Benefits of Subjunctive Interface Support for Exploratory Access to Online Resources

Aran Lunzer

Meme Media Laboratory[⋆], Hokkaido University, Sapporo 060-8628, Japan
aran@meme.hokudai.ac.jp

Abstract. When exploring online resources, users often make many separate retrievals – specifying in turn various parameter values to make up a request, and examining the corresponding results. Any interface that supports only one retrieval at a time, i.e., one set of parameter values and the corresponding results, places a heavy burden on a user who wants to compare available results, or simply to try a range of retrievals. A subjunctive-interface approach may reduce this burden. Subjunctive interfaces support the setting up, viewing and adjustment of multiple scenarios in parallel, leading to more efficient iteration through related scenarios, and the opportunity for side-by-side instead of temporally separated viewing. We examine how these facilities can be applied in a range of information-access applications, and the benefits that can be obtained.

1 Introduction

People explore online resources in a wide variety of domains – notionally as wide as the variety of databases, or of Web applications. However, many resource-access applications only deliver results in response to explicit, pinpoint requests. For example, interfaces for flight enquiries typically require the user to specify exactly one destination city; while this is convenient for users with precisely formulated travel plans, someone who wants to compare the deals and schedules available for a range of destinations must embark on an exploration: submitting various requests and analysing their respective results. Depending on the applications available, such explorations can present the following kinds of challenge:

Poor support for covering a range of requests. When a user wants to submit a range of requests, the details of those requests often follow some pattern. For example, when searching for flights, a user might request information for several routes on a given date, or for a single route over many dates, or combinations of a few routes and dates.
If the interface only supports the handling of a single request at a time, this burdens the user not only with a potentially high number of interface actions to specify and submit the requests, but also mental effort in planning the requests, remembering which requests have been made so far, and remembering where interesting results were found.

[⋆] Since May 2004. The work reported here was partly carried out during earlier employment at the same laboratory (1997–2001), and partly at the Natural Sciences ICT Competence Centre, University of Copenhagen, 2200 Copenhagen Ø, Denmark.

Poor support for comparing results. Few things in life can be evaluated in isolation. In many kinds of information exploration, comparing the results of alternative requests is what gives meaning to those results – for example, in judging whether a given flight is good or bad value for money.

However, many application interfaces only display the results from the latest request; each request replaces the previous one. Therefore comparing results requires the user to remember – or to have written down, or to request again – the details of those that are currently out of sight. This again can constitute both a physical and a mental burden.

Poor support for ambiguous retrievals. A well known challenge in the field of Information Retrieval is sense disambiguation. For example, because the word 'bank' might refer to a financial institution or the edge of a river, results retrieved for one sense may be completely unhelpful for a user interested in the other. Similarly, in the burgeoning field of context-sensitive information delivery, the provided information can be tailored in accordance with details of the user's current context. But context is subject to interpretation, so again the results may not be what the user expects.

A common tendency is for application designers to invoke some form of heuristics in deciding on a single way to interpret the situation. In some cases, inevitably, users will feel that they have been offered irrelevant information. The situation is even worse if there is no way to remove the heuristic bias, in order to access the information that was intended.

This paper reviews how applications equipped with 'subjunctive interface' facilities [1–3] may help to relieve these problems. The idea of a subjunctive interface is to let the user of a computer application set up multiple scenarios that can then be viewed and adjusted in parallel; the concept was inspired by Hofstadter's [4] 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.

Before going any further, we should clarify our use of the term 'scenario'. Taking an abstract view of an application, as a program that computes some outcome on the basis of user-specified inputs, a scenario comprises a combination of inputs and the corresponding outcome. In some applications time is also significant – in the timing of input specifications, evolution of the outcome, or both. Typically scenarios that have different inputs will also have different outcomes; in addition, if a computation depends on factors other than the user's inputs, such as some randomisation within a simulation, scenarios may have different outcomes even if their inputs are identical.

The key features of an application with a subjunctive interface are as follows:

Multiple scenarios can co-exist. At any given time, the application can support multiple independent scenarios. Typically these are created by a user (though exceptions will be shown later). For example, when choosing a value for some application input, the user may be equally interested in several alternative values, such as alternative destination cities; creating a separate scenario to cater for each value allows all those interests to be captured.

The user can view scenarios side by side. The application can display all its currently existing scenarios side by side, in a way that lets the user compare them, or read the values (inputs and outcomes) for each scenario individually.

The user can adjust scenarios in parallel. The user can control scenarios in parallel, for example by adjusting an input parameter that is shared by many scenarios and seeing simultaneously how this adjustment affects each scenario.

How might such facilities help in addressing the three information-exploration challenges outlined above?

- For a start we can expect that covering a range of retrievals will become less laborious, at least if there is some repeating pattern in the retrievals. For example, consider a user who is interested in several possible dates of travel, and wants to enquire about several possible destinations for each of those dates. This can be achieved by setting up a scenario for each city, then successively working through the various dates in all the scenarios in parallel (or the converse: setting up a scenario for each date, then changing the city in all those scenarios). In either case, the number of interface actions needed will be reduced, as will the burden of remembering which retrievals have been run so far.
- Comparison is supported, because of the ability to see multiple scenarios (i.e., multiple retrievals) side by side rather than having to remember their contents or revisit them repeatedly.
- In a situation where the specified retrieval is ambiguous, the ability to establish multiple scenarios can be used to deliver alternative results reflecting the various available interpretations. Seeing those scenarios, the user can evaluate which of them reflect(s) the most relevant interpretation for the current occasion – especially if each scenario is annotated with the reasoning used by the system in offering it.

In the following section we introduce five applications that demonstrate multiple-scenario support applied to information exploration. These applications have been implemented primarily to test out concepts and interface approaches, rather than to stand as generally useful tools; that said, the census browser described in Sect. 2.1 has undergone several design iterations in response to user trials, and even the prototype Web-application clipping tool described in Sect. 2.2 turns out to be capable of addressing a wide range of situations.

After the application descriptions, in Sect. 3 we outline related work, while Sect. 4 looks to a generalisation of the findings reported here.

