Emerging designs for hierarchical Web sites offer some insight into flexible, adaptive, and responsive dialogs that invite users to interact, not click out in frustration.

Saverio Perugini and Naren Ramakrishnan

Interacting with Web Hierarchies

From store catalogs and news services, to community classifieds, hierarchies are ubiquitous. They are at once a natural way to organize, present, and navigate online information and a major source of user frustration when they do not mirror the user's conception of the information-seeking process. In an effort to get the user to the desired information more quickly, interface designers have developed many variants of the traditional drill-down motif.

Web site interfaces are a particularly good fit for hierarchies in the broadest sense of that idea—a classification with multiple attributes, not necessarily a tree structure. Several adaptive interface designs are emerging that support flexible navigation orders, exposing and exploring dependencies, and procedural information-seeking tasks. These designs offer a sneak peek into the variety of ways organizations have improvised on the vanilla hierarchy. In describing them, we provide a context and vocabulary for thinking about hierarchical Web sites and their design: How should user tasks and a site's content combine with design to shape and facilitate interactions?

We have identified three features that interfaces to information hierarchies should include—flexible navigation orders, the ability to expose and explore dependencies, and support for procedural tasks—and offer some examples of these features.

FLEXIBLE NAVIGATION ORDERS

With flexible navigation orders, inputs can arrive in any user-specified order. Consider a user of an online car-shopping site, such as http://autotrader.com:

> "I am here to buy a car. The most important criteria for me are price, safety rating, and color, in that order. What's available?"

In conducting such a search, the most helpful hierarchy from the user's view is one that organizes attributes in the specified preference: price at the top, safety rating next, and color after that. Such a hierarchy dictates a total order over automobile facets, since the system must receive attribute values in a particular order. The user would be hard-pressed to find such a classification, however, since most automobile sites we surveyed organize their inventory along more traditional facets such as make, model, and year.

The next best choice, then, is an interface that enumerates all possible navigation orders or one that lets users generate their own totally ordered hierarchy.

Enumerative interface designs

Enumerative interfaces support flexible navigation orders by exposing all the individual choices for unspecified facets at every level. The Epicurious site (http://epicurious.com) is an example, as shown in Figure 1. You can view enumerated interfaces as multiple totally ordered hierarchies, one for each of the $n!$ possible navigation orders, where n is the number of facets that the classification considers. To avoid overwhelming the user with all such orders, sites such as Kelley Blue Book online (http://kbb.com) present only a few facets at each level, yielding a partially enumerated hyperlink structure.

1520-9202/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

Figure 1. Using an enumerative interface to browse recipes.

(a)

(b)

The screen enumerates all the individual choices for each facet (main ingredient, cuisine, preparation method, season/occasion, and so on) and the user selects bake, a preparation method (a). The next screen (b) no longer offers a choice for preparation method. Such enumerative interfaces support flexible navigation orders by exposing all choices for unspecified facets at every level.

Other enumerative designs use pull-down menus that hide enumerated choices behind user interface components. An example is at http://sonystyle.com, a site for users to browse digital cameras and camcorders. Other interfaces, such as that for Apple's iTunes, present all facets and choices on a single page, typically in tabular form, and explicitly relay to the user how the system prunes attribute values during an interaction. These interfaces are enumerative, but support browsing within a page rather than between pages.

Enumerative designs that present all facets risk becoming cluttered if the product has too many facets of classification. Automobiles, for example, can have more than 20 individual attributes, which can give rise to issues related to limited screen real estate.

Generative interface designs

In contrast to enumerative interfaces, generative interfaces support the construction of a totally ordered hierarchy for subsequent browsing. The site requests a navigation order first and subsequently generates a totally ordered hierarchy conforming to it. Figure 2 illustrates how such an interface might work for the car shopping scenario given earlier.

An interface to support such a task might let the user specify the navigational order of the classification's facets on the fly, thus meshing drill-down browsing with order specification. The user can specify the navigational order up front, as in Figure 2, or incrementally, depending on the task. Generative interfaces react to the user's directives and organize subsequent pages to reconcile what is left of the hierarchy with what the user wants to do next.

