster than other two methods, with the average of 18.5 and 24 % increase in performance, respectively. FCV ($\mu = 316.00$, $\sigma = 186.22$) performed slightly better than NCV ($\mu = 493.00$, $\sigma = 189.54$) by 5.5 % in average. Based on ANOVA analysis, it indicated that subjects took shorter time for completing Task 1 using NCV, but FCV performed better than no-context view in other three tasks

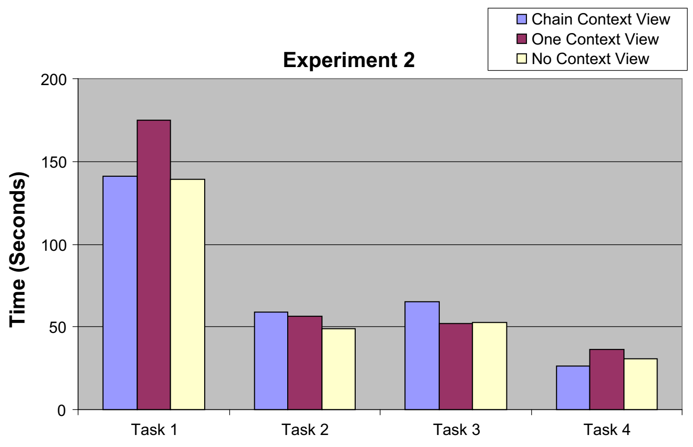

Fig. 14 Mean response times in experiment 2

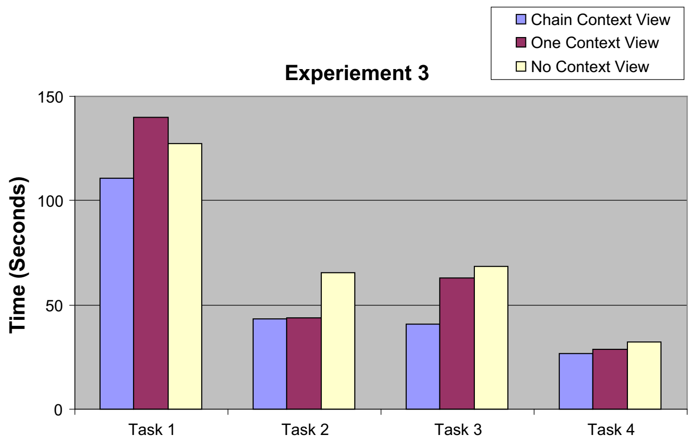

Fig. 15 Mean response times in experiment 3

(Fig. 15). The results have proven our hypothesis; however, adequate time for training is required in order to achieve the outstanding results.

Effectiveness is measured by evaluating if context information has enhanced the accuracy of users' decisions and comfort level of making decisions. Accuracy can be measured by correctness and giving up in questionnaires. For correctness, only two mistakes were found with NCV. For user's confidence, we recorded the frustration level by giving up trials while using three methods. We observed that most subjects showed the higher frustration about finding a target in NCV and FCV by verbal expression and body language, and easily gave up in the questionnaires in NCV and FCV than CCV. Total trials of giving up were 15 for NCV, 12 for FCV and 6 for CCV. These results suggest that CCV can effectively support interactive visualization and reveal the data structure efficiently, as the experiment results showed that subjects completed the tasks faster and with fewer errors with the use of CCV. The relatively smaller size dataset in experiments implies that the benefit of full context navigation will be amplified along with increasing density and complex of dataset. In terms of user satisfaction, subjects were asked to rate their opinions as to whether using CCV, FCV or NCV. We would like to obtain user's preference among three techniques for navigating large structures. With a five-point Likert scale (5—Very Good, 4—Good Barely, 3—Acceptable, 2—Poor, 1—Very Poor), subject's average ratings for CCV, FCV and NCV were 3.6, 3.2 and 3.2, respectively; CCV technique has the highest user satisfaction in comparison with FCV and NCV.

7 Conclusion and future work

Interaction scheme is a very important component in visualization systems for exploring large structures. This paper presented a CCV interactive visualization technique for navigating the large datasets. We proposed a chain of context views to provide user with rich contextual information. The awareness and perception of rich contextual information provided in a series of historical context views could increase accuracy of user's decisions on taking further actions. This could greatly reduce the number of "unsuccessful trips" and "unnecessary views" towards the efficiency of navigation processes. A usability study has also been conducted to evaluate the effectiveness and efficiency of the CCV, by measuring the user performance and confidence while navigating through large relational structures.

We are currently generalizing our technique for the interactive visualization of general graphs and other types of datasets. For future work, we will investigate the filtering schemes to achieve the visual abstraction in view transformations. We will also investigate optimal algorithms to implement efficient interaction for view transformations and effective context browsing technique of datasets with millions of items.

Acknowledgments This work was supported by Australia ARC Discovery research Grant #DP0665463.

