(In Proceedings of IFIP TC. 13 International Conference on Human-Computer Interaction (INTERACT '99), Aug–Sept 1999, Edinburgh, Scotland, 474–482)

Choice and comparison where the user wants them: subjunctive interfaces for computer-supported exploration

Aran Lunzer

Meme Media Laboratory, Hokkaido University, Sapporo 060-8628, Japan
EMail: *aran@meme.hokudai.ac.jp*

There are many kinds of computer system that are intended to support users in performing an exploration among a range of possible results. However, such tools often allow only narrow progress through the result space, forcing users to trade off the breadth of search they would like to pursue against the time and effort required. The aim of the 'subjunctive interface' concept is to make broader searches more manageable, by letting users propose multiple alternative values for each parameter where normally only a single value can be supplied, and by supporting the viewing and comparison of the various resulting outcomes.

This paper clarifies the motivation and principles underlying the subjunctive interface concept, describes implementation work that illustrates the approach, and outlines directions for further pursuit of this research.

Keywords: user interface, interaction styles, interaction techniques, cognitive dimensions, subjunctive interface, parameter exploration, medical image segmentation

1 INTRODUCTION

How pleasing it is that flight enquiries, like many other services, are now available through the World Wide Web. How depressing that the enquiry procedure is just as constrained as it ever was at the travel agency: typically, information about available seats is only revealed for a single fully specified journey at a time—in particular, with precise dates for departure and return. So if one wishes to compare, for example, the available schedules and fares for a two- or three-week trip to Scotland starting during any weekend in August, one has to submit a laborious sequence of individual queries and to make a separate record of the most interesting-looking items returned in each case—maybe putting up with an occasional response of the form 'Nothing available. Please try again.'.

Should the system be more helpful? It is entirely within the scope of modern database technology to deal with much richer query specification, but this seems to be overridden by an imperative to keep the interface simple. Maybe the system should at least avoid producing the 'nothing available' response, by attempting to find a result that *almost* matches the query—but what compromises are to be assumed acceptable? Some travellers may want to try other dates; some might be insistent on keeping the separation of outbound and return date constant; some might be restricted, as suggested above, to travelling at the weekend. There may be desires regarding airline, route, flight duration and, naturally, cost. The complexity of attempting to cater for compromises may be just as much a factor in the decision to support only fully-specified queries as is the need for efficiency in handling the enormous transaction traffic. Furthermore, this is a domain in which one cannot generally extrapolate among results: even if some special-deal fare is sold out on one day it may be available on the day before or after, or there may be similar or better deals to other nearby cities. An exhaustive (and tedious) search of the full range of results of interest is therefore the only way to discover all the possibilities.

In the *cognitive dimensions* framework being developed by Thomas Green and others to express the usability or otherwise of interactive computer systems (e.g., see Green & Blackwell 1998; Green & Petre 1996), all interaction is characterised as the building or modification of an information structure. Flight enquiry fits into their broad category of *exploratory design*, involving a mixture of adding to and modifying the structure that is being built (in this case, a query that produces acceptable results), and with the crucial

property that the user does not know in advance which combination of these activities will result in a desirable end state. Other activities in this category would include simulation-based exploration of real-world entities ranging from astrophysics to home decoration, and the design of purely computer-generated artifacts such as Web pages and animations. In all cases it is the presence of complexity at some level—whether in the processing that generates results from parameter settings, or in the synergistic effects of numerous simple elements—that makes it hard for the user to predict the result that will emerge from a given specification.

Cognitive dimensions that are relevant to exploration include *visibility*, which concerns the ability to obtain a view of some desired information, *juxtaposability*, being the ability to place different items side by side (helpful in making comparisons), *viscosity*, which expresses cost of changing something that has already been specified, and *premature commitment*, which captures the concept of users being forced to make decisions before sufficient information is available. Some difficulties in using exploration-support systems can be described as follows:

- systems in which results are only available one at a time in response to a precise specification have poor *visibility*—it takes a lot of separate attempts to work through a range of alternatives;
- some systems have high *viscosity* insofar as moving on from one request to another involves numerous fiddly actions;
- poor *juxtaposability* arises where the handling of one request causes its results to replace the previous ones, making comparison difficult;
- the above combine to engender a feeling of *premature commitment* in making choices, since a user feels pressured to use the first reasonable result rather than keep trying alternatives.

