(To appear in Proceedings of OZCHI 2003, Brisbane, Australia, 26–28 Nov '03)

Side-By-Side Display and Control of Multiple Scenarios: Subjunctive Interfaces for Exploring Multi-Attribute Data

Aran Lunzer & Kasper Hornbæk
Natural Sciences ICT Competence Centre, University of Copenhagen
Universitetsparken 5, DK-2100 Copenhagen Ø, Denmark
{*alunzer, khornbaek*}*@nik.ku.dk*

Abstract

Information exploration often involves specifying alternative values for some set of parameters, and comparing the corresponding results. Some interfaces allow only one scenario, i.e., one set of parameter values, to be handled at a time. To compare results, the user must therefore switch repeatedly among the scenarios of interest and must remember details of the results seen so far. A subjunctive-interface approach may reduce this burden on the user. Subjunctive interfaces let users establish, view and adjust multiple scenarios in parallel, so that results can be compared side by side. As an illustration, we describe two subjunctive interfaces for comparing queries over a multi-attribute dataset. In both designs the query results are shown side by side, but in one case the input parameters' available values are laid out in menus, marked to show which queries use each value; in the other case the parameters are controlled by sliders, with the parameters' values in the different queries displayed side by side like the results. Both designs appear to offer advantages over other exploration interfaces, because they reduce the number of interface actions required and the information that users must remember.

1. Introduction

This paper describes a form of user-interface support for information exploration, designed to increase efficiency and decrease mental effort in tasks where the user must compare available results. The support is based on the coordinated use of multiple views, building upon earlier research into what we call subjunctive interfaces (Lunzer, 1999).

Interactive exploration of information (Waterworth and Chignell, 1991) forms part of numerous tasks – including the navigation of web sites, querying from databases, experimentation with simulations or spreadsheets, and exploratory design of artifacts. Typically, the tools available for such tasks provide results only in response to explicit, pinpoint specifications by the user. Exploration thus requires the user to undertake an iterative process of requesting and comparing results (Bates, 1989).

For example, interfaces for flight enquiries typically require the user to specify a single destination city. Although this is convenient for users with precisely formulated travel needs, someone who would prefer to compare the deals and schedules available for a range of destinations must embark on an exploration. This kind of exploration can burden the user in the following ways:

1. A high number of interface actions

   Making the different specifications needed to obtain the results may require many actions, e.g., mouse clicks or key presses. In the flight-booking task, having to submit separate queries for all the airport cities within a country or region may deter a user from pursuing a thorough search.

2. A need to remember earlier results

   When only one result is visible at a time, comparing results requires the user to remember the relevant details of those that are currently out of sight. In searching for flights, a user might even resort to writing down details of results before they disappear from the computer display.

3. Mental effort in organising the exploration

   In cases where the results of interest depend on variation in two or more parameters – for example, travel date as well as destination – the user must expend effort in working through the desired combinations of parameter values.

The flight-enquiry application is one example of an exploration where each result is obtained by specifying values for a fixed set of parameters. Existing visualisation

and interaction techniques addressing this class of explorations offer some assistance for the above problems. For example, the need for high numbers of separate interface actions is reduced by Dynamic Query techniques (Ahlberg and Shneiderman, 1994; Shneiderman, 1994), that let users rapidly work through different values for an input parameter by moving a slider. This technique can also reduce the user's mental effort, because different results that only differ in terms of a single input parameter can easily and quickly be revisited. It is less effective for exploring results that differ in terms of many parameters, because the user must still take care of organising a search through those parameters' individual ranges. This also remains a concern for other forms of dynamic result generation such as Magic Lenses (Bier et al., 1993) and Movable Filters (Fishkin and Stone, 1995). Goldstein and Roth (1994) demonstrated a simple combination of the Dynamic Query approach with their Aggregate Manipulator, which allows the user to establish and work with long-lived result subsets and thus also helps to organise the exploration and reduce the need for memory. The scenario-management facilities in Microsoft Excel® likewise allow a user to obtain a tabular view for comparing the results from user-defined scenarios involving different values in specified spreadsheet cells. However, these facilities in Excel and the Aggregate Manipulator can be seen as disrupting exploration progress, in that the user must spend time on the distinct activity of defining regions of the parameter space that are of interest.

