Rich and Dynamic Library Catalogs: A Case Study of Online Search Interfaces

Jesse Prabawa Gozali
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
jessepra@comp.nus.edu.sg

Min-Yen Kan
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
Tel:65-6516-1885
kanmy@comp.nus.edu.sg

ABSTRACT
We redesign the user interface of an online library catalog, leveraging current web technologies that allow dynamic and fine-grained user interaction. Over the course of our iterative design and test cycle, we identified four key areas where such dynamic web technologies can be used to improve the support for typical information seeking strategies: namely, 1) the use of overview + details, 2) a tabular data display, 3) using tabs as a history mechanism and 4) embedding a suggestion bar. We believe that the revised affordances created by our changes in these four areas will inform the design of future search interfaces.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: OPAC, Human-Computer Interaction, GUI, AJAX

INTRODUCTION
In the past decade, many integrated library systems have migrated their patron library catalog to web-based applications. The resulting online public access catalogs (OPACs) embraced the advantages and limitations of the HTML medium. With HTML 2.0, vendors had access to a simple and uniform user interface toolbox that was limited to clicking hyperlinks and submitting fill-in forms for receiving user input and sending an entire HTML page to report relevant results. While the resulting systems were instantly accessible worldwide, they were static, having limited or no interaction with the user after page rendering.

In the past few years, a set of new technologies have changed this interaction pattern. These technologies push the computational load of handling the interaction to the client's web browser. Dynamic HTML (DHTML), the Document Object Model (DOM) and JavaScript enable the client to respond to fine-grained user interactions with a web page to update and redraw parts of the page. Asynchronous XML + JavaScript (AJAX) extends this notion of interactivity further, allowing the client to both pull data from and push data to the server. Similarly, CSS and XSLT offload the burden of rendering a logical document to the client, easing the transmission load. When coupled together appropriately, these "dynamic" web technologies enable web-based applications to interact with the user in a much more immediate and fine-grained manner than is possible using earlier Web technology. Contrast this to "static" interfaces that use a click-and-wait paradigm and require an entire page refresh to respond to any type of interaction.

How can such technologies improve UI design in online catalogs? Such dynamic web technologies enable a tighter loop of interactivity between the user and the data presentation. We believed that such interactivity could be leveraged to better support the information seeking tasks of library patrons in their catalog use.

In the past year, we have done several iterations of design, implementation and focus group testing, resulting in a new design for our university's catalog system. While our redesign touches on many different usability aspects, we focus on the four areas that are enabled as a direct result of employing dynamic web technologies (Figure 1).

- **Overview + Details Panes**: We replace the separate pages for results and item details with a single page featuring resizable panes.
- **Tabular Data Display**: We replace the listing of results with a sortable, extensible data table that uses AJAX to retrieve additional results without page refresh.
- **Tabs as History Mechanism**: We use tab controls to display both past and current queries. This is a simple, lightweight solution that fits the characteristics of most user catalog sessions.
- **Embedded Suggestion Bar**: We embed a permanent suggestion bar into data display to offer suggestions such as spell check. When patrons choose a suggestion, the new search is executed in a separate tab.

We use these four aspects of our UI as themes to present the main work in our paper, discussing selected related work, our iterative design process and conclusions in each of the next four sections. We then detail our iterative development and summative evaluation process in the Development Process and Evaluation Section, and conclude with a short discussion on our current plans for deployment and future directions in our catalog project.