1.1 Improved support for static data

For exploration among collections of data that can be held entirely within a local computer and processed quickly enough to produce highly dynamic visualisations, a number of techniques have been developed to assist the user. A well known exemplar is the *dynamic query* approach (e.g., Ahlberg and Wistrand 1995). These tools keep *visibility* high by being able to map the entire data set into a single visualisation with interactive zooming; *viscosity* low by mapping parameters to sliders and filtering the display in real time (less than 100ms) as the sliders are manipulated; *juxtaposability* is improved by the ability to move rapidly back and forth between settings; thus also reducing *premature commitment* insofar as every setting is easily reversible. The key is that you can summarise the items in the collection and work with them at an overview level, requesting detailed views just for those that look particularly interesting.

Arising from these ideas, one approach to improving exploratory tasks is to generate in advance a set of data items representing the range of results that are of interest. For example, if a suitable subset of the worldwide flight database could be downloaded to one's own computer it would become straightforward to use dynamic-query facilities to explore the available options. The challenge then becomes that of defining and collating this 'suitable' set of results, especially where—as in explorations based on simulation or design—the results don't actually exist until they are specifically requested. Existing approaches range from explicit user-defined result ranges, as in the author's earlier work on *result-space reconnaissance* (Lunzer 1996), to automated sampling such as in Design Galleries (Marks *et al.* 1997) or even randomised adjustment as in Mutator (Todd & Latham 1992).

1.2 When static sampling is not enough

Transforming an exploratory design task into the postprocessing of a set of sample results is not always satisfactory. Not only are the non-sampled results sidelined, but it remains difficult to steer the search towards regions that are interesting—analogous to the difference between visualisation of the end results of a simulation, as opposed to *computational steering* that allows an experimenter to direct progress on-the-fly.

So how might one reconcile the goal of working with multiple alternative results at the same time—to support overviews, comparisons and dynamic filtering—with the goal of interactively steering the specification of these results? This is the question that led to the proposal for *subjunctive interfaces*, as first described in (Lunzer 1998).[†] The following section outlines how the approach is intended to improve exploration support; subsequent sections discuss an implemented example, the principles of what is being proposed, and plans for further work.

[†]The subjunctive mood deals with acts or states not as facts but as possibilities—how things might be, considered in contrast with how they currently are. The term *subjunctive interface* was chosen in appreciation of Douglas R. Hofstadter's idea of the *Subjunc-TV*, a magical television that would be able to show alternative versions of a given broadcast corresponding to different circumstances chosen by the viewer—for example, showing how a football match would have played out if the weather had been fi ne instead of wet.

2 BROADENING ONE'S OUTLOOK

This section gives examples of how the subjunctive interface concept can alleviate a user's dissatisfaction with only being able to pursue a single, narrow path of interaction within a rich result space governed by parameter settings.

2.1 Pursuing provisional choices

During parameter-based exploration, a user may often be asked to specify a single value for some parameter for which he or she does not (yet) have a clear preference. For example, the flight-enquiry interface mentioned above can only handle queries based on precise dates for departure and return, whereas the customer in our scenario has a range of alternative dates in mind. To meet the constraints of the underlying computer system while also satisfying the user's wish to explore a range of alternative values, a 'subjunctive' approach would involve the creation of separate processing 'realities' in which the different values could be handled—one reality for each alternative that the user wants to try. With each reality maintaining its own thread of control, the handling of the multiple queries could appear to the user to be simultaneous, so the various results that could normally only be obtained through a series of separate interactions would become available at the same time. The issue that then arises is how to provide an interface that supports the examination, comparison, filtering and so on of these multiple results.