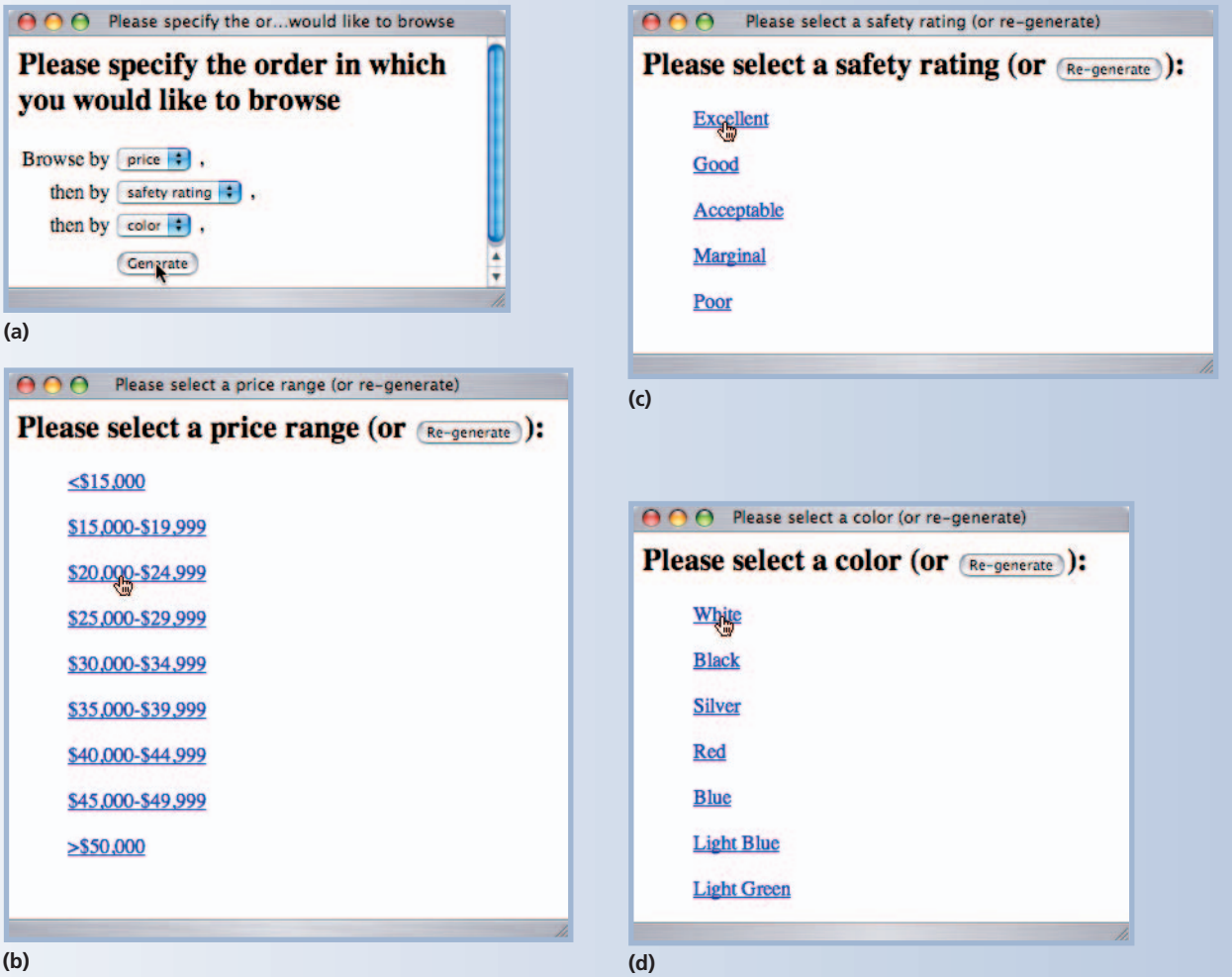
Out-of-turn interface

Figure 3 shows an out-of-turn interface, which is somewhere between enumerative and generative designs. This interface supports an online car shopper with a different information-seeking goal:

> "I am here to buy a car. I'm interested only in fuel-efficient cars—those that yield greater than 40 miles per gallon (mpg)—and would like to begin my search from there."

Unlike interfaces based on facets, pull-down menus, or a table, the out-of-turn

Figure 2. Using a generative interface to support online car shopping.



The user specifies browsing by price, safety rating, and color, in that order (a). The site generates a totally ordered classification, and the user progressively clicks on hyperlinks labeled "$20,000–$24,999" (b), "excellent" (c), and "white" (d) to retrieve a list of cars with these features or additional classification facets, such as make and model.

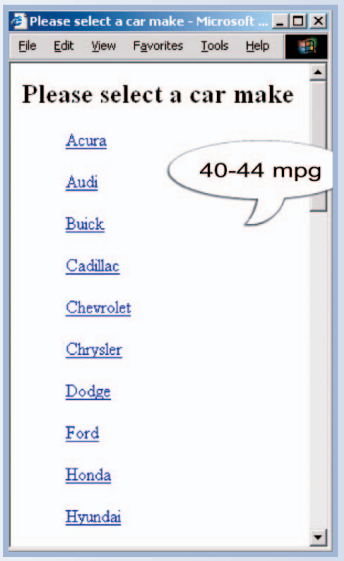
interface does not enumerate all the individual choices at each level. In contrast to a generative interface, it does not let the user browse in any order. Rather, it simply empowers the user to supply out-of-turn values for facets other than that on the current Web page if the current organization is not to the user's liking.

The user can supply input out of turn by speaking to the browser through a voice interface, as in Figure 3, or by using a toolbar embedded in the browser. Because an out-of-turn interface does not enumerate all remaining facets at each level, users must have meta-inquiry capabilities so that they can inquire about unspecified attributes. However, out-of-turn interaction does not imply free-form input, only the ability to communicate input that the interface normally solicits later in the interaction.