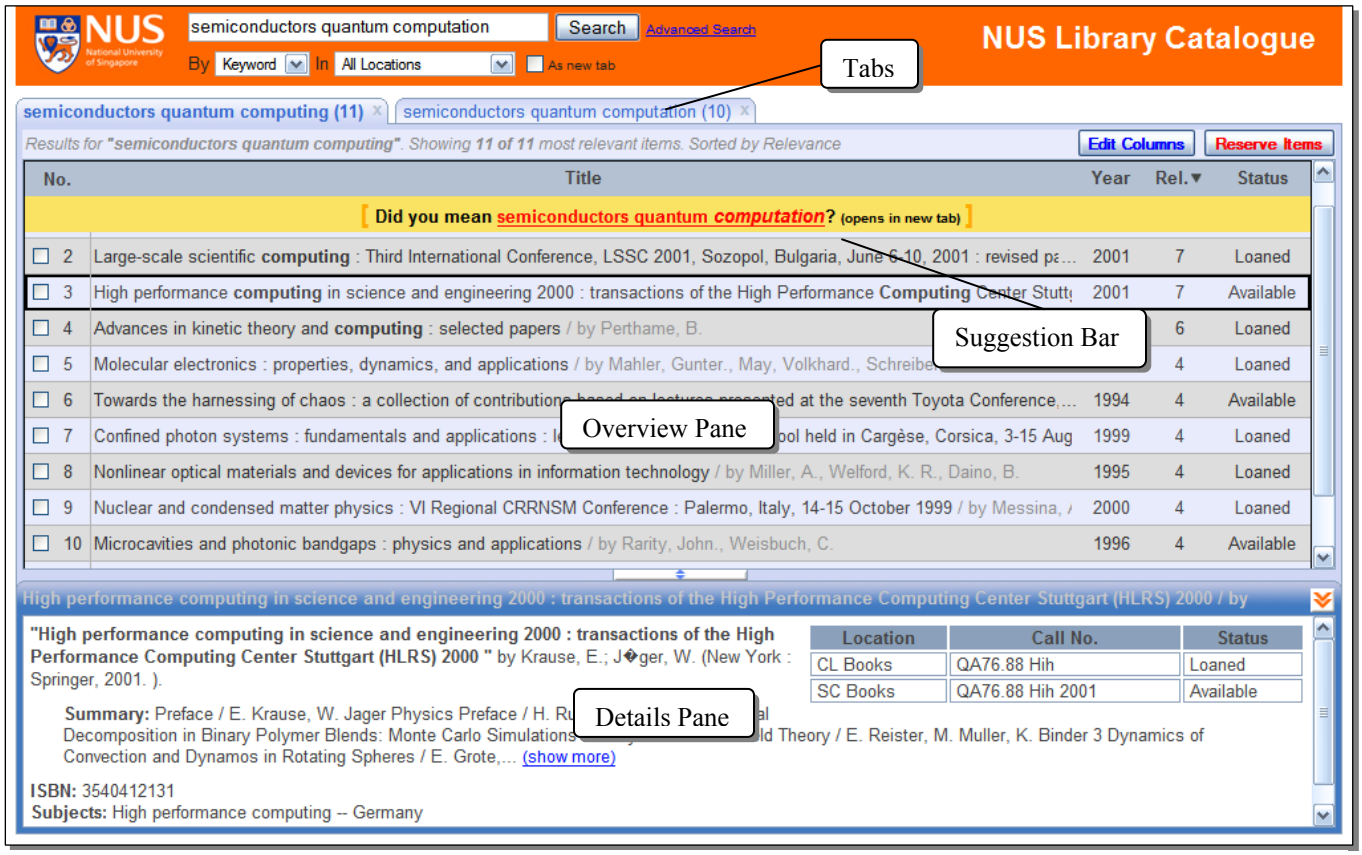


Figure 1: Prototype OPAC User Interface with AJAX.

OVERVIEW + DETAILS PANES

In the typical static OPAC UI, result lists are typically shown in response to a search and the details of an individual item are shown when its record is clicked. However, when users need to compare items in the query results (e.g. during research), such interfaces makes it difficult as direct comparisons is not possible. On the other hand, displaying multiple records at once takes up too much space and presents too much clutter. We believe that in such scenarios, users would benefit from keeping the results list visible while viewing item details, as it would help keep track of their context with respect to their main task [6].

A dual pane UI is one solution showing an overview of items in one pane and displaying the details of a selected item in the other. A survey of such interfaces informed us on our OPAC design.

Our first encounter with a web-based dual pane UI was the Snap web search engine (www.snap.com). This UI allows users to preview thumbnails of the web pages in their search results so that they can assess appropriately before visiting (Figure 2). Snap utilizes a horizontal dual pane layout, taking advantage of normal display aspect ratios to position the site thumbnail optimally. Snap allows the division between the preview and list results pane to be resized but requires both panes to remain visible; neither pane can be completely removed.