Figure 1 shows a working model built as a canonical example of subjunctive-interface principles. It simulates the launch and flight of unpowered projectiles under the influence of gravity and air resistance. Launch energy is fixed, but the projectile weight, the turret angle and headwind speed can be adjusted. If the user is interested in comparing flight paths that correspond to these settings—for example, to find a weight of projectile that can travel far under a range of wind conditions—an interface that supports just a single set of parameter values at a time is painfully restrictive. The user might set one value for weight, then vary the wind speed to generate various results, then change the weight and vary the wind speed again, trying to remember the previous results to compare them with the new ones. In figure 1 we see that the user has specified two separate settings for wind-speed, and three separate values for projectile weight. In this case it is simple to overlay the alternative realities to support overview and comparison of the different parameter settings and the different flight paths that result.

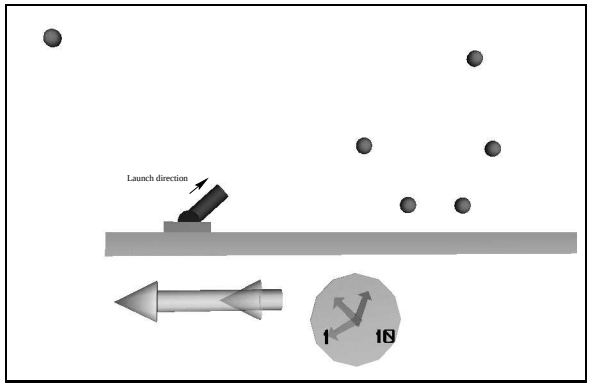

Figure 1: A snapshot of a simple subjunctive interface built using the locally developed 'IntelligentBox' 3D construction environment (Okada & Tanaka, 1995). Here a user is exploring two different speeds of headwind (set using the large arrow at bottom left) and three values of weight (set using the dial) on the trajectory of a launched ball. The widget states corresponding to these different settings are overlaid using semi-transparent rendering. The six resulting parameter combinations lead to independent but synchronised launches; this image shows the six balls that emerged simultaneously a short time ago, all following different trajectories.

In general, where a computer-based exploration would normally demand a single value—a date for a flight, an additional criterion to refine a query, a yes/no decision on forcing a page-break before some part of a document—our aim is to free the user to supply multiple alternatives if that will help the pursuit of the task. In cognitive dimension terms such support contributes to *provisionality*, i.e., reduced commitment to actions, and thus directly counteracts premature commitment.

2.2 Steering multiple alternatives

Although the 'dynamic query' tools make parameter adjustment cheap and its feedback instantaneous, the user is still only steering a single path through the result space. A subjunctive-style interface supporting multiple alternative settings can multiply the rate of examining alternative result-space regions.

For example, in the flight-booking scenario one may decide to switch from an originally specified destination (say, city x) to a nearby alternative city y, or to try flying into x but home from y. A user who has already specified a range of alternative dates is now likely to want to re-submit all those queries with the new itineraries. In a standard interface the re-specification of all the queries with the new routes would be almost unthinkable. But if the realities created to handle the various dates are linked so that they share all attributes apart from the date, a change to an independent parameter such as the destination can be reflected automatically in all the alternative cases.

The projectiles model exhibits this form of parameter sharing. For example, of the six realities shown in the picture, three are using one value of wind-speed while three are using the other. If the user decides to reduce the higher of the wind settings, all three realities using that setting will automatically change. Such sharing can be said to improve the system's rating on the *visibility* scale.

2.3 Adjustment from a distance

One of the frustrations that can arise during parameter-based exploration is being unable to see the effect of some setting at the time when it is being made. For example, a flight-enquiry interface may have one 'screen' on which the user specifies dates and the route, which then disappears during query processing and is replaced by the result display. To try other settings the user must abandon the result display and return to the setting screen. This kind of *viscosity* is side-stepped by interfaces such as the dynamic query tools, by casting the exploration into the form of a single view manipulated using controls that fit alongside it. For the more general case, a subjunctive-interface approach should let users work on one view while being able to see simultaneously what would be appearing if the viewpoint were somewhere else instead—e.g., by use of multiple windows connected to linked realities.

Since many explorations are driven not just on the basis of a mapping from parameter state to results, but by the history or path of interactions, a subjunctive approach that allows a user to revisit a decision early in the history will require a mechanism for re-playing all user interactions that may have a new effect under the new setting.