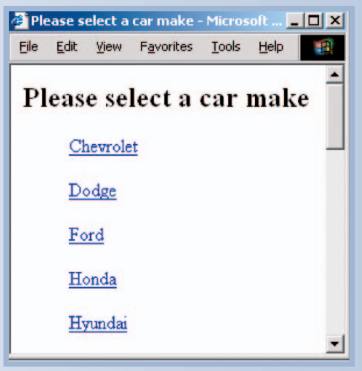
Comparisons

As these examples imply, the nature of a hierarchy's content affects which interface designs are most suitable for interaction. Some content is inherently faceted—when presented in a hierarchy, each level corresponds to a classification facet—be it main ingredient, cuisine, or preparation method at Epicurious, or make, model, or year for the Kelley Blue Book. On the other hand, large directories of links to Web sites, such as Yahoo! and the Open Directory Project (ODP, http://dmoz.org), present unfaceted content. ODP's top level, for example, presents Arts, Music, and Sports hyperlinks, which each correspond to nothing more specific than a topic. Because there is no concept of a facet, purely navigational links typically provide flexible interaction with these structures.

Figure 3. Using an out-of-turn interface in a voice-enabled browser.



(a)



(b)

The user decides not to pursue any of the presented hyperlinks and instead says, "40-44 mpg" out of turn (a). Processing the out-of-turn input results in a reduced set of choices, since some makers do not fit the gas mileage specification (b).

ODP offers three special link types:

shortcuts, links whose target is a page deeper in the hierarchy;
back links to pages at levels higher than that of the current page; and
multiclassification links, which bridge unconnected but seemingly related topics.

These links let users circumvent the rigid nature of the site's static navigation structure. Except for the out-of-turn interface, none of the interfaces described so far are suitable for interaction with an unfaceted hierarchy. To construct an enumerative interface, the designer must enumerate all possible facet orders (which of course requires facets), and in using a generative interface, the user must be able to specify an order of facets (again requiring facets).

Some interfaces force the user to supply values for facets in an order completely predetermined by the hierarchy. Others, such as the out-of-turn interface or enumerative interfaces, give the user carte blanche to communicate inputs in any preferred order. Still other sites, such as Project Vote-Smart (http:// vote-smart.org), provide a combination, either enforcing order at the beginning of the interaction and permitting flexibility at the end or vice versa; or mixing and matching both to create novel interaction experiences.

Sometimes retaining at least partial control over input order is desirable and even necessary. Weather report sites, for example, require a zip code before even beginning user interaction.

EXPOSING AND EXPLORING DEPENDENCIES

In any Web site, individual selections are likely to have several dependencies, and the site should expose and provide ways for its visitors to explore them. At a basic level, progressive drilling-down and retracing paths exposes such relationships. After selecting a car model, for example, the user might notice that the model does not meet the desired mileage and thus backtrack to try other selections. Obviously, designers would like sites to expose dependencies without requiring these backtracking interactions.

In Figure 3, when the user says, "40-44 mpg," the site removes choices that do not meet the gas mileage specification. Such hyperlink pruning on the current page provides immediate visual feedback about dependencies. An interface with this capability obeys or exploits the dependencies implicit in the hierarchy and provides facilities to help expose them to users. Web functional dependencies can range from the simple "every Civic is a Honda" tautologies to less salient but more interesting relationships such as "all cars with a safety rating of 3.5 or higher achieve less than 20 mpg on average" or "the Honda Civic hybrid doesn't come in red."

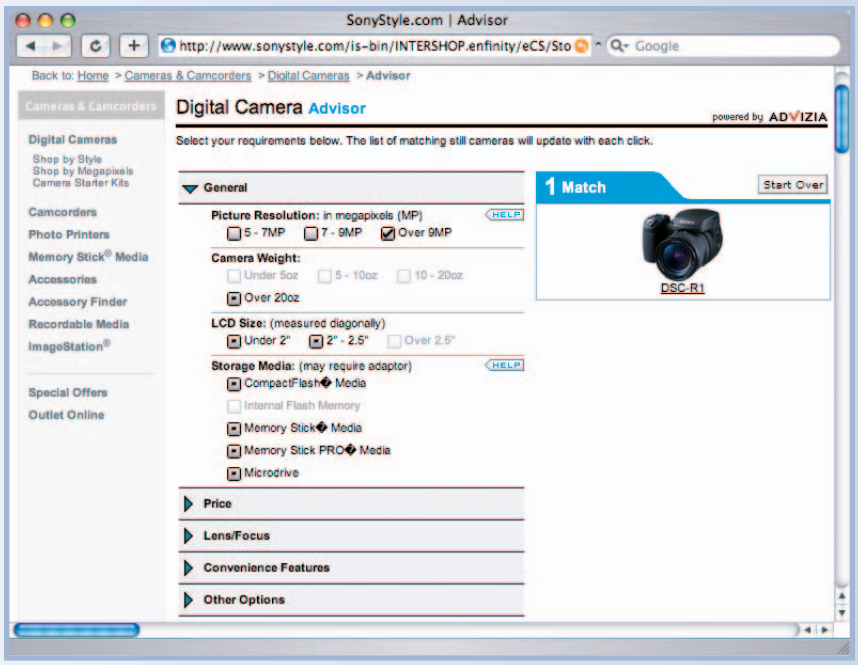
Web functional dependencies can come from either domain knowledge or a data-driven analysis of the hierarchy. For example, if none of the site paths that contain the hyperlinks "Honda" and "Civic hybrid" also contain the hyperlink "red," then the "the Honda Civic hybrid doesn't come in red" dependency holds. Techniques from association-rule mining are relevant in this context. Designers can summarize functional dependencies using Æ to denote implies and a ¬ to denote negation. Thus, for the car shopping scenario, "Honda Æ ¬ Toyota," "Civic Æ Honda," "safety rating > 3.5 Æ mpg < 20," and "Honda, Civic hybrid Æ ¬ red."