2 Sample Applications

In this section we introduce five applications that illustrate how the use of multiple scenarios, as supported by subjunctive interfaces, can provide benefits to users pursuing explorations among online resources. The applications are as follows:

1. A browser for census records, in which the use of multiple scenarios helps the user to view and compare trends among the records.
2. C3W: a framework that lets users extract and reuse elements of Web applications, for which multiple scenarios allow for efficient execution and viewing of multiple Web retrievals.
3. TopicaBrush: an interface for highlighting subsets of data in a multi-faceted repository, where linked scenarios help the user to compare data subsets based on subtly differing criteria.
4. HIBench: a simple framework for setting up queries against a database of multi-attribute data items; scenarios are used to represent parallel searches requested by the user, or implied by ambiguity in such requests.
5. ScheduleBlender: a prototype of a calendar that would support experimentation with alternative placements of events – or, as shown here, could use tailoring information to calculate and deliver event-related data.

The applications can be broadly categorised according to whether the creation of scenarios is entirely on the user's initiative (1–3), or can also be triggered by the system (4, 5). The demonstrations of the latter, mixed-initiative systems address situations in which a user's request turns out to be ambiguous.

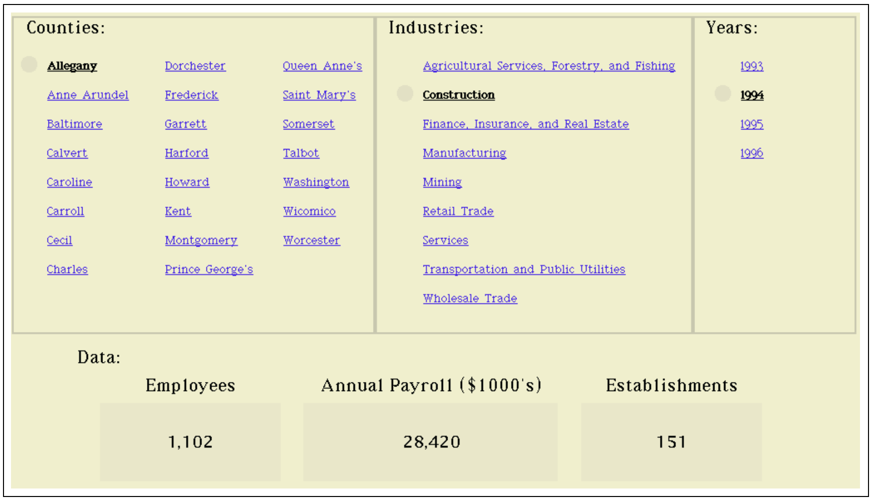
2.1 Browsing Census Records

First we describe a simple browser for census data, that we extended with a subjunctive interface. We have evaluated the relative performance of the browser with and without this extension.

Figure 1 shows a browser based on the 'simultaneous menus' interface created by Hochheiser and Shneiderman [5], for browsing census data on commercial activities in the state of Maryland. The data set contains 828 records, holding statistics for nine industry areas in each of twenty-three counties over four successive years. Each record specifies the number of employees, the number of establishments, and the total annual payroll. The user specifies a record by selecting a county, industry and year in the three menus in the upper part of the browser; the statistics appear in the result displays below those menus.

However, this browser does not provide good support for iterative retrievals, or for comparisons. For example, a user who wants to compare the statistics for a given county and industry over several years must click each year in turn, read off the statistics for that year, and remember them (or write them down) for comparison with the other years. To make the equivalent comparison for the same county but a different industry, the user must change the industry selection and again work through the years.

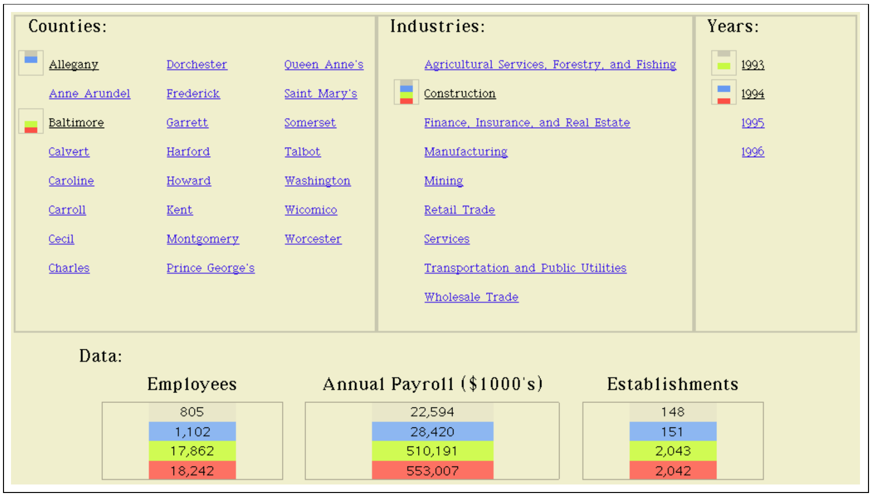
Establishing Multiple Scenarios. In this application a scenario comprises a selection in each menu, and the corresponding statistics. Figure 2 shows a subjunctive-interface version of the application, in which the user has established four scenarios – for two years' records in each of two counties. Therefore the three result displays are each showing four values, and instead of the one-marker-per-menu style of the simple interface, the menus' markers can now show



Data:

Employees	Annual Payroll ($1000's)	Establishments
1,102	28,420	151

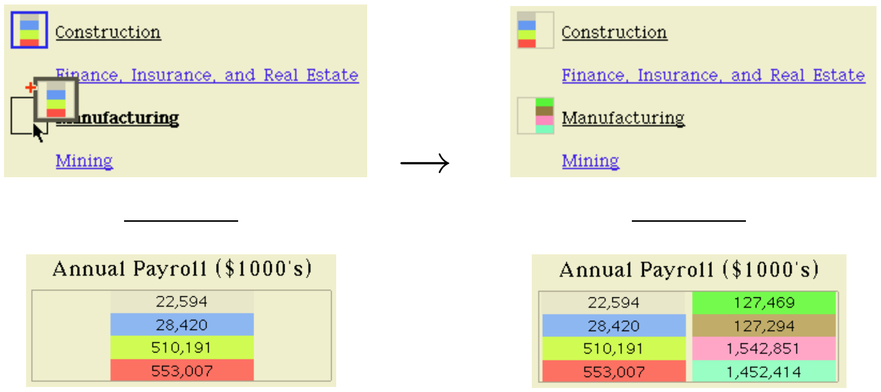
Fig. 1. A 'simultaneous menus' style of exploration interface for census data, after [5]. For each combination of County, Industry and Year specified on the three 'menus' of links, the dataset contains the three statistics that are retrieved and displayed across the lower part of the browser.