This paper presents a subjunctive-interface approach to exploration within fixed parameter sets, specifically designed to reduce the three kinds of burden noted above. The principles of subjunctive interfaces are described in the next section, followed by a section illustrating how they are applied to information exploration within a small dataset from a US census. Finally we discuss how one may evaluate the benefit of subjunctive interfaces over other techniques.

2. Subjunctive interfaces

The idea of subjunctive interfaces is to allow the user of a computer application to establish and control alternative scenarios that differ in their values for the application's input parameters. The concept was inspired by Hofstadter's (1979) playful notion of a Subjunc-TV – a magical television whose tuning knobs would provide access to alternative versions of a given broadcast based on arbitrarily different circumstances chosen by the viewer.

2.1. Parallel scenarios in exploring information

In (Lunzer, 1999) it was suggested that subjunctive-interface principles may provide benefits in many different styles of application. Terry and Mynatt (2002) have demonstrated the principles' application in support of open-ended graphical design tasks; our goal here is to apply the principles to information exploration. The chosen approach is to use alternative scenarios to hold alternative queries, and to offer the following scenario-handling facilities:

- The user should be able to establish multiple, mutually incompatible queries based on different parameter values.

  Referring back to the flight-booking example, a user should be able to establish separate queries based on a range of destination cities.

- The results of all the queries should be viewable simultaneously, in a way that helps the user to compare them, and to see which results arise from which parameter values – for example, which fare applies to which city.

  Having the results continuously visible side by side allows the user to compare them without significant use of memory.

- At the user's discretion, any adjustment in a parameter value should be applicable to more than one query at a time. This would allow, for example, simultaneous exploration of the effect of different departure dates on the results for several different destinations.

  In general, this facility reduces both the number of interaction operations and the mental effort involved in organising a multiple-parameter exploration.

2.2. Handling many scenarios simultaneously

Enhancing an application to let users handle many scenarios in parallel brings an interface-design challenge: how to present all the scenarios' details so that a user can distinguish them from each other, and can confidently target particular scenarios for parameter adjustment.

In this respect the simplest applications to enhance are those in which differences between scenarios result directly in gross differences in spatial layout of their interface elements. In that case, multiple scenarios can be presented in parallel simply by overlaying their individual displays. Our early demonstrations of subjunctive-interface techniques addressed applications of this kind.

However, for many forms of complex graphical display, and for the countless applications whose displays include text, simple visual overlay would result in an unreadable mess. An alternative approach is needed.

One possibility is to render multiple copies of the entire application display. But naively replicating even those parts of the display that are identical in all scenarios is wasteful of space; furthermore, it may force the user to expend a substantial amount of time and effort in figuring out which aspects of the scenarios are actually different.

Our approach involves the use of what we call *widget multiplexers* (Lunzer, 2002). Each multiplexer takes over

the presentation and manipulation of the display area for a single interaction widget. If the display of that widget is the same in all currently defined scenarios, the multiplexer just shows that display in its normal, full-size form. But if the widget has different contents in different scenarios, the multiplexer shows those various contents side by side.

2.2.1. Showing values side by side

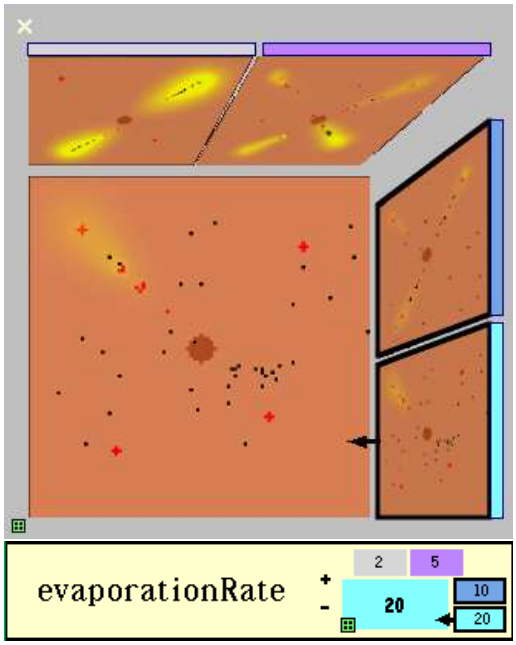

Figure 1: Graphical and textual widget multiplexers.