References

Abello J, Kobourov SG, Yusufov R (2004) Visualizing large graphs with compound-fisheye views and treemaps. In: Proceedings of graph drawing, pp 431–441

Andrews K, Heidegger H (1998) Information slices: visualising and exploring large hierarchies using cascading, semi-circular discs. In: *Proceedings of late-breaking hot topics—IEEE symposium on information visualization*, pp 9–12

Baker MJ, Eick SG (1995) Space-filling software visualization. J Vis Lang Comput 6(1):119–133

Baldonado MQW, Woodruff A, Kuchinsky A (2000) Guidelines for using multiple views in information visualization. In: Proceedings of advanced visual interfaces, pp 110–119

Balzer M, Deussen O (2005) Voronoi treemaps. In: Proceedings of IEEE symposium on information visualization, pp 49–56

Bederson BB (2001) Quantum treemaps and bubblemaps for a zoomable image browser. In: Proceedings of user interface systems and technology, pp 71–80

Bederson BB, Shneiderman B, Wattenberg M (2002) Ordered and quantum treemaps: making effective use of 2D space to display hierarchies. ACM Trans Graph 21(4):833–854

Bruls M, Huizing K, van Wijk JJ (2000) Squarified treemaps. In: Proceeding of joint Eurographics and IEEE TCVG symposium on visualization, pp 33–42

Card SK, Pirolli P, Mackinlay JD (1994) The cost-of-knowledge characteristic function: display evaluation for direct-walk dynamic information visualizations. In: Proceedings of CHI'94, pp 238–244

Chen Y, Zhang X, Feng Y, Liang J, Chen H (2015) Sunburst with ordered nodes based on hierarchical clustering: a visual analyzing method for associated hierarchical pesticide residue data. J Vis 18:237–254

Dachselt R, Ebert J (2001) Collapsible cylindrical trees: a fast hierarchical navigation technique. In: Proceedings of IEEE symposium on information visualization, pp 79–86

Fekete JD, Plaisant C (2002) Interactive information visualization of a million items. In: Proceedings of IEEE symposium on information visualization, pp 117–123

Fekete JD, Wang D, Dang N, Aris A, Plaisant C (2003) Overlaying graph links on treemaps. In: Proceedings of information visualization 2003 symposium poster compendium, pp 82–83

Herman I, Melancon G, Marshall MS (2000) Graph visualization in information visualization: a survey. IEEE Trans Vis Comput Graph 6:24–44

Hightower RR, Ring LT, Helfman JI, Bederson BB, Hollan JD (1998) Graphical multiscale Web histories: a study of PadPrints. In: Proceedings of ACM conference on hypertex, pp 58–65

Jin L, Banks DC (1997) Tennisviewer: a browser for competition trees. IEEE Trans Comput Graph Appl 17(4):63–65

Johnson B, Shneiderman B (1991) Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In: Proceedings of IEEE visualization, pp 284–291

Lee B, Parr CS, Plaisant C, Bederson BB, Veksler VD, Gray WD, Kotfila C (2006a) TreePlus: interactive exploration of networks with enhanced tree layouts. IEEE Trans Vis Comput Graph 12(6):1414–1426

Lee B, Plaisant C, Parr CS, Fekete JD, Henry N (2006b) Task taxonomy for graph visualization. In: Proceedings of the 2006 AVI workshop on beyond time and errors: novel evaluation methods for information visualization, pp 1–5

Liang J, Nguyen QV, Simoff S, Huang M (2013) Visualizing large trees with divide and conquer partition. In: Proceedings of 6th international symposium on visual information communication and interaction, pp 79–87, Tianjin, 2013.8.17–2013.8.18

Lieberman H (1994) Powers of ten thousand: navigating in large information spaces. In: Proceedings of 7th annual ACM symposium on user interface software and technology, pp 15–16

McGuffin MJ, Davison G, Balakrishnan R (2004) Expand-ahead: a space-filling strategy for browsing trees. In: Proceedings of IEEE symposium on information visualization, pp 119–126

Nguyen QV, Huang ML (2003) Space-optimized tree: a connection + enclosure approach for the visualization of large hierarchies. J Inf Vis 2(1):3–15

Nguyen QV, Huang ML (2004) A Focus + context visualization technique using semi-transparency. In: Proceedings of international conference on computer and information technology, pp 101–108

Nguyen QV, Huang ML (2005) EncCon: an approach to constructing interactive visualization of large hierarchical data. J Inf Vis 4(1):1–21

North C, Shneiderman B (2000) Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In: Proceedings of advanced visual interfaces, pp 128–135

Plaisant C, Grosjean J, Bederson BB (2002) SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation. In: Proceedings of IEEE symposium on information visualization, pp 57–64

Shi K, Irani P, LiB (2005) An evaluation of content browsing techniques for hierarchical space-filling visualizations. In: Proceedings of IEEE symposium on information visualization, pp 81–88

Smartmoney Marketmap. http://www.smartmoney.com/marketmap. Accessed 6 March 2007

Stasko J, Zhang E (2000) Focus + context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In: Proceedings of IEEE symposium on information visualization, pp 57–65, 2000

van Wijk JJ, van de Wetering H (1999) Cushion treemaps: visualization of hierarchical information. In: Proceedings of IEEE symposium on information visualization, pp 73–78

Wattenberg MA (2006) Note on space-filling visualizations and space-filling curves. In: Proceedings of IEEE symposium on information visualization, pp 181–186

Zhao S, McGuffin MJ, Chignell MH (2005) Elastic hierarchies: combining treemaps and node-link diagrams. In: Proceedings of IEEE symposium on information visualization, pp 57–64