Data:

Employees	Annual Payroll ($1000's)	Establishments
805	22,594	148
1,102	28,420	151
17,862	510,191	2,043
18,242	553,007	2,042

Fig. 2. A subjunctive-interface version of the browser in Fig. 1. The browser is showing four scenarios corresponding to the Construction statistics for both Allegany and Baltimore, in 1993 and 1994. Correspondence between menu selections and result values is indicated with position and colour cues in the result displays and in the markers next to menu items; for example, the values 805, 22594 and 148 at the top of the result displays are for Allegany in 1993.


Construction
Finance, Insurance, and Real Estate
Manufacturing
Mining
Construction
Finance, Insurance, and Real Estate
Manufacturing
Mining

Annual Payroll ($1000's)

22,594
28,420
510,191
553,007

Annual Payroll ($1000's)

22,594	127,469
28,420	127,294
510,191	1,542,851
553,007	1,452,414

Fig. 3. A user cloning four scenarios that relate to Construction, giving the new scenarios the value Manufacturing. At top left, the user has picked up the menu marker for Construction while holding the Ctrl key, and is about to drop it onto Manufacturing; at top right is the menu state after this operation. Below are corresponding snapshots of one of the result displays, showing the increase in the number of scenarios.

when several scenarios have the same setting. As shown in Fig. 3, a user creates new scenarios by cloning existing ones, through interaction with these markers. Picking up a marker while holding the Ctrl key, and dropping it on some other item within the same menu, will clone the scenarios shown in the dragged marker. The new scenarios are identical to the originals, except for having the setting of the menu item where the marker was dropped.

Side-by-Side Viewing. The interface in Fig. 2 is based on a refinement of our ‘widget multiplexer’ approach [2], that allows an application to show multiple scenarios without wastefully replicating the parts of its interface that are identical in all cases. Each multiplexer handles the presentation of and interaction with a single input or output widget, and provides side-by-side display of the widget’s various states in the existing scenarios. In this application, multiplexers are used for the menus and the result displays. All multiplexers within an application use correlated geometrical layout and colouring for the scenarios’ values, to help the user locate the values that constitute each scenario; here the menu markers are designed to be correlated easily with the values in the statistics displays.

Parallel Adjustment. The only kind of ‘adjustment’ supported in the original application is changing which census record is retrieved, by changing the selection within some menu. For example, having requested the statistics for Allegany Construction in 1994, as shown in Fig. 1, the user might click on Dorchester to retrieve that county’s corresponding statistics instead.

In the subjunctive interface, clicking within a menu can affect more than one scenario at a time, according to the following two rules: (1) When none of a menu’s items is selected, a click on any item sets that value in all scenarios;

(2) Further clicks in a menu affect just the scenarios shown in the currently highlighted marker within that menu. This feature is shown in Fig. 4.

In some situations, a user might explicitly want to adjust only a single scenario, avoiding the parallel adjustment that would happen with a mouse click. Figure 4 also shows the interface mechanism that enables this.

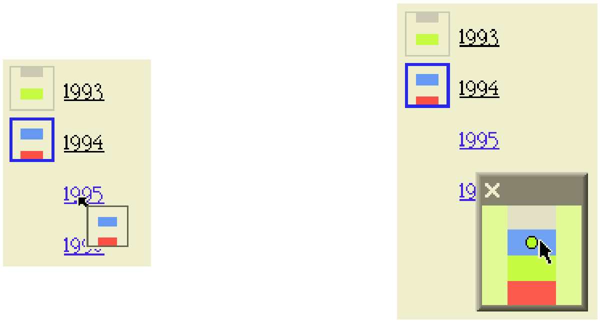


Fig. 4. Left: By default, a mouse click in a menu updates the scenarios shown in the menus's currently highlighted marker. Here the marker for 1994 is highlighted (shown by its dark blue outline, and replicated in the mouse pointer), so a click on 1995 would switch the two 1994 scenarios to that value. The user can highlight a different marker by clicking on it. **Right:** Clicking and holding the mouse button on a menu item creates a pop-up that lets the user choose a single scenario to receive the clicked value. Here the user has clicked on 1996; releasing the mouse in its present position would set the year to 1996 in just the second scenario.

Benefits. We briefly describe two experiments that we ran to compare the usability and performance of the original and subjunctive-interface census browsers.

The experimental tasks used by Hochheiser and Shneiderman to evaluate the simultaneous-menus browser [5] were all simple two-case comparisons. For this experiment we defined more complex retrieval and comparison tasks, of the following three types:

Intra-set comparison. These tasks require pairwise comparisons between many combinations of the records in some set. For example, one task asks: 'Considering Wholesale Trade in [five named counties] in 1993, find how the counties are ordered in terms of number of Employees. In order from fewest Employees to most, what are the Payroll values for these counties?'

Iterative examination. These tasks call for examination of records that lie in a repeating pattern. For example, 'In 1996, for which of the industries do [three named counties] all have 1000 or more Employees?'

Iterative comparison. These tasks are similar to iterative examination, but call for comparison between the records rather than merely checking whether each record satisfies some criterion. For example, 'In which counties does the Payroll for Wholesale Trade fall in every year from 1993 to 1996?'

We expected that, for each task type, appropriate use of the subjunctive interface would provide some benefit over using the simple interface. For intra-set comparisons the benefit is merely in being able to keep values on view rather than having to remember them or write them down. For iterative examinations and comparisons, the iteration can be performed more efficiently if the user first sets up scenarios that express the repeating pattern required by the task.

In the first experiment, twenty subjects were each given sets of tasks to complete with each interface. The subjects significantly preferred the subjunctive interface, and rated it as being more satisfying to use. When using the subjunctive interface they depended less on writing down or remembering data, as suggested by fewer interim marks made on paper and by reports of lower mental workload. They also used fewer interface actions to complete the tasks. However, we found no corresponding reduction in task completion time, mainly because some subjects encountered problems in using the facilities for setting up and controlling scenarios.

By examining the subjects' interface logs, we found a number of commonly made mistakes and difficulties that seemed to stem from the interface design. In addition, many subjects commented that they would probably become more effective at using the subjunctive interface if they had more practice. We therefore ran a second experiment, using an interface that was modified to address the most common difficulties, and incorporating a considerably longer period of use. Our main hypothesis was that, after extended practice with both the simple and subjunctive interfaces, the subjunctive interface would become significantly faster than the simple interface.