Reactive menu designs

Some commercial Web sites now explicitly recognize that dependencies are ubiquitous in hierarchies and provide facilities to exploit them. Kelley Blue Book has a search for make by model name, which invokes Web functional dependencies of the form "model Æ make," thus leading the user to the desired automobile make. The reduction of available options as the user interacts with enumerative designs, such as pull-down menus, is another simple method of exposing depend-

Figure 4. Exposing and exploring dependencies in the Sony Advisor reactive menu.

(a)

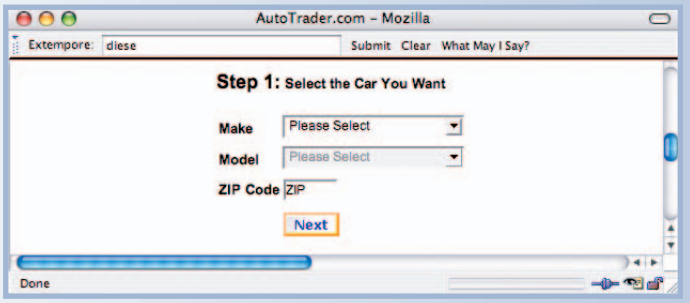


(b)

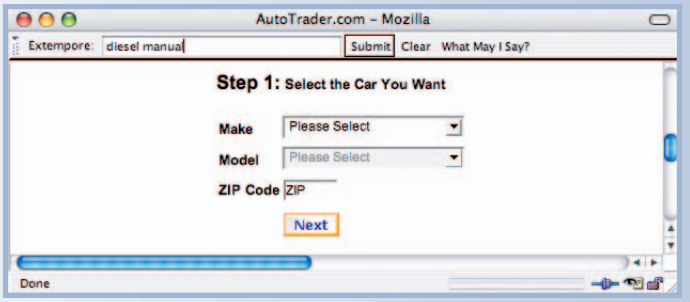
The user sees 14 digital cameras and checks the box specifying cameras with a resolution greater than 9 megapixels (MP) (a). Only one camera meets the criterion—the DSC–R1 model (b). Through this interaction, the site reveals the "MP > 9 Æ DSC–R1" dependency to the user, as well as several others, such as "MP > 9 Æ ¬ DSC–T9."

encies. This approach extends the dependency exposure implicit in out-of-turn interaction's hyperlink pruning. The Sony Advisor reactive menu in Figure 4 (http://sonystyle.com) updates a set of products—digital cameras and camcorders, for example—as users add and remove options, such as picture resolution. The automatic addition and deletion of

Figure 5. Using a real-time query expansion interface in online car shopping.



(a)



(b)

The user types "diese" when the site is soliciting the desired car make and model (a). As soon as the user enters the "l" and thereby completes the specification of "diesel," the site automatically expands the query in real-time to "diesel manual" (b). This reveals the "cars with diesel engines come only with a manual transmission" dependency and hence the user can immediately see that no vehicles with an automatic transmission and diesel engine are available from this site.

products from the current set (right side of screen) based on the status of the check boxes for each facet (left side of screen) exposes the dependencies underlying the products Sony offers, and lets shoppers naturally explore the interplay of the features in the available models. A wine retail site, http://wine.com, has a similar interface for exploring dependencies while browsing wines.

Real-time query expansion

Dependency exploration is also possible through query expansion. For example, when the user says, "safety rating greater than 3.5" in response to the site's solicitation for car make, the site can expand this query to "safety rating greater than 3.5, mpg less than 20" if all cars with a safety rating greater than 3.5 also yield less than 20 mpg. When conducted in real time, such automatic query expansion provides immediate user feedback. Consider another car shopping scenario:

"I am interested in buying a car that runs on diesel fuel, but I've heard that diesel engines are usually available only for cars with a manual transmission. I want an automatic transmission. Are any diesel cars equipped with an automatic transmission?"

A real-time query expansion interface can easily support such a scenario by letting the user know that the "diesel Æ manual" dependency holds. Figure 5 illustrates how this might occur.