2.4 Combining partial results

Rather than regarding the various realities as separate entities illuminating alternative possibilities, in some cases having the various different solutions available simultaneously can help the user to pull complementary parts of those solutions together into one result that could not otherwise have been generated.

An example of this form of result combination appears in the following section, which describes recent implementation work on applying subjunctive techniques to enhance an existing tool.

3 APPLICATION EXAMPLE

The system currently being used as a test platform for implementing subjunctive interface ideas is 3DVIEWNIX, a portable Unix-based medical-image processing tool.[‡]

Work to date has focussed on the support for image segmentation—i.e., the division of a multi-dimensional scene into distinct regions of interest to a doctor, such as individual bones or organs. The segmentation support in the current release of 3DVIEWNIX (1.2) includes a simple form of *live wire* boundary-detection tool that must be 'trained' to detect the kind of edge that is of interest.[§] Since only one instance of training can be in force at a time, with this tool there is currently poor support for tracing accurately and reproducibly any boundary that has different features in different regions—such as that highlighted in figure 2. We realised that if 3DVIEWNIX were extended to support multiple 'subjunctive' states we could let the user handle a range of alternative training settings simultaneously, and hence overcome this restriction.

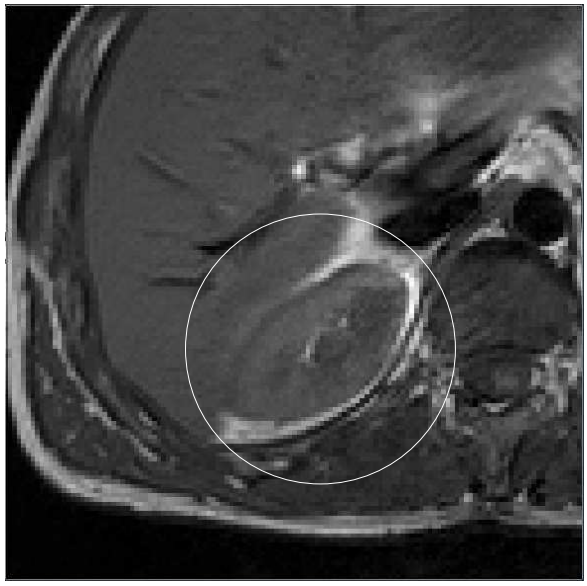

Figure 2: An example of a boundary with different characteristics in different regions. Within the white circle on this MRI image of a horizontal slice through a person's abdomen is the outline of a kidney. However, the nature of the boundary on the left (where it abuts the liver) is rather different from that of the rest of the organ. This presents a problem to some kinds of image-segmentation support tools.

3.1 Implemented subjunctive features

To date, the following operations have been implemented for the 3DVIEWNIX segmentation tools:

Reality cloning: Creation of a new reality based

[‡]3DVIEWNIX has been developed over several years by the Medical Image Processing Group of the Department of Radiology at the University of Pennsylvania, Pittsburgh. The system and its source code can be purchased from MIPG for research use.

[§]The 'live wire' technique has been developed principally at Brigham Young University; for an overview see (Mortensen & Barrett 1998).

on a copy of the current application state. The user's actions are directed to the new reality, leaving the previous reality as a snapshot that can be returned to at any time. As well as allowing the realities to be developed further in parallel, this operation can also be used simply as a way of freezing the current state for later use. This would be useful, for example, in an application in which the user finds some result that might be satisfactory, wants to explore further in case a still better result can be found, but also wants to keep tabs on this state in case it turns out to have been the best available.

Event echoing: Enabling/disabling the broadcast of interface events (at the level of mouse movements and clicks) to all currently active realities. If a user has set up a number of realities with different parameters, this facility allows the same operations to be carried out simultaneously on all of them. Alternatively, if the difference between the realities is in the currently selected meaning of mouse input (for example, one is supporting live-wire while another is performing manual tracing), event echoing allows the user to observe the different effects of these various operations. Given appropriate provision to mix the realities' result displays, the different results can be watched simultaneously.