In the second experiment, seven subjects performed tasks using the simple interface and the modified subjunctive interface, over five sessions each lasting approximately one hour. In the final session, the tasks were completed 27% more quickly when tackled with the subjunctive rather than with the simple interface – and for some of the iterative comparison tasks, the subjunctive interface was more than twice as fast. Our hypothesis was thus confirmed.

2.2 C3W: Capturing and Reusing Web-Application Behaviour

In this section we describe C3W (Clip, Connect and Clone for the Web) [6], a prototype that enhances access to resources provided by Web applications. C3W is being developed by Hokkaido University Ph.D. student Jun Fujima.

Part of the motivation for developing C3W was that Web applications often exhibit the problems outlined in the Introduction – i.e., handling only precisely specified requests, thus making it hard for users to cover a range of requests or to compare results. A further challenge is that the Web pages serving as the interfaces to these applications are often cluttered with elements of no interest to the user; finding the relevant elements among that clutter incurs a cognitive load, especially in repeated use. Finally, we noticed a chance to automate the setting of inputs for one Web application based on results from others.

True to its name, C3W is based on the following three facilities:

Clipping. By drag-and-drop manipulation, a user can select and extract input and result elements from the pages of a Web application. Placing the elements on a substrate called a C3Sheet turns them into cells that work as viewports onto the original Web pages. Clipped input elements continue to support user input, and clipped result elements display the Web application's corresponding results. Thus the user can create compact, customised applications based on processing facilities offered over the Web.

Connecting. A single C3Sheet can hold input and result cells taken from multiple Web applications. For any input cell that takes a text-string value, the user can define a formula to derive its value from the values of other cells – typically, cells that hold results from other applications. This allows independent applications to be connected in user-specified ways.

Cloning. The user can set up multiple scenarios – i.e., different settings for the Web-application inputs, leading to different results – to be shown at the same time. Each cell displays, side by side, its contents in the various scenarios. This provides an efficient way for a user to explore and compare the different results available through a given Web application, or the user's own custom-built application combinations.

Establishing and Using Multiple Scenarios. Cloning, the key to C3W's subjunctive-interface support, is supported as follows:

Creating scenarios. The user creates new scenarios by cloning a cell that represents an input value. All cells downstream of the cloned input – either in terms of captured Web-application behaviour, or through formulas added by the user – are cloned too, so that they can display distinct results for each scenario. Cells that do not depend on the cloned cell's value are not cloned, and maintain their current value in every scenario.

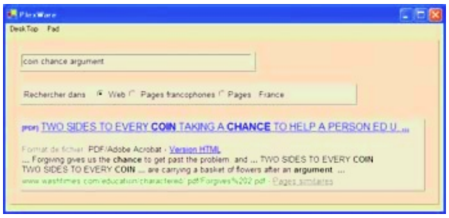
Cloning additional inputs. When multiple scenarios have been created by cloning, the user can ask to clone another input cell that is not currently cloned. That cell is automatically given a clone for each existing scenario, as are any cells downstream of it. The number of scenarios does not change[1].

Adjusting scenarios. Entering a new value in one clone of a cloned cell will only affect the scenario corresponding to that clone. Entering a new value in an uncloned cell will affect all scenarios simultaneously.

Deleting scenarios. The user can delete a scenario. This removes all cell clones associated with that scenario.

Benefits. Figure 5 shows a simple instance of C3W in use. By cloning cells, the user has set up an interface that allows three different Google searches to be run automatically whenever new search keywords are supplied. Even this

[1] An alternative approach would be to multiply the number of existing scenarios. However, since Web-based retrieval is still a relatively resource-hungry activity, we are reluctant to lead users down this path.

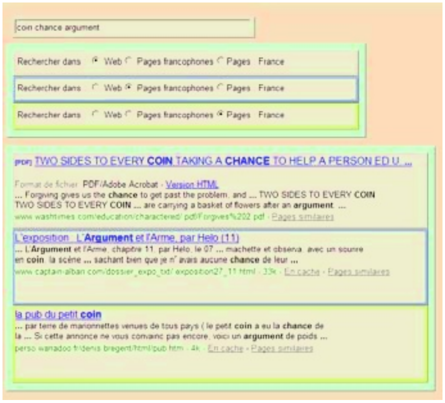


coin chance argument

Rechercher dans [✓ Web] Pages francophones Pages France

[PDF] TWO SIDES TO EVERY **COIN** TAKING A **CHANCE** TO HELP A PERSON ED U ...

Format de fichier: PDF/Adobe Acrobat - Version HTML
... Forgiving gives us the **chance** to get past the problem. and ... TWO SIDES TO EVERY **COIN**
TWO SIDES TO EVERY **COIN** ... are carrying a basket of flowers after an **argument**. ...
www.washtimes.com/education/charactered/ pdf/Forgives%202.pdf - Pages similaires



coin chance argument

Rechercher dans [✓ Web] Pages francophones Pages France

Rechercher dans Web [✓ Pages francophones] Pages France

Rechercher dans Web Pages francophones [✓ Pages France]

[PDF] TWO SIDES TO EVERY **COIN** TAKING A **CHANCE** TO HELP A PERSON ED U ...

Format de fichier: PDF/Adobe Acrobat - Version HTML
... Forgiving gives us the **chance** to get past the problem. and ... TWO SIDES TO EVERY **COIN**
TWO SIDES TO EVERY **COIN** ... are carrying a basket of flowers after an **argument**. ...
www.washtimes.com/education/charactered/ pdf/Forgives%202.pdf - Pages similaires

L'exposition : L'**Argument** et l'Arme, par Helo (11)
... L'**Argument** et l'Arme, chapitre 11, par Helo, le 07 ... machette et observa, avec un sourire
en **coin**, la scène ... sachant bien que je n' avais aucune **chance** de leur ...
www.captain-alban.com/dossier_expo_txt/ exposition27_11.html - 33k - En cache - Pages similaires

la pub du petit **coin**
... par terre de marionnettes venues de tous pays (le petit **coin** a eu la **chance** de
la ... Si cette annonce ne vous convainc pas encore, voici un **argument** de poids ...
perso.wanadoo.fr/denis.bregent/html/pub.htm - 4k - En cache - Pages similaires

Fig. 5. Use of multiple scenarios in C3W. **Left:** A user has set up three cells containing elements extracted from a search at www.google.fr: the keyword input field; the switch that selects whether the search should cover the whole Web, just pages in French, or just pages in France; and the top result (showing page name and keyword context) from a sample search. **Right:** The user has cloned the switch cell to create three instances, and has given each instance a different setting. Because the state of this switch potentially affects the value for the top result, that cell has automatically been cloned too. Entering new search keywords into the keyword cell will therefore execute three Google queries with different scopes, and display the three respective top results.

simple example shows a clear benefit to the user: no matter how easy it is, in a normal browser, to re-run a given Google search with each position of the search-scope switch, the actions required to do so – adjusting the switch, re-running the search, figuring out whether the results have changed – stand as a barrier. A setup such as the one shown in the figure can, in effect, remove this barrier.