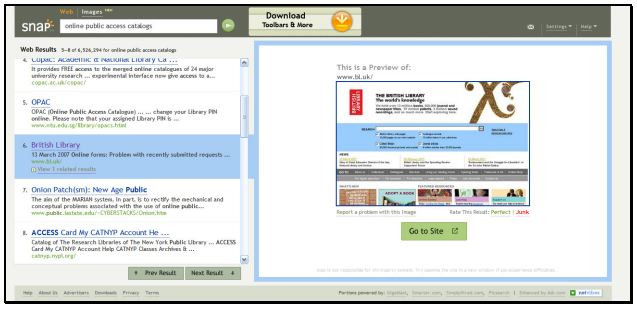


Figure 2: Snap's Preview-based Layout.

We felt such a UI would be too inflexible for catalog users. As a consequence of its horizontal layout, Snap's interface allows only a small amount of items to be visible in the results list, which would be disruptive given large sets of results, which are common. Secondly, certain user tasks may require only one pane, so functionality to minimize or hide one pane is needed. For example, users trying to locate a known item often would prefer to zoom directly to the detailed record.

A more fitting example comes in the form of desktop email clients such as Microsoft Outlook or Mozilla Thunderbird, which features dual panes in a vertical layout. On the web, Outlook Web Access and Yahoo! Mail Beta are current examples, which emulate the UIs of their desktop equivalents. In these UIs, the size of the preview pane can be arbitrarily adjusted depending on user preference. The preview

pane can also be hidden from view such that item details are displayed in a new window instead.

Similar to email users, OPAC users have varied needs and switch information seeking tasks in a single session. Users may initially search for a known item but then move to search for other relevant items (say if the item is checked out). Different from current email client UIs, OPAC users require a dynamic and flexible UI that allows for easy task-oriented customizations. For example, searching for unknown items may rely heavily on previews unlike searching for known ones. Transposing such an interaction model onto the Web would however, be impractical using static web technology. Such a UI would be too slow if the page has to be refreshed every time an item is selected for preview. While using HTML frames may suggest a viable solution, its inherit usability problems detracts most of today's web UIs from using them [9]. Such UIs can only be achieved with dynamic web technologies as they allow for the necessary fine-grained user interactions and page update/redraw, e.g. when an item is selected for preview or when users customize the UI.

Leveraging these technologies, we designed an Overview + Details interaction model for OPACs. Our novelty comes from adapting the advantages of the vertical dual pane model for OPACs in a hybrid approach that allows for other interaction models that do not require item previews. We have also organized item details in a compact layout that maximizes the space it occupies in the Details pane (Figure 3). Here, we display the item summary as a snippet. We believe that this simple strategy suffices. While other strategies like Accordion Summarization [4] work well for page summarizations, OPAC summaries are typically either a brief paragraph or a table of contents.

Similar to the email clients, a vertical layout was adopted as OPAC items have many faceted metadata to be displayed. Spanning them horizontally facilitates better comparison. As users' needs vary even within sessions, the panes must be easily resizable: this is done in the interface by dragging the partition of the panes. We have also introduced a resize handle at the center of the partition to promote the discoverability of this feature (Figure 4).

On the top right hand corner of the Details pane, we have incorporated a button for minimizing or restoring the Details pane. This allows users to quickly switch between dual and single pane modes when their information seeking tasks change. When the Details pane is visible, the button displays as a double downward arrow (Figure 5): clicking on the button minimizes the pane. When minimized, the button changes its display to a double upward arrow: clicking it restores the pane to its prior size. When the Details pane is minimized and the user clicks on an item in the results table, the Details pane is restored to show the item details.