Event echoing can be enabled and disabled at any time, for example to make adjustments to just one reality in the midst of a broadcast operation. However, in the current implementation it is the user's responsibility to ensure that each reality is in an appropriate state to receive the broadcast events.

Primary reality switching: Moving the application on to the next/previous reality, replacing the current operation context, including the display. During the use of event-echoing, operations to copy the results from one reality through to others use the 'primary reality' as the source.

3.2 Usage scenario

Figure 3 shows the behaviour of the 'live wire' algorithm when trained for the liver/kidney boundary as found in the displayed slice. Although it follows the edge well in the trained region (and would also work for such an edge found in other slices within this study), it is not useful for the remainder of the kidney boundary. Normally the boundary would have to be guided manually around any untrained segments; although this would not be especially laborious for this simple shape, an undesirable side-effect is the risk of inconsistent edge paths being plotted for different slices and different studies, making comparisons difficult.

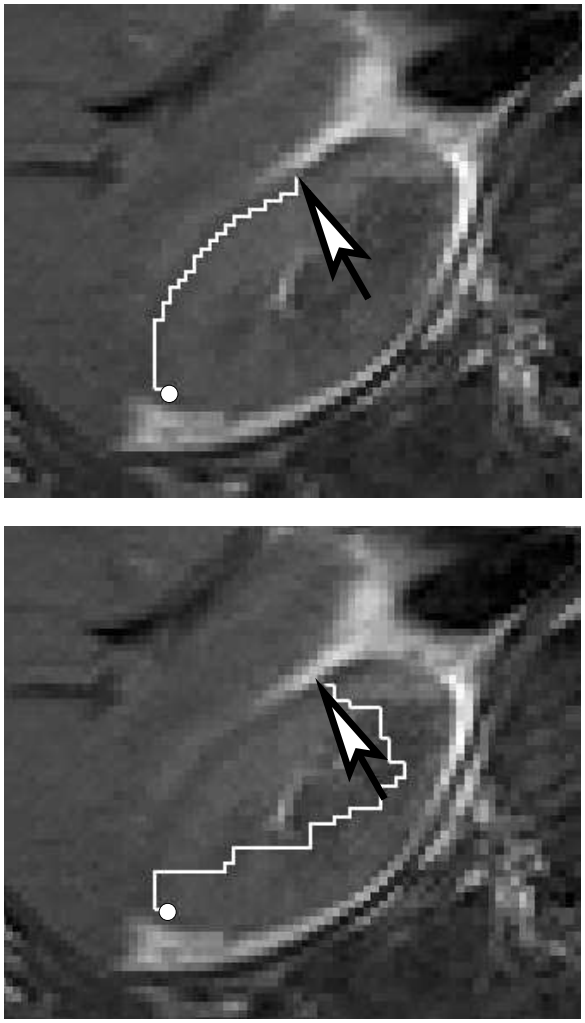

Figure 3: A segmentation in progress using the 'live wire' tool, after training it to follow the liver/kidney boundary. The user clicks somewhere on the desired boundary to establish a start point (here highlighted with a white blob). Thereafter, a live wire is continuously plotted linking the start point to the pointer's current position (shown by the arrow); clicking the mouse again captures the displayed wire and establishes a new start point. The wire follows the trained boundary well (upper image), but when the mouse is outside the region of good fit for the current training (lower image) the algorithm may be attracted to paths other than the intended one.

With the subjunctive interface features an alternative approach is available: the definition of multiple training settings to handle various parts of the boundary, by cloning the application state and training each one using a different edge section. In this case just one further definition suffices; figure 4 shows an instant during outlining using 'event echoing' to join two application states.