When a user has connected multiple applications using formulas, the labour-saving afforded by processing multiple scenarios in parallel becomes even more significant.

2.3 TopicaBrush: Dynamic Exploration of Relational-Data Subsets

Under the Topica [7] approach, information providers gather diverse resources into a relational table, then combine that table with a Web-page-like presentation object. The resulting assembly is called a Topica Document. Within a Topica Document's presentation, each column of its underlying relational table appears as a 'topos' (plural 'topoi') – an interactive region of the presentation, through which a user gains access to the values in that column. If a user examines the contents of one topos and marks an interest in some subset of the resources shown there, then resources at other topoi, correlated through the underlying table, will automatically be marked too.

TopicaBrush was built to explore Topica-Document interaction; in essence, how to support a user in marking and examining portions of relational tables. TopicaBrush combines two well known data-exploration techniques: *brushing* (a feature of, for example, Visage [8]), in which an object coloured by the user in one view carries that same colour wherever else it appears; and *aggregation* (in the sense of [9]), by which a user creates new subsets of a multi-attribute data collection in terms of constraints on the attributes' values. By applying the same coloured mark to selected values in one or more columns, the user specifies the bounds of a subset of the underlying data table; this subset then reveals its colour on all the data values it contains.

We now describe how the features of TopicaBrush for letting a user pursue a range of requests, and compare their results, are based on a multi-scenario approach.

Establishing and Using Multiple Scenarios. Figure 6 shows TopicaBrush being used to investigate a dataset describing travel packages. The four lists across the bottom of the figure are topoi whose contents the user has chosen to view; these lists support mouse-click selection to indicate values that are of inter-

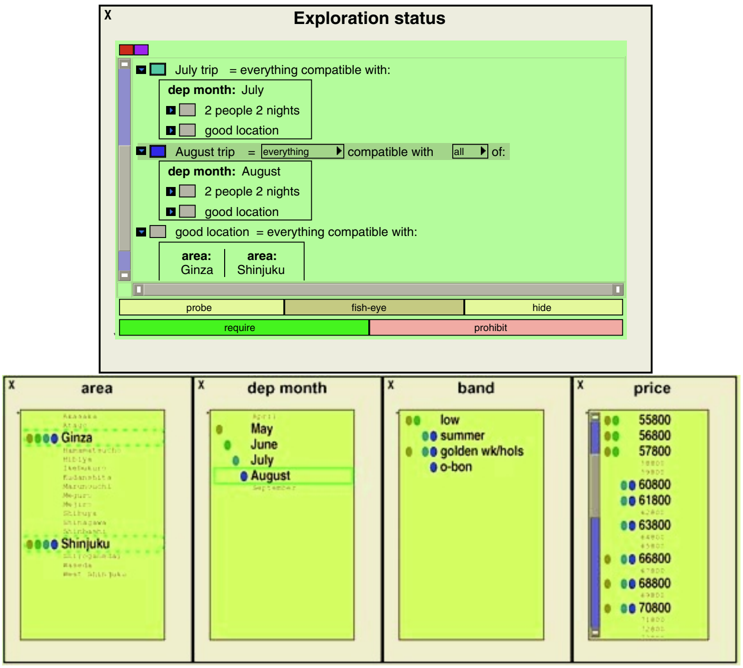


Fig. 6. Highlighting data subsets in TopicaBrush, on a database of travel packages. The user has set up and marked four main sets, addressing the months May to August respectively. Their markings are seen in the topos lists; in the price topos, for example, it is seen that both May and June have deals at the same low prices, while some other prices in May are high. The criteria defining each of the month sets include both the '2 person 2 night' set (definition not shown here) and the 'good location' set, currently defined as a disjunction of Ginza and Shinjuku. If the definition of 'good location' were changed, all four derived sets and their marking would be updated simultaneously.

est. The 'exploration status' view supports the further operations needed to manipulate and combine the subsets identified in the topoi, and to assign coloured marks to whichever derived sets are of interest. It is these sets, representing potentially complex relational queries, that are the scenarios in TopicaBrush.

The sets are displayed by drawing the user-assigned marks against the appropriate values within the topos lists. Rather than use a widget-multiplexer approach, displaying numerous copies of the lists, TopicaBrush takes advantage of the fact that the lists themselves have static content. Each element within a list is tagged with markers representing all the sets of which that element is a member; as shown in the figure, elements with no marking can be shown at reduced size to save screen space while maintaining context for the other elements.

Allowing the sets to be displayed independently gives more information to the user than if the system could only support a single set at a time. In the case shown in Fig. 6, for example, the user could have specified a single set where the month was 'May OR June OR July OR August' – but this would contain a mass of undifferentiated prices, with no clues as to the seasonal variation.

TopicaBrush supports users in adjusting multiple scenarios in parallel, in that sets can be defined in terms of other sets. If many scenarios include some common set S within their definition, any update to S will simultaneously affect all those scenarios.

Benefits. The concepts explored in TopicaBrush seem promising for the general challenge of retrieval from faceted repositories, where even recent interfaces (such as [10]) embody the dubious assumption that a user is always willing to define his or her interests as a single – albeit complex and evolving – query.