Figure 1 shows part of a simple application that simulates ants' food-foraging behaviour. In this case it is showing four different scenarios. In the large graphical multiplexer in the upper part of the picture, thumbnails for the four scenarios are laid out around the top and right. A second multiplexer, with textual contents, is seen in the figure's lower part.

The intention of the pseudo-3D visual distortion applied to the ant-simulation thumbnails is to help the user interpret them as all having the same aspect ratio, despite their being squeezed and stretched to fit the narrow margin around the larger *working view* (described below). However, at the limited resolution offered by today's typical computer screens, such distortion would make a textual display unreadable. Therefore for text-based displays we have developed the alternative form of widget multiplexer seen in the bottom-right of Figure 1, that does not employ this visual effect. The textual multiplexer in this example is showing four different values for the evaporation-rate parameter in the four scenarios. Notice that because the label 'evaporationRate' has identical appearance in all scenarios, there is no need to replicate its display.

Whether graphical or textual, all widget multiplexers are coordinated to lay out the scenarios in the same way. In Figure 1 the user can tell that the scenario whose simulation display appears at top left in the graphical multiplexer is the one with the value of 2 for evaporation rate. Colour coding is also used to reinforce this correspondence, as will be explained later.

In the discussions that follow, scenario positions within multiplexers are referred to by numbers – starting at top left and increasing in a clockwise direction. The style of layout shown here allows a maximum of eight scenarios to be shown.

2.2.2. Interacting with selected scenarios

At any given time, one scenario chosen by the user is designated the *primary scenario*. Each multiplexer highlights the primary scenario's thumbnail with a black border and an arrow indicating that its contents are mirrored into the working view. In Figure 1 the primary scenario is scenario 4, whose thumbnail appears at bottom right in each multiplexer. The user can select a different primary scenario by clicking on a different thumbnail.

Only the working view supports the interaction facilities (e.g., using mouse and keyboard) that the application would normally offer through this widget. Its size is a compromise between creating space for the thumbnails, and ensuring that the information in the primary scenario can be seen and manipulated at a comfortable scale.

The user can choose to have interactions in the working view affect many scenarios simultaneously. At any given time, the *active scenarios* – i.e., those that are influenced by interactions – comprise the primary scenario and any others that the user has selected by control-clicking on their thumbnails. In Figure 1, scenario 3 has been selected as an additional active scenario.

Having introduced these basic features of subjunctive interfaces, we now discuss how we have applied them to a specific information-exploration activity.

3. Interfaces for exploring multi-attribute data

We illustrate some of the design possibilities for subjunctive interfaces in exploring a census data set that was used by Hochheiser and Shneiderman (2000) for an experiment comparing menu-based interaction styles. First we show an interface similar to that designed for the original experiment, and outline its problems. We then show two alternative subjunctive-interface approaches to addressing these problems.

3.1. The simultaneous-menus interface

Hochheiser and Shneiderman worked with a small census dataset revealing commercial activity in the state of Maryland. For each of the state's twenty-three counties the set contains statistics for each of nine industries