Real-time query expansion is user-independent; the expansion is the same for all users and depends only on terms occurring together on the paths through the hierarchy. Prior queries have no bearing on the expansion. The site adds terms to the query only when the composite query yields the same result as the unexpanded query.

Web site interfaces are emerging that use a similar style of real-time query expansion for information exploration and discovery, including Google Suggests (http://www.google.com/webhp?complete=1&hl=en) and Stanford's auto-complete Search on TAP.

The complete set of dependencies that a classification satisfies changes as the user interacts with it. Suppose the user clicks on the Lexus hyperlink in response to a solicitation for car make. This interaction eliminates the "Lexus Æ ¬ BMW" dependency and can cause the "coupe Æ sunroof" dependency to emerge. The "coupe Æ sunroof" dependency indicates that all Lexus coupes have a sunroof and is not likely to exist before the user selects Lexus unless *all* coupes from *all* car makes have a sunroof, which is unlikely. By revealing the all-or-nothing dependencies underlying the values of the classification's facets, real-time query expansion can help users understand the constraints implicit in a domain.

Comparisons

Clearly, there are multiple levels of dependency exploration. Drilling-down a hierarchy through out-of-turn interaction typically prunes the available choices for all facets. However, since this design presents only one facet per page, the user can observe how input affects the set of choices only for the current facet.

Manipulating reactive menus also prunes the choices for all facets. However, in contrast to an out-of-turn interface, this interface has an enumerative design, which lets the user observe the pruning in all facets. Adding real-time query expansion to an out-of-turn interface helps reveal dependencies involving facets beyond the current page.

PROCEDURAL TASK SUPPORT

The interactions we have described so far entail drilling down a classification and thereby reducing the hierarchy's size and number of remaining items. Such interactions are inherently destructive and involve only one line of inquiry, or control flow. In contrast, strategies that are constructive or *procedural* in nature require cascading information across multiple subgoals. Consider another car shopping scenario:

> "I'm interested in a Lexus. I want one whose fuel efficiency is comparable to the Toyota Camry."

This scenario involves two subgoals. First, the user must find the fuel efficiency of the Toyota Camry and then use that information to find a particular Lexus. With a faceted or table-based interface, the user would need to manually remember the information retrieved in the first subgoal, start over with the fully populated instance of the table, and supply the retrieved information to satisfy the second subgoal. The more subgoals, the more complex the process. Interfaces that support procedural tasks provide some way for the user to aggregate retrieved information naturally into a new line of inquiry without having to remember any intermediate results or start over.

One way for the user to cascade information from one subgoal onto another is through a *user-initiated continuation*—an approach that takes its name and motif from the concept of a continuation in programming languages. A continuation indicates a promise to do something and summarizes the work remaining at a particular point in the execution of a program. To cascade one information-finding goal's output to the input of another, the site essentially replaces the current pruned hierarchy (the current continuation) with a fresh copy of the original hierarchy, pruned according to the information retrieved in the previous subgoal. This process provides the user with a smooth transition between subgoals.

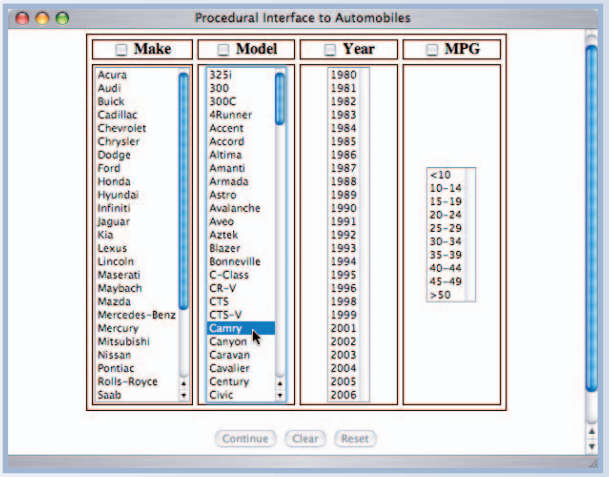
Figure 6 on the next page illustrates how a user might approach the latest car shopping scenario (interested in a Lexus; wants one with fuel efficiency comparable to that of the Toyota Camry) using a table-based continuation interface design. Figures 6a and 6b illustrate one line of inquiry, yielding a result that the interface then helps the user to cascade (in Figure 6c) into another line of inquiry (Figures 6d and 6e). This interaction reveals that the Lexus with fuel efficiency comparable to the Toyota Camry is the ES 330 model.

As this example illustrates, the continuation facility lets the user abandon a given line of conversation (after obtaining the required information) and enter into another line of inquiry carrying the required information to the next subgoal.