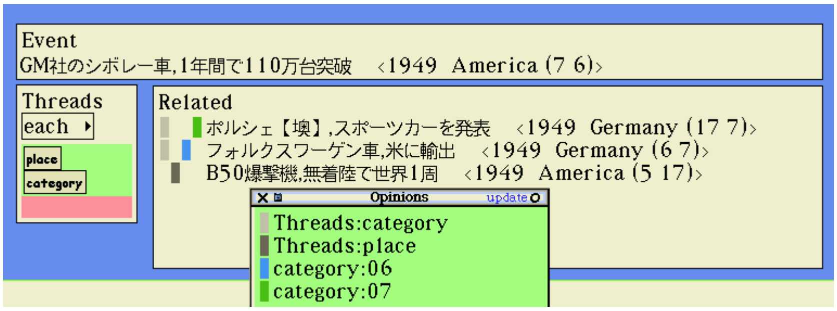
2.4 HIBench: Browsing Relationships in Multi-attribute Data

Whereas the previous examples all involve explicit request of alternative scenarios by the user, this and the next are examples of mixed initiative; the application itself can generate supplementary scenarios to assist the user. The typical pattern of use is that a user makes a request for some information, and the system provides a number of equally appropriate but distinct results, along with information that clarifies how each result has arisen.

Figure 7 shows a display detail from HIBench, a specialised retrieval interface inspired by, and built around, an online historical-event collection that mirrors the contents of [11][2]. The notional goal in building HIBench was to help people understand the context of any event in the repository, by using queries to discover similar events in its chronological vicinity.

Like C3W (Sect. 2.2), HIBench is a spreadsheet-like framework in which users can create cells whose contents are derived from other cells. However, in addition

[2] In total there are approximately fifty thousand items; we used just the latter half, covering events in the 20th century. All information in the repository is in Japanese; the English place-name translations seen here were added to assist non-Japanese explanation.


Event
GM社のシボレー車,1年間で110万台突破 ‹1949 America (7 6)›

Threads
each ▸
place
category

Related
ポルシェ【墺】,スポーツカーを発表 ‹1949 Germany (17 7)›
フォルクスワーゲン車,米に輸出 ‹1949 Germany (6 7)›
B50爆撃機,無着陸で世界1周 ‹1949 America (5 17)›

Threads:category
Threads:place
category:06
category:07

Fig. 7. Matching in HIBench, with ambiguity. The cell called Related is deriving its contents by applying the matchers in the cell Threads to the reference event currently held in Event. Because Threads specifies both ‘place’ and ‘category’ matching, and because the current reference event belongs to two categories (7 and 6), matching events are found for each category as well as for the place (America). The three results are marked with coloured tags whose meanings are listed in the ‘Opinions’ view.
Key/translation: Category 7 covers corporate business, while 6 is international trade. The reference event here comments on Chevrolet production. The top related item (another corporate business item) is an announcement by the Porsche company; the next (matching international trade) tells of the export of Volkswagens; the last (matching America) is a round-the-world flight by an American B50 bomber.

to standard arithmetic and string functions as normally found in spreadsheets, HIBench offers specialised functions for searching against the event repository. One such function takes a reference event and a match condition, specified by the user as a combination of property-specific matchers, and retrieves the next event (in the repository’s chronological sequence) that matches the reference event in the specified way. The properties available for matching against an event are its news-headline-like description, a date, one or more locations (typically countries) affected by the event, and one or more categories (denoting fields such as politics, education, various branches of science and technology, literature and so on) assigned by the repository’s editors.

Establishing and Using Multiple Scenarios. HIBench supports both user-requested and system-generated creation of scenarios. Figure 7 shows a case involving both. The user has set up matching based on the contents of the cell labelled Threads, into which two matchers (‘place’ and ‘category’) have been dropped. At the top of the Threads cell is a menu, in which the value *each* is currently selected. This menu, which also contains the values *any* and *all*, specifies how multiple matchers within the cell are to be combined. The *any* setting is like a relational OR; the compound condition will be satisfied by the first event to satisfy any matcher. *All* is like AND, only matching an event that satisfies all matchers. The setting *each* forces a separate search to be performed for each matcher, in separate scenarios, and the distinct matches (which may coincidentally be the same event) to be collected and displayed together.

In addition, the system has detected that in this case a ‘category’ match is ambiguous, because the reference event happens to be registered under two categories. Correspondingly, separate scenarios have been created to perform matching for each category. The total number of scenarios is thus three: one for place, and one for each of the two categories. These three scenarios have each delivered a different matching event for display in the Related cell.

HIBench, like TopicaBrush (Sect. 2.3), does not use widget multiplexers to create multi-scenario displays. Cells containing multiple values are presented as lists, with each list element marked to show which scenario(s) it is related to.

HIBench supports parallel adjustment of scenarios in the sense that, when the user has set up compound match conditions using the *each* setting, a replacement of the reference event will automatically cause multiple retrievals to be executed in parallel. This can be applied at nested levels. For example, although not shown in Fig. 7, the ‘place’ matcher is itself defined as a compound condition involving matchers for city, country and continent; it is up to the user to decide whether those matchers are combined using *any*, *all*, or *each*.

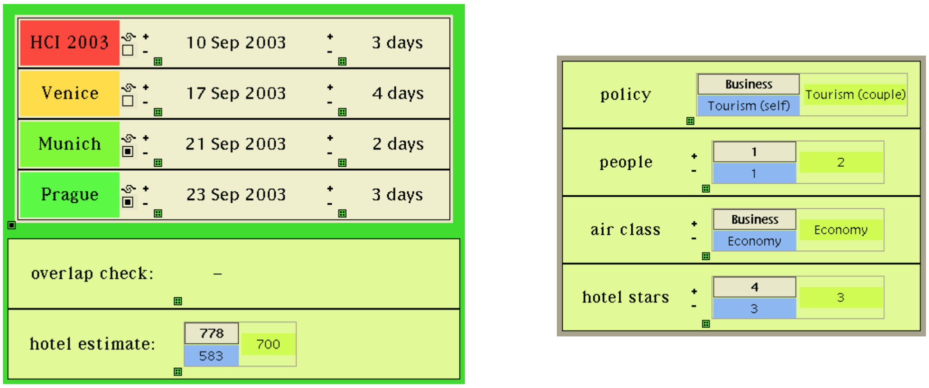
Benefits. We believe that HIBench’s transparent, open way of handling ambiguous queries is helpful for users. Rather than silently adopting some default behaviour – resolving the ambiguity in an arbitrary way in order to deliver a ‘best guess’ result – we believe it is appropriate for systems to highlight and capitalise on ambiguity; offering distinct results that arise from distinct interpretations, and letting the user choose.

2.5 ScheduleBlender: Tailoring in the Face of Ambiguous Context

Finally we show another system that, like HIBench, offers mixed-initiative creation of scenarios to support user exploration. ScheduleBlender is a demonstration of how a tool might support provisional factors in one’s personal calendar, such as undecided placement, duration and order of events. It also demonstrates how a system can behave in situations that demand a context-specific, tailored response – but where the system cannot determine, at any given time, which elements of the detected context are most relevant to the user.