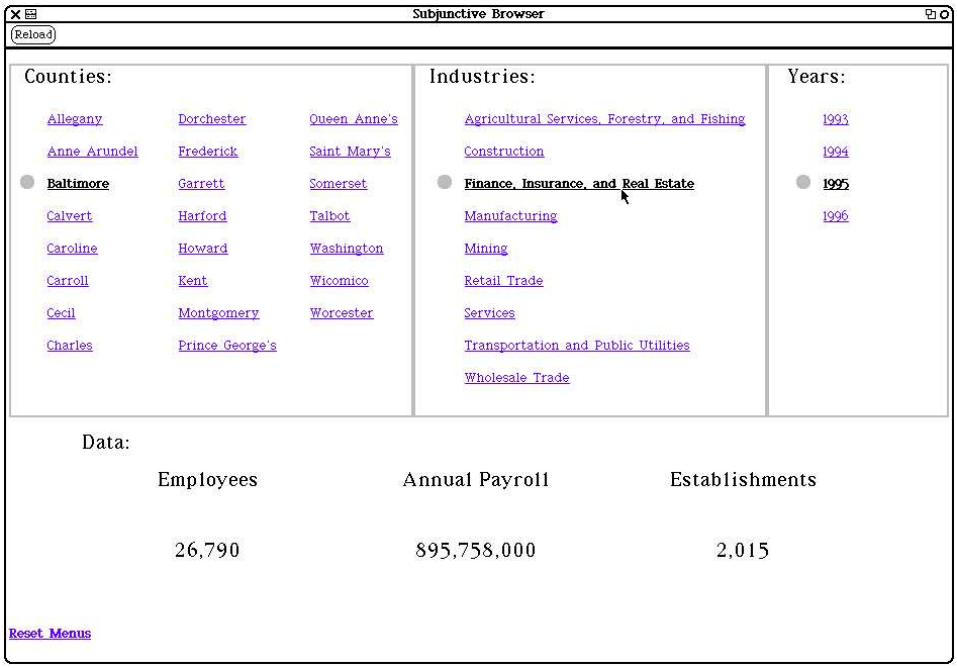

Figure 2: An example of a 'simultaneous menus' style of exploration interface for census data, after Hochheiser and Shneiderman (2000). For each combination of County, Industry and Year specified on the three 'menus' of links, the dataset contains the three statistics that appear across the bottom.

in four successive years. The statistics specify the number of employees, number of establishments, and total annual payroll. Each of the three input parameters (county, industry, year) was presented as a menu from which the user could make a single selection. An experiment compared two styles of interacting with these menus.

In the first interface style, typical of much Web-based data access, each menu was presented as an individual web page. Making a selection by clicking a menu-item link in the first menu brought up the page containing the second menu, and so on. Making a selection on the third menu brought up a page showing the relevant statistics. This was referred to as the 'sequential menus' interface.

By contrast, in the 'simultaneous menus' interface, the three menus and the result-display page were all visible at all times (using HTML frames). Once the user had made input selections by clicking on one link in each menu, the result display showed the appropriate statistics. Figure 2 shows our own interface reproducing this design.

Retrieving a single set of statistics – a given combination of industry, county and year – is straightforward using either sequential or simultaneous menus. However, Hochheiser and Shneiderman showed that, for simple comparisons between different statistics, using simultaneous menus resulted in lower task-completion times.

We were interested in further examining the comparison issue, in retrieval tasks involving more complex comparisons than in the original experiment. For example, a user may notice that the payroll record for Agricultural Services in Caroline county in 1993 was 'withheld', and may want to check all other payroll statistics for that county (for all industries, all years) to see which others – if any – were also withheld. Or, having noticed that Dorchester county's payroll statistics for Wholesale Trade show a year-on-year decrease throughout the recorded period, a user may want to check whether this is unique to Dorchester, or applies similarly to the Wholesale Trade figures for other counties.

Attempting to answer questions of this complexity with even the simultaneous-menus interface would give rise to the kinds of challenge that we outlined in the Introduction, and that we believe a subjunctive interface can address:

1. A high number of interface actions

   In this interface, making a selection for each parameter requires one click with the mouse. If a user wants to explore combinations of values for different parameters (e.g., to look through Caroline county's statistics for all industries in all years), a large number of mouse operations will be needed.

2. A need to remember earlier results

   Every change in the input values replaces the previous results with those reflecting the current selections. Making comparisons between values, e.g., to understand a trend from year to year, requires users to remember several values.