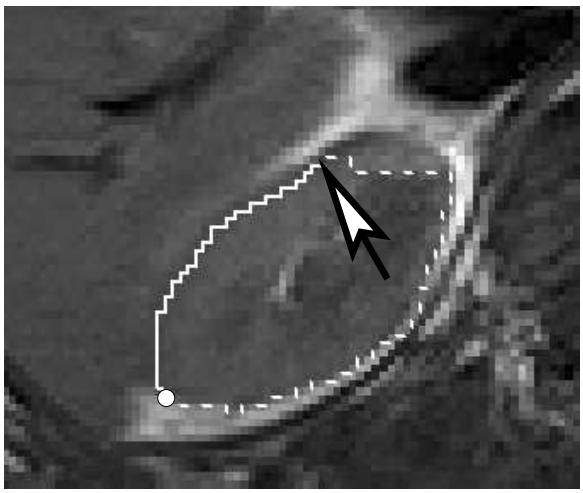

Figure 4: A subjunctive display. Here a second ‘reality’ has been added with different training, and both are performing the live-wire operation in parallel by means of event-echoing. The wire contributed by the secondary reality is displayed as a dashed line, its attraction to the bright edge of the kidney being so strong that it almost spans the whole path from the present mouse position back to the start point. By assembling the path using selected contributions from the different realities, an entire outline can be built quickly.

3.3 Experiences

These initial experiences of implementing and experimenting with subjunctive interface features for 3DVIEWNIX have been rewarding. At this stage of still learning how the segmentation activity is steered by the environment’s parameter settings, and thus how best the addition of multiple realities might be of benefit, simply the facility of storing and recalling a snapshot of the entire set of parameters while experimenting with alternative values provides considerable relief. Merging the interactive responses of multiple realities has also proved to be practicable, and appears to resolve some of the difficulties of using the original system.

With the guidance of our medical-faculty colleagues we aim to refine the subjunctive facilities, for example to decide on how to handle gracefully any divergence between the states of realities that are being driven simultaneously. We aim to produce demonstrations of how these new facilities enable novel interaction techniques with benefits over existing medical image handling procedures, both in the area of segmentation and in other operational aspects in which exploration appears to be needlessly difficult.

4 PRINCIPLES AND RELATED WORK

4.1 What is a subjunctive interface?

Starting from observation of difficulties commonly encountered in the use of computer systems for exploratory tasks, this work aims to improve support for these tasks. In particular, it was noticed that facilities now commonly available for exploration of static data—overview, juxtaposition, reversible interaction—are not yet supported for exploration involving dynamically generated results (although Truvé (1995) provided a wake-up call and a useful example). A ‘subjunctive’ interface is simply one that attempts to provide these facilities, by supporting the modelling, comparison and adjustment of numerous alternative states simultaneously.

4.2 An extension, not a new tool

Support for simultaneous exploration of multiple results could be provided using novel kinds of interface that re-cast the task into a form that particularly suits this kind of interaction—much as *dynamic query* tools re-cast parameter-based data filtering into a combination of sliders and a single two-dimensional view. However, given an existing tool that supports some exploratory task, there are various incentives to providing the additional support as an extension to the existing interface components—i.e., for provisional settings to be specified using the same widgets as usual, and for the different results of provisional choices to be displayed using the same output devices.

One benefit is the ability to stick with an interface has been optimised to suit the task in question, and that is familiar to existing users of the system. Learning to use new exploration mechanisms should be less of a burden if they are based on the existing controls than if based on new exploration-specific tools and abstractions.

Given that a system designer cannot predict all the kinds of exploration a user may wish to do, it is desirable to let the standard facilities constitute a framework within which a user constructs the facilities needed for each occasion of exploration—in line with the *visual formalisms* approach proposed by Nardi and Zarmer (1993). Facilities for *programming by demonstration* (e.g., Myers 1998) likewise seek to integrate the definition of occasion-specific facilities with the normal use of the system, aiming to minimise any self-conscious switch between ‘use’ and ‘programming’. Perhaps our ideal should be to engender in users the feeling that it’s acceptable to make tentative rather than firm decisions on any setting in the interface.

4.3 Interface challenges

Particular challenges arise in helping the user to view and work with the various alternative results, and to understand their relationship to the controlling parameters. The projectiles model provides help in isolating individual cases temporarily to clarify which tentative parameter settings belong with which results: while the user is adjusting any parameter value the interface temporarily dims out all objects in realities other than those being fed by that value. Similarly, if any ball is probed with the mouse, all widgets other than those involved in its creation are faded away so the user can see which settings gave rise to a particular flight path.