Figure 8 shows a user experimenting with a travel itinerary, while the system performs lookups to provide an overall hotel-price estimate for the nights designated as being in foreign cities. However, this lookup involves an ambiguity: the application has access to travel policies set up in the past by this user – relating to business trips, single-person tourism, or tourism as a couple – but the user has not specified which of those policies applies to the current trip. Common approaches to such ambiguity would be to force the user to choose a policy, or for the system to choose one based on some heuristics (such as whichever policy was used most recently) and leave the user to change that choice if wanted. As in the case of HIBench, our belief is that explicitly revealing and working with the ambiguity is more helpful.

Establishing and Using Multiple Scenarios. By creating multiple scenarios, ScheduleBlender can apply all the applicable policies in parallel, dynamically updating them all as the trip details are changed.



HCI 2003	10 Sep 2003	3 days
Venice	17 Sep 2003	4 days
Munich	21 Sep 2003	2 days
Prague	23 Sep 2003	3 days

overlap check: –

hotel estimate: 778 / 583 / 700

policy	Business / Tourism (self)	Tourism (couple)
people	1 / 1	2
air class	Business / Economy	Economy
hotel stars	4 / 3	3

Fig. 8. A ScheduleBlender display showing a tentative travel plan, where the user has not yet specified who will be travelling. The display on the left shows the user's proposed order and duration of visits on a multi-city trip. Underneath are two pieces of information supplied by the system: a check for overlaps between the dates, and an estimate of the total hotel cost. For the hotel cost, the system is proposing three alternative values. The reason for this is shown in the display on the right: hotel price depends on the requested hotel class and room type, for which the system has three stored policies – and the user has not yet specified which policy to apply for this trip. The system has therefore applied all three in turn, providing the corresponding values for the user to consider.

One difference between ScheduleBlender and the preceding examples is that it shows how multiple scenarios are sometimes relevant in a highly localised way. While a hotel-cost calculation may need to take account of travel policies, other lookups within the same situation may call on other context elements – such as the preferred airlines to use for international flights, or the format to use in displaying dates. We are now in the early days of investigating how such localised scenario-management should best be presented to users.

Benefits. Today's computing environments include many features designed to help users by suggesting parameter values that, according to some designers' heuristics, are believed to be of interest to the user. One common example is the suggestion of which folder to use as the destination for saving a file. When the heuristics match an individual's pattern of use, these suggestions are welcome; when there is a mismatch, the same suggestions can become a great frustration.

Explicit enumeration and offering of alternatives, as demonstrated here, increases the chance that the value wanted on a given occasion is among those offered by the system. There are also cases where the alternatives may remind the user of forgotten options, or lead to serendipitous discoveries – for example, happening to find a trip for which the cost of business-class travel is so near the tourist price that it's worth making the upgrade.

The gathering momentum of context-aware ubiquitous computing brings many new opportunities for such tailored suggestions – and many opportunities for frustration. We believe that subjunctive-interface techniques have an important role to play in helping users to feel in control of the options available to them.

3 Related Work

The origin of the subjunctive interface research theme was the observation that users often want to explore alternatives, and that this is poorly supported by applications that always require the user to commit to unambiguous input values. Terry and Mynatt [12], examining this issue in the domain of open-ended design tasks, characterised existing applications as being tied to a 'single state document model' that works against the legitimate need of designers to work by experimentation, branching, evaluation and iteration. Terry et al. [13] later demonstrated an interface that supports a form of simultaneous development of alternative versions of a graphical design.

Recognising a corresponding need for exploration and experimentation in information access also has a long history: in 1989, Bates [14] argued against the classic model of information retrieval, that posits a user iteratively refining the definition of a query, with the goal of arriving at a single precise expression of his or her information needs. Instead, Bates argued, interfaces should support the less tidy reality of information-seeking in which users may adjust their expectations, try alternative approaches, and benefit from relevant information that they gather ('berrypick') along the way. Facilities for pursuing queries in parallel, as described in this paper, at least free users from having to encapsulate their interests in a monolithic query.

The emphasis on presenting alternatives in parallel can be seen as helping users to place alternatives into context. This concern has been addressed in various fields within HCI, including information visualisation [15] and the development of direct-manipulation interfaces [16]. For viewing and comparing alternatives within tabular data, for example, interfaces such as Polaris [17] provide rich facilities for constructing and reconfiguring tables, while the Table Lens [18] lets a user visualise chosen rows relative to each other and to the full range of values in each column. These tools, however, are limited to data suitable for row-and-column display, while the goal of subjunctive interfaces is that alternatives can be handled for any input or display region in an application.

Tools specifically designed to support interactive experimentation include the Influence Explorer [19] and Spotfire [20], both of which project numerical or ordinal data onto a two-dimensional graphical layout and provide interactive controls allowing a user to highlight data elements or ranges. Comparison is supported by the user's ability to switch the display rapidly and reversibly among different settings for the highlighting. Such dynamic switching is good for drawing attention to subtle distinctions, especially along some continuous range, but in other cases comparison may be better supported by simultaneous, side-by-side presentation of a set of key cases as provided by a subjunctive interface. In particular we believe that the use of widget multiplexers constitutes an instance of the 'small multiples' approach, which Tufte [21] supports with the comment that 'Multiples directly depict comparisons, the essence of statistical thinking' (p.105). Roberts [22] likewise recommends view multiplicity in computer interfaces as a way to encourage users to try out alternatives.

The cell-and-formula approach of C3W and HIBench aims to capitalise on users' familiarity with the spreadsheet model. Other well known spreadsheet-like systems include Forms/3 [23] and the Information Visualization Spreadsheet [24]. In general, such projects selectively adopt or abandon various features of what could be called the traditional spreadsheet. Our use of formulas conforms to the tradition that each formula determines the value of just one cell; on the other hand, our implementations are not alone in breaking free of the restriction that cells should contain only text or numbers, or the idea that cells should be positioned and named according to a tabular grid. And while we do use Microsoft Excel®'s API to support formulas in C3W, our handling of multiple scenarios is quite different from Excel's scenario management facilities – most notably in our insistence that cells' contents always be manipulated and displayed in place, rather than by calling on specialised input dialogues or pivot tables.