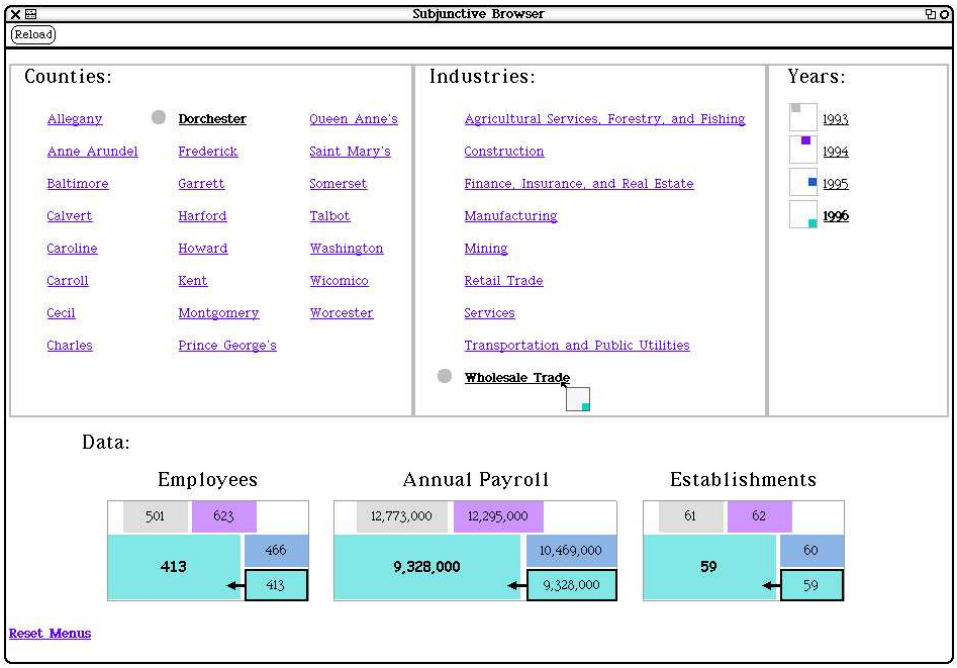

Figure 3: The census-browsing interface shown in Figure 2, operating on four scenarios in parallel. All four scenarios share the parameter settings Dorchester and Wholesale Trade, but each has been assigned a different year. Each of the result displays is therefore split into a four-scenario multiplexed view showing the values for the various years.

3. Mental effort in organising the exploration

   To work through multiple values of one parameter in combination with multiple values of another requires mental effort. Iterating through counties and years to find cases where the payroll statistics have been withheld, as discussed above, requires an orderly search to ensure that all county/year combinations are tried.

3.2. A menu-based subjunctive interface

Figure 3 shows a subjunctive interface for this data set, implemented in a way that remains as close as possible to the original simultaneous-menus design. In this figure the user has established four scenarios. The widget multiplexers handling the various result displays have all taken on a multiple-view form, because each scenario contains a different result value.

3.2.1. Distinguishing between scenarios

Figure 4 shows some details from Figure 3, revealing how the interface helps the user to distinguish the settings and the results that belong to different scenarios.

The user has established one scenario for each of the four values in the Years menu. On the left of Figure 4 is the markup of that menu, and on the right the corresponding state of a textual multiplexer showing the payroll amounts for the four different scenarios (i.e., the four different years). The payroll amount for 1995, for example, is 10,469,000.

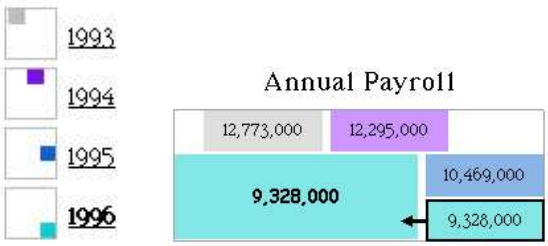

Figure 4: Menu markup and a multiplexed result.

Notice that in Figure 3 all three multiplexers use the same arrangement of views for the different scenarios. As described before, the values found in the equivalent locations within each multiplexer – for example, at bottom right – belong to the same scenario. The marker boxes by the menu items in Figure 4 also use the same layout. We believe that having standard layouts for two, three, or more scenarios will help users to understand quickly which element of a display relates to which scenario. This layout correlation is reinforced by the use of colour – in the backgrounds of textual views, the small squares within the menu markers, and the thin rectangles along the outside edges of graphical multiplexers such as the simulation in Figure 1.

3.2.2. Setting values in selected scenarios

Figure 5: Mouse-pointer indication of active scenarios.

When the mouse pointer is moved over an interactive element within a subjunctive interface, the pointer form reveals the active scenarios – i.e., those that will be affected by user interaction. This is shown in Figure 5. In this case scenarios 2 and 4 are active, so a click at the position shown here will change the Industry value in those scenarios from Construction to Finance – while in scenarios 1 and 3 the value will remain as Agricultural Services.

Figure 6: A pop-up *scenario specifier*.

Figure 6 shows a *scenario specifier* that pops up if the user holds down the mouse button over an interactive element – in this case the link for Transportation. The available scenarios are represented by coloured rectangles, in the now familiar arrangement. Dragging the small circular handle to one of these rectangles, then releasing the mouse, has a dual effect: first it changes the application's active-scenario selection, and then it executes a mouse-click affecting the newly active scenario(s). In the case shown here, releasing the mouse within the grey rectangle at top-left in the specifier would set scenario 1 as the (sole) active scenario, and would set Transportation as the value in that scenario. All with a single click-and-drag action. Releasing instead within either of the two rectangles that contain ticks – meaning that this menu value is already set in those scenarios – would just cause the ticked scenarios to become the active ones.