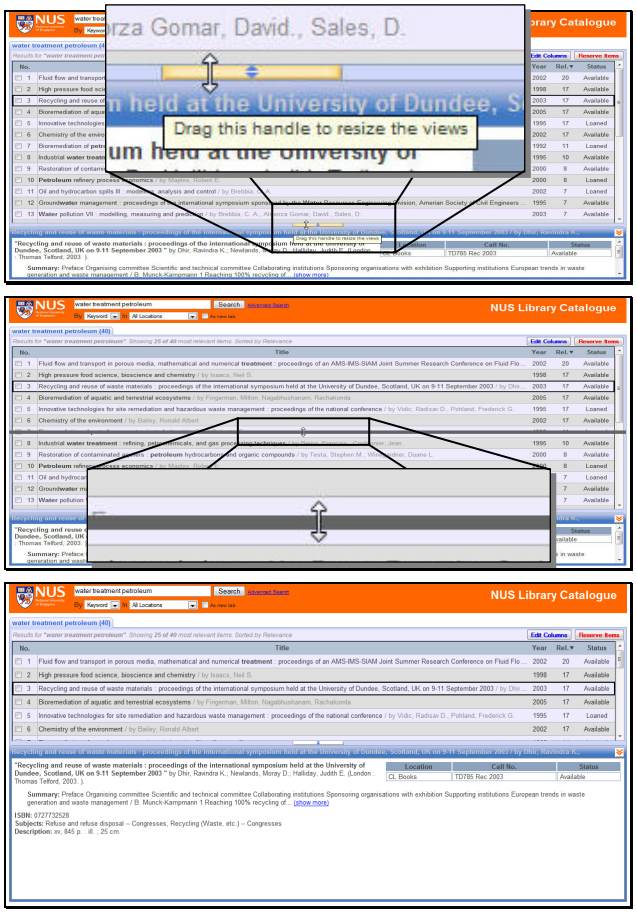


Figure 4: Hovering over the resize handle changes the cursor icon and brings up a tooltip (top). As the handle or partition is dragged to a new position, a grey bar follows the cursor, previewing the new position of the partition (middle). Releasing the mouse button will resize the panes (bottom).

Through our testing (discussed later), we learned that users prefer having both the Overview and Details panes visible at the start. Subsequently, users may want to customize the

Figure 3: Compact item details and the show more link.

UI in different ways depending on the nature of the task.

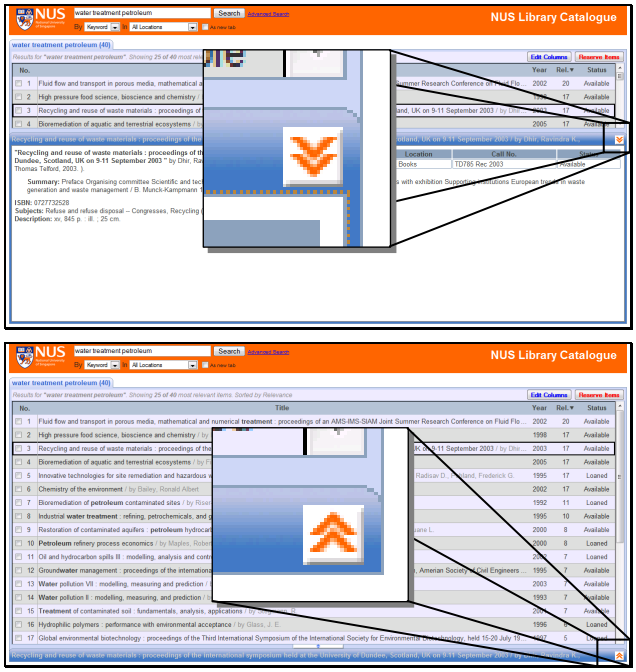


Figure 5: The minimize button (top). The restore button (bottom).

How does this Overview + Details UI assist users in typical OPAC information seeking strategies? We discuss several use case scenarios.

Unknown-Item Searches

When users are unsure of what items are relevant, e.g. for research or casual browsing, users may want to have both panes visible. With both panes visible, users can then evaluate and compare multiple items directly on the same page without having to go back and forth between the results list and item details page.