A similar need can arise in Computer-Supported Collaborative Work, when collaborative tools have to handle alternative versions of some information to reflect choices or enhancements made by separate users. The work on conflict display and resolution in the Timewarp collaborative toolkit (Edwards & Mynatt 1997) provides excellent guidance on the interface and system-level handling of such presentation issues.

In some domains, alternative results cannot be merged just by overlaying them—for example, textual lists that would become unreadable, or images that would become a jumble. Various policies could be pursued—for example, one might merge salient aspects of the results into a single list or image, as shown in the 3DVIEWNIX example above, or one could show the various results in separate regions of the display. A disadvantage of the latter approach is that the user has to pay attention to multiple items rather than having the results all placed at a single focus of attention, but in some cases this could be compensated by mechanisms for increasing the salience of results identified as being of particular importance.

4.4 Processing model

Underlying the subjunctive display one needs a system model that can represent and process the alternative results. A simplistic method would be to replicate the entire underlying environment, again mirroring CSCW mechanisms in which different users each run their own copy of the software connected only by update propagation. However, one can envisage considerably more efficient approaches based on the techniques being developed to support multiple versions of system state for provision of rich undo/redo facilities (e.g., Abowd & Dix 1992; Berlage 1994), or Atwood *et al.*'s (1996) powerful support for history reversion and rewriting as offered by 'time travel' facilities in visual programming environments.

5 RESEARCH DIRECTIONS

The subjunctive interface concept is still young, but it is hoped that the ideas described so far will motivate the enhancement of various kinds of exploration-support system.

For each such branch of investigation, empirical testing will provide the litmus test as to whether the additional complexity is providing any benefit in the ability of users to find good results for their explorations. As part of this evaluation there will be much to learn about how the addition of subjunctive facilities changes the paths that users normally follow in their explorations among dynamically constructed results. There seem to be few existing studies of such paths, although Lee's (1998) account of users' progress in database exploration sessions provides welcome insight, especially as his work bridges the domains of static data in databases and the dynamic construction of views derived from the data.

Another field that awaits experimentation is support for parameters whose values occupy a continuous range, rather than a few discrete alternatives. Typically a user working with such a parameter would choose discrete values to represent what are felt to be salient regions within its range, then as long as there are no strong discontinuities in parameter effect between these values the results may be interpreted with the help of interpolation. Where discontinuities do exist, a subjunctive-interface approach that allows detailed sampling and simultaneous consideration of the parameter's effects may be useful in highlighting the nature of the affected operational regions.

As shown by the work by Ahlberg and Truvé (1995) and Tweedie (1997) on 'design spaces' for input devices and interactive visualisations respectively, the development of a design space describing subjunctive behaviour in general will be a major undertaking. However, even at this early, implementation-biassed stage of exploring the concept we can benefit from these classifications in considering how it may eventually be generalised. A long-term goal would be a stage where subjunctive interaction mechanisms can be built into frameworks for application construction so that, like the windowing facilities that effectively come 'for free' for applications built with today's interface toolkits, implementors will only need to supply minimal details to be able to offer subjunctive facilities within their applications.

6 CONCLUSIONS

This paper has reported progress on the characterisation of *subjunctive interface* features in terms of the

cognitive dimensions framework, and has introduced recent implementation work that is starting to illustrate benefits derived from this interface approach.

Although simple in concept, the idea of giving exploratory interfaces 'subjunctive' behaviour—representing simultaneously what would happen under the influence of a variety of choices—appears to give rise to a complex yet potentially rewarding area of interaction research. In the many domains of computer-supported human activity in which the computer can only act as a tool steered by the user to illuminate interesting results, this new form of support brings the hope of greater confidence in that steering and in the quality of the results that are found.

ACKNOWLEDGEMENTS

The author is currently a research assistant funded under a donation by Hitachi Software; parts of this work were undertaken during a previous term at the same laboratory as a European Union postdoctoral research fellow. I am very grateful to the staff, students and visiting researchers for providing a supportive atmosphere, and especially to Professor Yuzuru Tanaka, laboratory director, and to Matthew Chalmers for discussions and encouragement. The medical image used here is a sample from the PACS web site of the University Hospital of Geneva.