3.2.3. Establishing new scenarios

The small interlocking-squares icon at the bottom of the scenario specifier signifies a duplication facility. If the user releases the mouse over this icon, the application will duplicate all the currently active scenarios (the number of active scenarios is shown in brackets), and will then set the selected value – here Transportation – in those new scenarios.

3.2.4. Specialised support for comparison

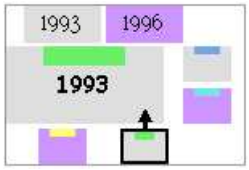

Figure 7: Colour coding to indicate identical values.

Textual multiplexers have an additional feature: a way to reveal when a subset of the scenarios share the same content value[1]. Figure 7 shows this feature in action. Among the six scenarios it is displaying, there are only two different values: the value 1993 is shared by scenarios 1, 3 and 5, while scenarios 2, 4 and 6 all hold 1996. Each value is only displayed within the view of the lowest-numbered scenario in which it appears. The views for the other scenarios sharing that value are blank, but take on the background colour of the view that is displaying the value. So in this case the views for scenarios 3 and 5 are shown with a background of the same colour as scenario 1 (by convention, a light gray). At the tops of these views are small bars filled with their 'own' colours.

We expect that users will find this feature beneficial in situations where they must judge rapidly whether scenarios have the same value for some property – especially in cases where, because of limited space, a textual-multiplexer view cannot display the whole of its contents.

3.3. A slider-based subjunctive interface

Figure 8 shows an alternative interface style for the census application. Instead of having menus for setting the input-parameter values, each parameter is controlled by a slider. A parameter's current setting is shown in a display widget above its slider; this display is under control of a widget multiplexer, allowing it to show different values corresponding to different scenarios.

This interface style has the advantage of taking up a fixed amount of space regardless of the number of alternative values a parameter may take. Therefore this style would be preferable for setting a parameter that has a large number of possible values, such as a continuous scale, or where the domain of input values is not decided in advance, such as when the user must supply an arbitrary string in a text field. A further advantage of using a draggable slider as opposed to clickable menu items is that the user does not need to be looking at the slider to try new values; he or she can move the slider while watching the impact somewhere else on screen, such as a changing result display.

[1] When a textual multiplexer has the same contents for *every* scenario, it shows a single, full-size display – just as the graphical multiplexers do.

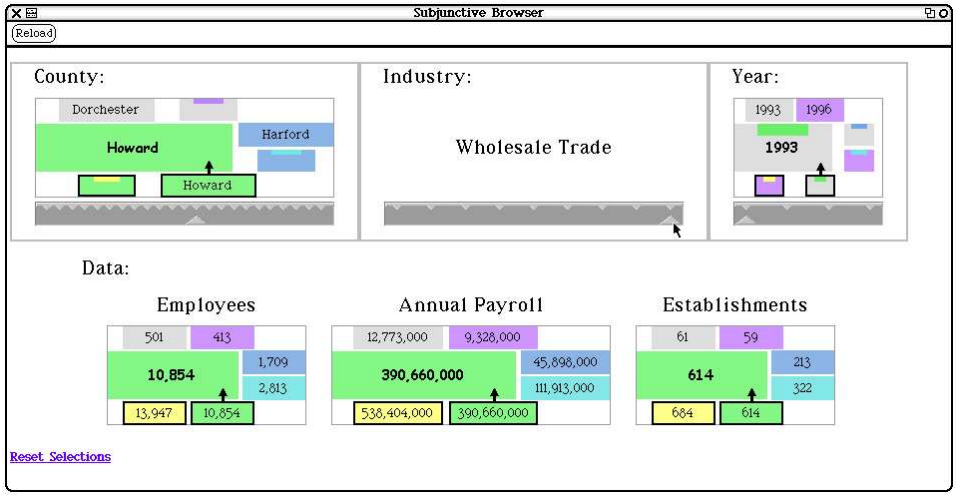

Figure 8: The census application with a slider-based input style. Instead of setting input parameters by clicking on menus that present all the available values, the user operates a slider for each parameter. Above each slider is a field showing the current selected value; when different scenarios have been given different values, that field shows the different values using an appropriate form of multiplexer.