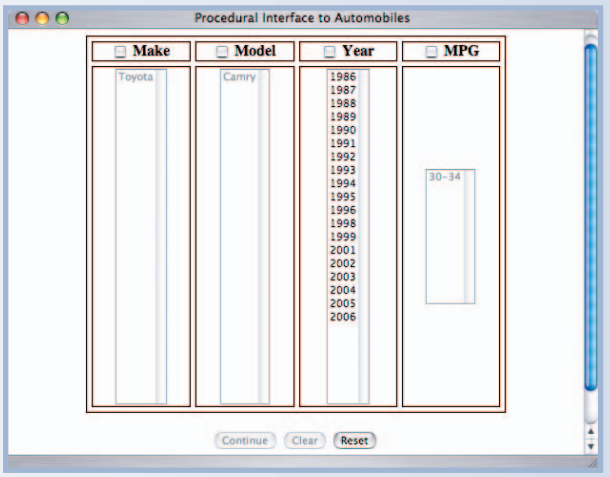
Resources

- **The Web page for the Out-of-turn Interaction Project (http://oot.cps.udayton.edu) contains links to software demos of the user interfaces described in the main text.**
- **"Mining Association Rules Between Sets of Items in Large Databases," R. Agrawal, T. Imielinski, and A.N. Swami, *Proc. ACM Int'l Conf. Management of Data* (SIGMOD 93), ACM Press, 1993, pp. 207-216.**
- **"Strategy Hubs: Next-Generation Domain Portals with Search Procedures," S. K. Bhavnani and colleagues, *Proc. ACM Conf. Human Factors in Computing Systems* (CHI 03), ACM Press, 2003, pp. 393-400.**
- **"Generating Mixed-Initiative Hypertexts: A Reactive Approach," B. De Carolis, *Proc. 4th Int'l Conf. Intelligent User Interfaces* (IUI 99), ACM Press, 1999, pp. 71-78.**
- ***Essentials of Programming Languages*, 2nd ed., D.P. Friedman, M. Wand, and C.T. Haynes, MIT Press, 2001.**
- **"Next Generation Web Search: Setting Our Sites," M.A. Hearst, *IEEE Data Engineering Bulletin*, Sept. 2000, pp. 38-48.**
- **"Finding the Flow in Web Site Search," M.A. Hearst and colleagues, *Comm. ACM*, Sept. 2002, pp. 42-49.**
- **"Keeping Found Things Found on the Web," W. Jones, H. Bruce, and S. Dumais, *Proc. 10th ACM Int'l Conf. Information and Knowledge Management* (CIKM 01), ACM Press, 2001, pp. 119-126.**
- **"Staging Transformations for Multimodal Web Interaction Management," M. Narayan and colleagues, *Proc. 13th ACM Int'l World Wide Web Conf.* (WWW 04), ACM Press, 2004, pp. 212-223.**
- **"Building Rich Web Applications with Ajax," L.D. Paulson, *Computer*, Nov. 2005, pp. 14-17.**
- **"Personalizing Web Sites with Mixed-Initiative Interaction," S. Perugini and N. Ramakrishnan, *IT Professional*, Mar.-Apr. 2003, pp. 9-15.**
- **"User Interface Continuations," D. Quan and colleagues, Proc. 16th Ann. ACM Symp. User Interface Software and Technology (UIST 03), ACM Press, 2003, pp. 145-148.**

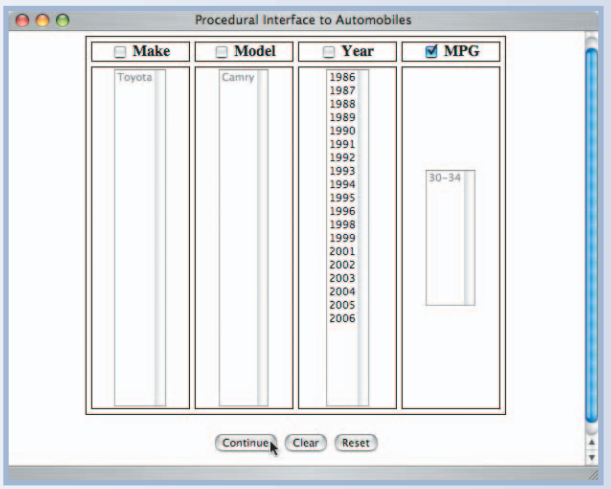
Figure 6. Car shopping through a table-based continuation interface.



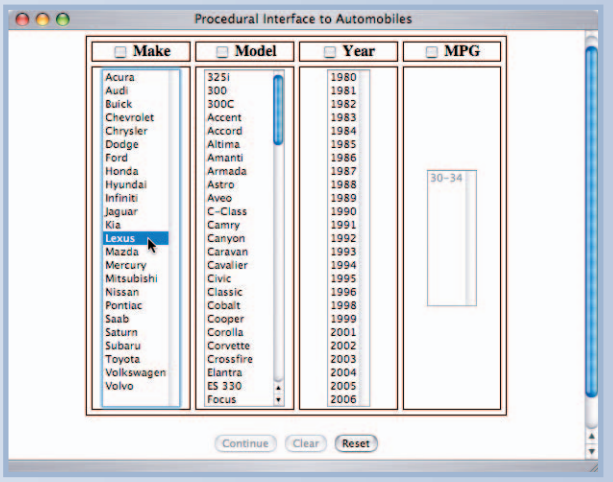
(a)