4 Future: Recipe-Based Exploration

As was mentioned in the Introduction, the example applications presented here have been developed in a spirit of exploring and illustrating interface concepts. While the studies carried out on the census browser show that users can understand facilities of the kind offered in a subjunctive interface, and that for certain tasks such an interface can give large, statistically significant performance benefits, clearly much remains to be done in delivering this work as a general contribution to HCI research.

Our main efforts are now focussed on the concepts demonstrated in the two spreadsheet-like systems: C3W (Sect. 2.2) and HIBench (Sect. 2.4). We consider these as early examples of tools supporting Recipe-Based Exploration – a specialisation of the subjunctive interface concept, suited to applications that can be characterised as spreadsheet-like acyclic graphs of cells and formulas. This application model, which we believe can subsume a wide range of today's applications (including, not least, spreadsheets), offers opportunities to simplify the handling of knock-on effects of alternative values assigned to cells within the dependency graph.

One goal for recipe-based exploration is to make effective use of mixed-initiative variation – i.e., where the system can take the initiative in creating alternative scenarios to help a user understand the range of possibilities on offer. We believe this will be important in handling the increasing amount of information offered by the systems pervading our working and living environments. When context-sensing computational agents clamour to provide people with information, recommendations, connections and a cornucopia of other services, no heuristics could sensibly boil down all these offers to a single recommended source. Users must be given the opportunity to explore and evaluate the alternatives that are on offer; in doing so, they should be supported by the multiple-scenario capabilities of subjunctive interfaces.

Acknowledgements

The work reported here could not have been carried out without the extensive support, of many kinds, provided by the staff, students and visitors of the Meme Media Laboratory in Hokkaido University and of the Natural Sciences ICT Competence Centre at the University of Copenhagen. Above all I am grateful to Yuzuru Tanaka in Hokkaido and Kasper Hornbæk in Copenhagen, for their continual collaboration, counselling and cajoling.

I also thank the organisers and other attendees of the Dagstuhl workshop – especially Klaus Jantke and Gunter Grieser – for creating an atmosphere so conducive to optimism and inspiration.

References

1. Lunzer, A.: 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), IOS Press (1999) 474–482
2. Lunzer, A., Hornbæk, K.: Side-by-side display and control of multiple scenarios: Subjunctive interfaces for exploring multi-attribute data. In: Proceedings of the Australian Conference on Computer-Human Interaction (OZCHI 2003), IEEE Computer Society Press, Los Alamitos, CA (2003) 202–210
3. Lunzer, A., Hornbæk, K.: Usability studies on a visualisation for parallel display and control of alternative scenarios. In: Proceedings of the 7th International Working Conference on Advanced Visual Interfaces (AVI 2004), ACM Press, New York, NY (2004) 125–132
4. Hofstadter, D.R.: Gödel, Escher, Bach: an Eternal Golden Braid. Basic Books (1979)
5. Hochheiser, H., Shneiderman, B.: Performance benefits of simultaneous over sequential menus as task complexity increases. International Journal of Human-Computer Interaction **12** (2000) 173–192
6. Fujima, J., Lunzer, A., Hornbæk, K., Tanaka, Y.: Clip, connect, clone: Combining application elements to build custom interfaces for information access. In: Proceedings of the 17th annual ACM symposium on user interface software and technology (UIST 2004), ACM Press, New York, NY (2004) To appear.
7. Tanaka, Y., Fujima, J.: Meme media and topica architectures for editing, distributing, and managing intellectual resources. In: Proceedings of 2000 Kyoto International Conference on Digital Libraries: Research and Practice. (2000) 208–216
8. Kolojejchick, J., Roth, S.F., Lucas, P.: Information appliances and tools in visage. IEEE Computer Graphics and Applications **17** (1997) 32–41
9. Goldstein, J., Roth, S.: Using aggregation and dynamic queries for exploring large data sets. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '94), ACM Press, New York, NY (1994) 23–29
10. Yee, K.P., Swearingen, K., Li, K., Hearst, M.: Faceted metadata for image search and browsing. In: Proceedings of the conference on human factors in computing systems (CHI 2003), ACM Press, New York, NY (2003) 401–408
11. Matsuoka, S.: The Longest Chronicle: History Informs. NTT (1996) In Japanese.

12. Terry, M., Mynatt, E.D.: Side views: persistent, on-demand previews for open-ended tasks. In: Proceedings of the 15th annual ACM symposium on user interface software and technology (UIST 2002), ACM Press, New York, NY (2002) 71–80
13. Terry, M., Mynatt, E.D., Nakakoji, K., Yamamoto, Y.: Variation in element and action: Supporting simultaneous development of alternative solutions. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI 2004), ACM Press, New York, NY (2004) 711–718
14. Bates, M.J.: The design of browsing and berrypicking techniques for the online search interface. Online Review **13** (1989) 407–424
15. Card, S.K., Mackinlay, J.D., Shneiderman, B.: Readings in Information Visualization. Morgan Kaufmann, San Francisco, CA (1999)
16. Shneiderman, B.: Direct manipulation: a step beyond programming languages. IEEE Computer **16** (1983) 57–68
17. Stolte, C., Tang, D., Hanrahan, P.: Polaris: a system for query, analysis and visualization of multidimensional relational databases. IEEE Transactions on Visualization and Computer Graphics **8** (2002) 52–65
18. Rao, R., Card, S.K.: The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '94), ACM Press, New York, NY (1994) 318–322
19. Tweedie, L., Spence, R., Dawkes, H., Su, H.: Externalising abstract mathematical models. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '96), ACM Press, New York, NY (1996) 406–412
20. Ahlberg, C., Shneiderman, B.: Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '94), ACM Press, New York, NY (1994) 313–317
21. Tufte, E.R.: Visual Explanations. Graphic Press, Cheshire, CT (1997)
22. Roberts, J.C.: Multiple-View and Multiform Visualization. In Erbacher, R., Pang, A., Wittenbrink, C., Roberts, J., eds.: Visual Data Exploration and Analysis VII, Proceedings of SPIE. Volume 3960., IS&T and SPIE (2000) 176–185
23. Burnett, M., Atwood, J., Djang, R.W., Gottfried, H., Reichwein, J., Yang, S.: Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. Journal of Functional Programming **11** (2001) 155–206
24. Chi, E.H., Riedl, J., Barry, P., Konstan, J.: Principles for information visualization spreadsheets. IEEE Computer Graphics and Applications **18** (1998) 30–38