These two ways of presenting a parameter's values – either using a menu or a multiplexed field – are examples of a more general distinction between what we refer to as *scenario marking* vs. *scenario layout*. Scenario layout, in which the values for different scenarios are shown in separate views laid out within a multiplexer, suits any display that normally consists of a single textual or graphical element. Scenario marking, exemplified by the markers seen alongside the menu items in the census application, suits displays whose values are indicated by the position of a mark within a spatial arrangement – such as a marking menu, a list with selections, or a map. Clarifying distinctions like this will help us in our ongoing development of a taxonomy of subjunctive-interface techniques.

4. Discussion

Our most pressing concern is the need for substantial empirical evaluation of the usability of subjunctive interfaces. In the first instance we are performing experiments on the census-application interface. We now discuss the benefits and limitations of subjunctive interfaces that we expect these experiments to illuminate.

4.1. A click-count estimate of benefits

One way to reason about the utility of subjunctive interfaces is to consider *the number of clicks* required for typical tasks, as has been done for other kinds of interface (Hochheiser and Shneiderman, 2000; Sears and Shneiderman, 1994). We may also consider the extent to which people have to use their memory when solving tasks. While such considerations are inherently limited, the aims are (1) to quantify the intuition that subjunctive interfaces are more efficient to use, and (2) to guide the selection of experimental tasks for an empirical evaluation, built on hypotheses about performance.

Imagine a simple task, such as comparing s values of a parameter. In the census data, such a task might be to find the year ($s = 4$) with the highest employment for a certain county and industry. When comparing the menu-based subjunctive interface to simultaneous menus, there is no difference in the number of clicks required to complete such a task; both interfaces require $d + s - 1$ clicks, where d is the number of parameters of the data set. The slider-based subjunctive interface also requires a similar number of clicks as a dynamic query interface. Yet neither subjunctive interface requires that the user remember result details, as is needed with simultaneous menus or a dynamic-query interface; values can be compared directly from the display. This advantage is similar to what has been called the rule of decomposition for multiple view visualisations (Baldonado et al., 2000), which states that using multiple views may help the user to 'divide and conquer'.

Consider a more complex task, such as comparing s values of one parameter across r values of another. In the census data, such a task could be to compare whether construction, mining and services ($s = 3$) showed the same trends in payroll through the years in the data set ($r = 4$). For the menu-based subjunctive interface, the number of required clicks is $d + (s - 1) + (r - 1)$. With simultaneous menus, the number of clicks is $d + s \times r - 1$ (with the given numbers, 14 against the subjunctive interface's 8). With either style of subjunctive interface less information must be remembered, as the results for multiple values of a parameter are shown simultaneously; in the above example, the interface can be set up to show all years' values at the same time.

An even more complex task would be to explore data to find trends over time ($r = 4$) in employment within retail, services and agriculture ($s = 3$) for exemplars of rural and urban counties ($t = 2$). For this task subjunctive interfaces also require fewer clicks ($d + (s - 1) + (r - 1) + (t - 1)$ = 9) than do for example the simultaneous menus ($d + s \times r \times t - 1$ = 26).

4.2. Barriers to realising benefits

The above reasoning suggests that we can expect better performance for subjunctive interfaces as task complexity increases. However, a simple click-count comparison overlooks the facts that (a) the user has to set up the subjunctive interfaces in an optimal way, (b) some clicks may be more complex than others (e.g., double click, or click and drag), and (c) locating and comparing data within a subjunctive-interface display involves its own form of mental effort. These factors may all reduce the benefit eventually achieved in practical situations of use.

In particular, the formula for subjunctive-interface click count is only valid if all but one dimension of variation can be set up for simultaneous display, leaving that last dimension to be manipulated dynamically. This depends on how many scenarios can be handled simultaneously. Given an eight-scenario limit, in the latter task the user could either set up six scenarios holding the combinations of industry type and county ($s \times t$), or eight scenarios for year and county ($r \times t$). But if t also had a value of 3, the optimal click count suggested by the formula would not be achievable in practice.

Evaluating user response to these challenges will help us to judge the interfaces' performance in terms of what Baldonado et al. (2000) term the principle of parsimony of related views. In other words, whether the burden of understanding and operating the views outweighs the advantages that they offer. Part of the issue will be the utility and comprehensibility of many detailed features of the interface, such as the layout and colour-coding of multi-scenario displays. In the first instance we are canvassing informal evaluations of these features from our experimental subjects.