REFERENCES

Abowd, G.D. and Dix, A.J. (1992) 'Giving Undo Attention.' *Interacting with Computers* **4**(3), 317–342.

Ahlberg, C. and Truv´e, S. (1995) 'Exploring Terra Incognita in the Design Space of Query Devices.' In *Proceedings of Engineering for Human-Computer Interaction (EHCI '95)*, IFIP Transactions series, Chapman & Hall.

Ahlberg, C. and Wistrand E. (1995) 'IVEE: An Information Visualization & Exploration Environment.' In *Proceedings of IEEE Information Visualization '95*, Atlanta, GA, 66–73.

Atwood, J.W., Jr., Burnett, M.M., Walpole, R.A., Wilcox, E.M. and Yang, S. (1996) 'Steering Programs via Time Travel.' In *Proceedings of IEEE Symposium on Visual Languages (VL'96)*, Boulder, CO, 4–11.

Berlage, T. (1994) 'A Selective Undo Mechanism for Graphical User Interfaces Based on Command Objects.' *ACM Transactions on Human-Computer Interaction* **1**(3), September 1994, 269–294.

Edwards, W.K. and Mynatt, E.D. (1997) 'Timewarp: Techniques for Autonomous Collaboration.' In *Proceedings of ACM CHI 97*, Atlanta, GA, 218–225.

Green, T.R.G. and Blackwell, A.F. (1998) 'Cognitive Dimensions of Information Artefacts: a tutorial.' Available for download, via http://www.ndirect.co.uk/∼thomas.green/.

Green, T.R.G. and Petre, M. (1996) 'Usability analysis of visual programming environments: a "cognitive dimensions" framework.' *Journal of Visual Languages and Computing* **7**, 131–174.

Lee, J.P. (1998) *A Systems and Process Model for Data Exploration.* Ph.D. thesis. Computer Science Department, University of Massachusetts Lowell.

Lunzer, A. (1996) *Reconnaissance: a widely applicable approach encouraging well-informed choices in computer-based tasks.* Ph.D. thesis. TR–1996–4, Department of Computing Science, University of Glasgow, Scotland. Feb 1996. 266pp.

Lunzer, A. (1998) 'Towards the Subjunctive Interface: General Support for Parameter Exploration by Overlaying Alternative Application States.' In *Late Breaking Hot Topics Proceedings of IEEE Visualization '98*, Research Triangle Park, NC, 45–48.

Marks, J., Andalman, B., Beardsley, P.A., Freeman, W., Gibson, S., Hodgins, J., Kang, T., Mirtich, B., Pfi ster, H., Ruml, W., Ryall, K., Seims, J. and Shieber, S. (1997) 'Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation'. In *Proceedings of SIGGRAPH 97*, Los Angeles, CA, 389–400.

Mortensen, E.N. and Barrett, W.A. (1998) 'Interactive Segmentation with Intelligent Scissors.' *Graphical Models and Image Processing* **60**, 349–384.

Myers, B.A. (1998) 'Scripting Graphical Applications by Demonstration.' In *Proceedings of ACM CHI 98*, Los Angeles, CA, 534–541.

Nardi, B.A. and Zarmer, C.L. (1993) 'Beyond Models and Metaphors: Visual Formalisms in User Interface Design'. *Journal of Visual Languages and Computing* **4**, 5–33.

Okada, Y. and Tanaka, Y. (1995) 'IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphics Applications.' In *Proceedings of IEEE Computer Animation '95*, Geneva, Switzerland, 114–125.

Todd, S. and Latham, W. (1992) *Evolutionary Art and Computers.* Academic Press Ltd.

Truv´e, S. (1995) 'Dynamic what-if analysis: exploring computational dependencies with slidercells and micrographs.' In *Conference Companion of ACM CHI 95*, Denver, CO, 280–281.

Tweedie, L. (1997) 'Characterizing Interactive Externalizations.' In *Proceedings of ACM CHI 97*, Atlanta, GA, 375–382.