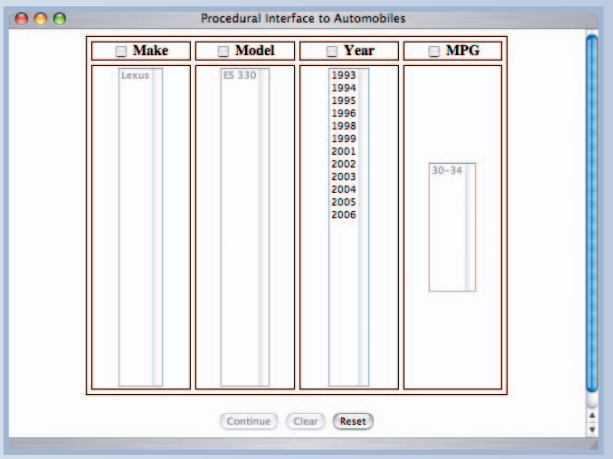
(b)



(c)



(d)



(e)

The user, who is interested in a Lexus that has fuel efficiency comparable to that of a Toyota Camry, clicks on Camry in the list labeled Model (a). The site prunes several choices from each list and reveals an mpg of 30 to 34 (b). To supply this retrieved information in a new line of inquiry to a fully populated instance of the table, the user checks the MPG box and clicks on the Continue button (c). This yields a table instance with only the makes, models, and years of automobiles that get 30 to 34 mpg, and the user clicks on Lexus to find the model with the desired fuel efficiency (d). The site reveals that the Lexus ES 330 model meets the user's fuel efficiency requirement (e).

Procedural information-seeking and information aggregation are suitable for a variety of tasks, especially information refinding—the process of pursuing information you found once and would like to find again.

Designers can add the continuation feature to a variety of interface designs with the exception of the generative interfaces. A generative interface creates a hierarchy tailored to the user's needs, not one that directly solves an information-seeking task, procedural or otherwise. It is possible, however, to add the continuation feature to the hierarchy that *results* from interacting with a generative interface.

Procedural tasks are common in solving complex constraint satisfaction problems. For example, consider a tourist planning a trip to Europe who must not only develop a carefully staged schedule of events (the train arrives in London at 3:00 pm, rental car will be available at 3:30 pm, and hotel check-in is at 4:00 pm), but also satisfy constraints in the process (the Louvre is closed on Sundays). The ability to automatically cascade output from one information-seeking process into another will be an important ingredient in supporting such activities.

Choosing the best implementation technology is an important part of developing interfaces to information hierarchies.

CHARACTERIZING THE DESIGN SPACE

As Figure 7 on the next page shows, the three features we have presented form three dimensions along which interface designs lie. Combinations of the three features form the cube's eight corners; however, only six corners contain examples. There are no examples at the intersection of

- rigid navigation orders, unexposed dependencies, and supported procedural tasks or
- rigid navigation orders, exposed dependencies, and supported procedural tasks

because these are not a good fit for interacting with Web hierarchy.

The figure also reveals interesting relationships. For example, exposing dependencies requires flexible navigation orders, but the reverse is not so. The use of enumeration and the capability to supply input out-of-turn are mutually exclusive. If the current Web page enumerates all possible choices for all possible unspecified facets, the user has no need to interact out of turn. Likewise, the use of enumeration and generation are mutually exclusive. Exposing and exploring dependencies comes free through the pruning of choices for facets in both enumerative and out-of-turn designs, but an enumerative design reveals more dependencies at once, since it presents all choices for all (unspecified) facets on one page. Thus, augmenting an out of turn interface with real-time query expansion can make more dependencies salient on a single page. Finally, adding the continuation facility to all but generative interfaces provides support for procedural tasks.

IMPLEMENTATION ISSUES

Choosing the best implementation technology is an important part of developing interfaces to information hierarchies. The main issue is how to distinguish the sophistication built into the interface from the back-end computations (databases, dynamic Web page generation, and so on).

If the user's interaction with the site is intended to communicate inputs and receive Web pages in return, implementers must assess how the site represents and captures inputs, transforms pages, and determines who is responsible for maintaining the state of the interaction. If there is no need to support out-of-turn input, for example, designs can use hyperlinks or pull-down menus at the browser level, possibly augmenting static pages with JavaScript code. Implementing such a design requires no costly lookup, mapping, or transformation operations. Out-of-turn input, on the other hand, requires some facility to map the user's unresponsive inputs to hyperlink labels (tags) and, more important, to transform the site to accommodate out-of-turn input.