At a more subtle level, the self-evidence rule of Baldonado et al. directs us to check whether users are confused by differences between the aspect ratios of different multiple-scenario layouts – for example, between the multiplexer and menu-marking boxes seen in Figure 4.

4.3. Experimental evaluation

In this paper we have introduced two different interfaces for the census application – menu-based and slider-based – and discussed how these exemplify scenario-marking and scenario-layout interaction styles. The two styles' relative benefits and disadvantages may be revealed with the help of controlled experiments. Our first priority, however, has been to evaluate the usability and performance of a subjunctive interface relative to an interface with no multiple-scenario capability.

We have begun by comparing the scenario-marking subjunctive interface against the simultaneous-menus interface, for complex multi-record retrievals from the census data set. The interface used for our initial 20-subject experiment differed somewhat from the version shown here, in ways that suit this text-only application and that were suggested by the results of a pilot study. The main difference was that the widget multiplexers showed displays of equal size, without a separate working view, and supported up to 12 parallel scenarios. The special colour-coding mechanism described in section 3.2.4 was also disabled.

While the results of that experiment were promising, they also highlighted some usability issues that we are now investigating with the help of further design iterations. In this way we aim to pursue an informed exploration of the subjunctive-interface design space, firstly for this specific application and then for other domains and other styles of interaction.

5. Conclusion

We have identified three kinds of burden that commonly affect users who want to compare results during interactive exploration of information: a high number of interactions, mental effort in remembering what has been seen, and mental effort in organising the exploration. We suggest that these can all be reduced by use of a subjunctive interface approach, which lets users establish, view and adjust many scenarios in parallel. Two styles of subjunctive interface for accessing a census dataset have been built, one of which is now being evaluated alongside a more traditional menu interface.

6. Acknowledgements

We thank Harry Hochheiser for lending us the census data, and Jørgen Gomme and Erik Frøkjær for comments on an early draft of this paper. The tools shown here are built as an extension of the Morphic interface library within Squeak Smalltalk; the basic ant simulation appears in Squeak's standard class library.

References

Ahlberg, C. & Shneiderman, B. (1994). Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of ACM CHI 94* (pp. 313–317). ACM Press.

Baldonado, M. Q. W., Woodruff, A., & Kuchinsky, A. (2000). Guidelines for using multiple views in information visualization. In *Proceedings of AVI 2000* (pp. 110–119).

Bates, M. J. (1989). The design of browsing and berrypicking techniques for the online search interface. *Online Review, 13*(5), 407–424.

Bier, E. A., Stone, M. C., Pier, K., Buxton, W., & DeRose, T. D. (1993). Toolglass and magic lenses: The see-through interface. In *Proceedings of ACM SIGGRAPH 93* (pp. 73–80). ACM Press.

Fishkin, K. & Stone, M. (1995). Enhanced dynamic queries via movable filters. In *Proceedings of ACM CHI 95* (pp. 415–420). ACM Press.

Goldstein, J. & Roth, S. (1994). Using aggregation and dynamic queries for exploring large data sets. In *Proceedings of ACM CHI 94* (pp. 23–29). ACM Press.

Hochheiser, H. & Shneiderman, B. (2000). Performance benefits of simultaneous over sequential menus as task complexity increases. *International Journal of Human-Computer Interaction, 12*(2), 173–192.

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: an Eternal Golden Braid*. Basic Books.

Lunzer, A. (1999). Choice and comparison where the user wants them: Subjunctive interfaces for computer-supported exploration. In *Proceedings of IFIP TC. 13 International Conference on Human-Computer Interaction (INTERACT '99)* (pp. 474–482). IOS Press.

Lunzer, A. (2002). Widget multiplexers for in-situ handling of alternative application states. In *Proceedings of 2nd Danish Human-Computer Interaction Research Symposium* (pp. 40–41). University of Copenhagen DIKU Technical Report 2002/19.

Sears, A. & Shneiderman, B. (1994). Split menus: Effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction, 1*(1), 27–51.

Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Software, 11*(6), 70–77.

Terry, M. & Mynatt, E. D. (2002). Side Views: Persistent, on-demand previews for open-ended tasks. In *Proceedings of ACM UIST '02* (pp. 71–80). ACM Press.

Waterworth, J. & Chignell, M. (1991). A model for information exploration. *Hypermedia, 3*(1), 35–58.