Known-Item Searches

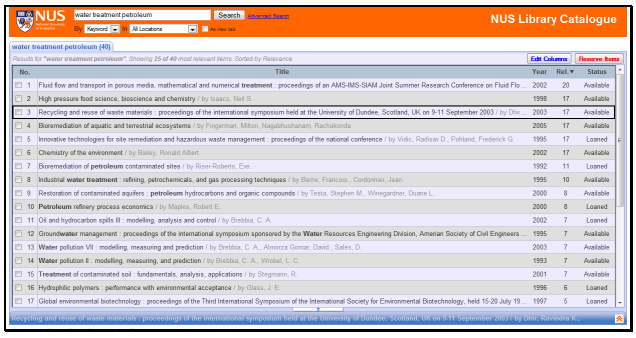


Figure 6: The UI with the Details pane minimized.

When users wish to retrieve a known item (e.g. a title given by a course instructor), recognition can often be done from just viewing the results list. Minimizing the Details pane would reduce clutter and also enable more items to be visible at a time. Figure 6 shows how 17 items in the Overview pane are visible. Also, known item searches typically return a small set of results. For searches that return exactly one result (the title of interest), the UI can maximize the item in the Details pane.

Other Information Seeking Strategies

Other than known and unknown–item searches, the Overview + Details interaction model also allows easy UI customization for more dynamic search behaviors. Users may minimize the Details pane and start with a known-item search. Upon finding the item or otherwise, users may then change their search behavior and direction in an exploratory way to browse for related items with the Details pane visible. Users may also resize the Details pane to a maximum when the item's full description, e.g. table of contents, is important in assessing appropriateness. At maximum size, users can view lengthy descriptions more comfortably (Figure 7). In this mode, the UI also behaves much like a static OPAC UI where the item details completely replaces the results list. When the user clicks on the minimize button, the results list completely replaces the item details.

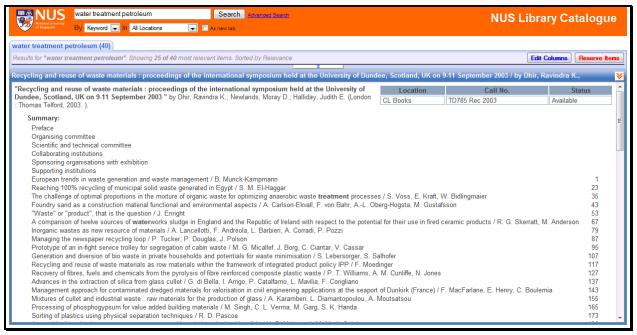


Figure 7: The Details pane maximized.

TABULAR DATA DISPLAY

Coupled with the Overview + Details interaction model is the query results list that occupies the Overview pane. Static OPAC UIs typically list the results in HTML tables that are static and non-interactive. On the other hand, desktop applications (such as Microsoft Outlook or Windows Explorer) feature tables that have more affordances allowing users to sort, add, remove or resize columns. Such affordances may benefit OPAC users as they are important for aiding search strategies, as recommended in [8].

On the web, table/columnar customizations are not supported by many UIs. The advanced web email clients discussed previously only provide limited support; Yahoo! Mail Beta supports column sorting and resizing while Outlook Web Access only supports column sorting. These affordances can only be effectively brought from the desktop to the web by employing dynamic web technologies. With a static web UI, the entire page needs to be refreshed every time a column is sorted, added or removed although these changes only affect the results table.

Library catalog items have many metadata fields posing another concern. A UI that leverages this to its advantage is Flamenco [7]. It uses unique visual representations of its items (architectural objects or other art objects) to support

an interplay of searching and browsing, i.e. users may switch from search to browsing midway allowing for more dynamic search behaviors. Flamenco displays its items in a grid layout with the item thumbnails as its main representation for each item. Although OPAC metadata fits a similar profile, we believe a grid layout does not work well as item thumbnails (perhaps as book covers) may not be the best representation as they recognize items by title/author more than by thumbnails [1]. Such information is arguably more suited for a layout as multiple rows rather than multiple rectangular cells.