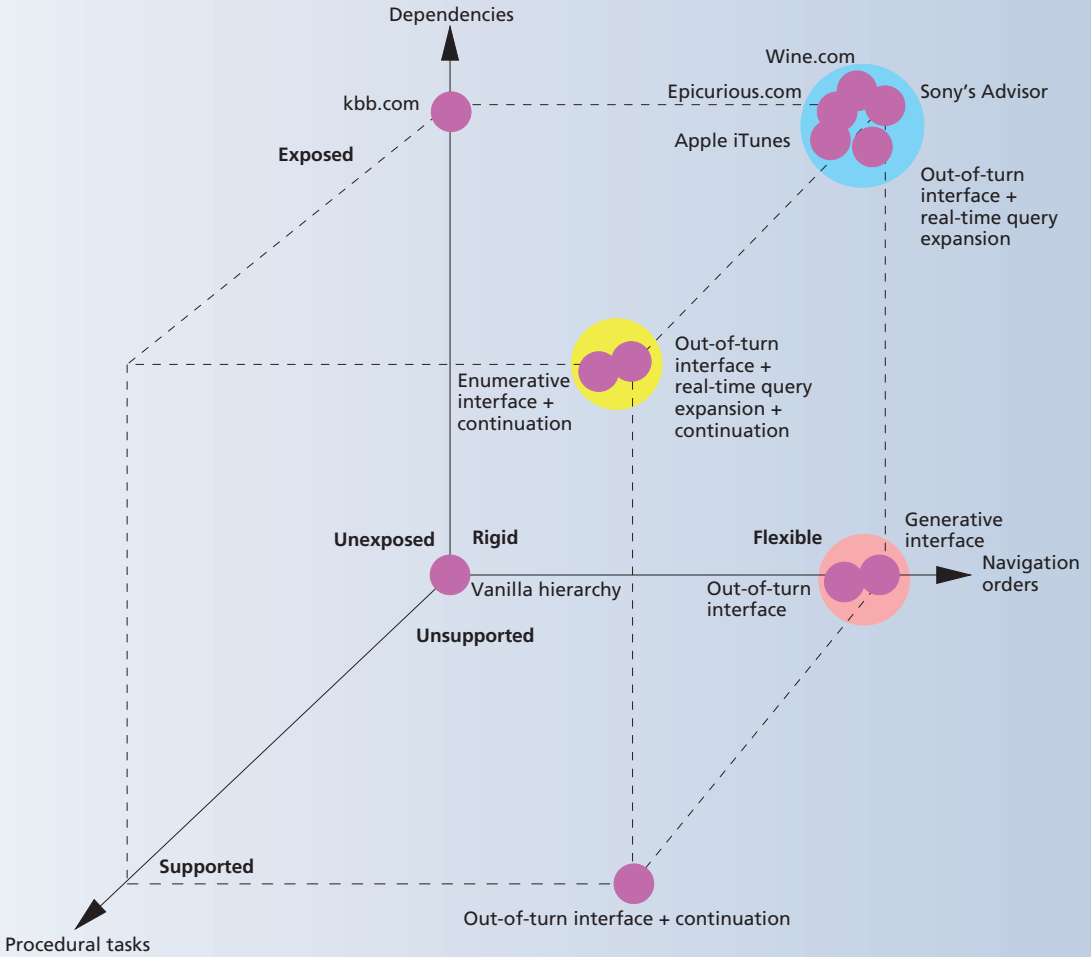
XUL (XML User Interface Language, http://xulplanet.com) supports the construction of cross-platform browser toolbar plug-ins, to capture and communicate out-of-turn input, for use with the Mozilla Firefox Web browser. SALT (Speech Application Language Tags) and X+V (XHTML+Voice) provide suitable support for voice interaction. XSLT (Extensible Stylesheet Language Transformations) is helpful for Web site transformation.

New Web technologies such as Ajax (Asynchronous JavaScript and XML) support more effective communication between front- and back-end components. Ajax is particularly helpful in handling any asynchronous user communication, a requirement for rich and responsive Web interaction. For this reason, Ajax is ideal for implementing real-time query expansion and other within-page designs, such as those in table-based interfaces. Ajax facilities are crucial to the draggable maps in Google Maps (http://maps.google.com), for example. Ajax is also suitable for developing the mapping module while minimizing browser-server communication. In addition, toolkits such as the PLT Scheme servlet API (http://www.plt-scheme.org), which is based on first-class closures and continuations, provide a sound approach for realizing stateful Web interactions, especially those required for the continuation feature.

NEXT-GENERATION OPTIONS

New designs will undoubtedly emerge as the user interface design community better understands users' goals.

Figure 7. Characterizing the design space of interfaces to hierarchical Web sites.



Three key features—flexible navigation orders, exposing and exploring dependencies, and procedural task support—form the cube's three dimensions. At all but two corners, are examples of interfaces, some of which are existing Web sites.

Supporting what-if analysis, for example, requires additional operations such as union and intersection over intermediate results, which might be stored in a scratchpad akin to the online shopping cart. Another practical interaction feature is one that would let the user preview a page before clicking on the link or button that retrieves it. This "I just want to see what's behind the current page" capability is important in scenarios where users are unsure if clicking the submit button, for example, will place a charge on their credit card. Such novel features will provide the next generation of interface options to support compelling interactions with Web hierarchies. ■

Saverio Perugini *is an assistant professor of computer science at the University of Dayton. Contact him at saverio@udayton.edu.*

Naren Ramakrishnan *is an associate professor of computer science and faculty fellow at Virginia Tech. Contact him at naren@cs.vt.edu.*

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.