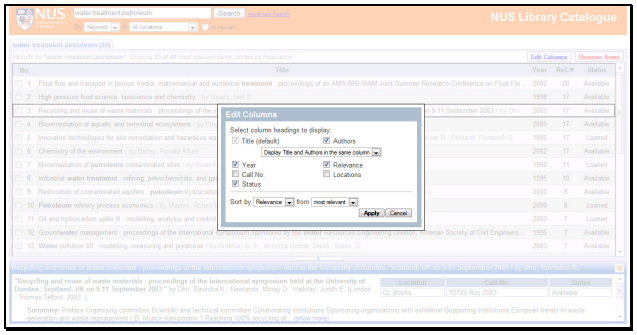
Using a tabular layout with current web technologies, we have thus designed an advanced tabular data display that affords sorting and adding/removing columns. By default, results are fetched by relevance but may be sorted according to other fields by clicking on the column headings: Title, Call No., Year, and Status. A triangle icon accompanies the column heading of the current sort to enhance its affordance as being sortable by clicking. The column heading will also highlight with an orange bottom border whenever the cursor hovers over it (Figure 8). This is consistent the Windows Explorer UI.

Figure 8: The column heading "Year" shows an orange bottom border, showing that it is currently in focus and may be clicked.

However, not all fields may be sorted, e.g. items may have multiple locations or authors. As these fields may have multiple values, sorting such columns poses a problem. While there may be many solutions, for our UI we sort such columns according to the first value, e.g. sorting by locations sorts the items by the first location, sorting by authors sorts by the first author's surname. We have adopted this simple solution because it introduces minimal complexity to the UI.

Column additions or removals are done with a modal dialogue box invoked by an "Edit Columns" button (Figure 8). To promote discoverability, we have placed it on the right side of the UI, just above the column headings. Clicking on the button brings up the Edit Columns dialogue box as shown in Figure 9. Here, users may also change the sorting order while customizing the displayed columns.

The Edit Columns dialog box also gives users the option of displaying both title and authors in the same column (Figure 9). This compound column enables a more compact layout without introducing UI clutter (Figure 10). It is also the default display method used in the original static version of the target OPAC that we are replacing. Also, only a small minority of users sort by author names (often not a field available to sort by in many OPACs, in contrast to date relevance and title) [5] For these reasons, this compound column is enabled by default; separate author and title columns can be produced using the dialogue box.



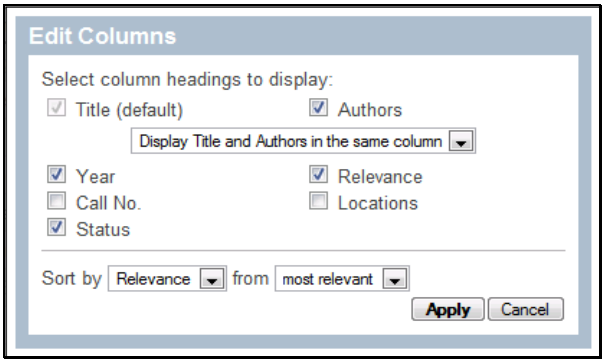


Figure 9: The Edit Columns dialog box with the rest of the UI appearing to be frozen (top). The Edit columns dialogue box with default settings (bottom).

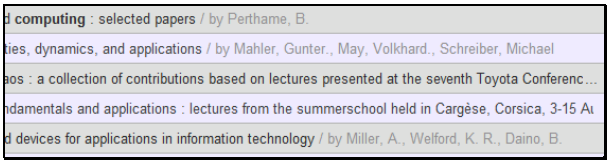


Figure 10: Title and Authors are displayed as a single column by default.

Our UI enables sorting by clicking on column headings, just as desktop UIs do. However, we did not enable adding/removing of columns by right-clicking column headings. Doing so in a desktop environment where users expect the UI to respond to double-clicks, right-clicks, drag-and-drops is understandable. However, static web UIs use single-clicks for interaction. We feel that enabling such desktop interaction pattern makes such behavior difficult to discover. As most web applications are single-click driven, we do not believe users will expect such desktop interaction modes to exist on the web.

While the UI allows for sorting and adding/removing columns, it deliberately disallows column resizing. As such, fields whose contents are longer than the column width will be cut off and appended with an ellipsis, i.e. "…" (Figure 11). To view the full contents, users would have to select the corresponding item for preview. We adopt this simple solution as opposed to using tooltips or resizable columns as they add too much clutter, complexity and unnecessary customizations to the UI.

Figure 11: Fields with contents longer than column width are indicated by an ellipsis.

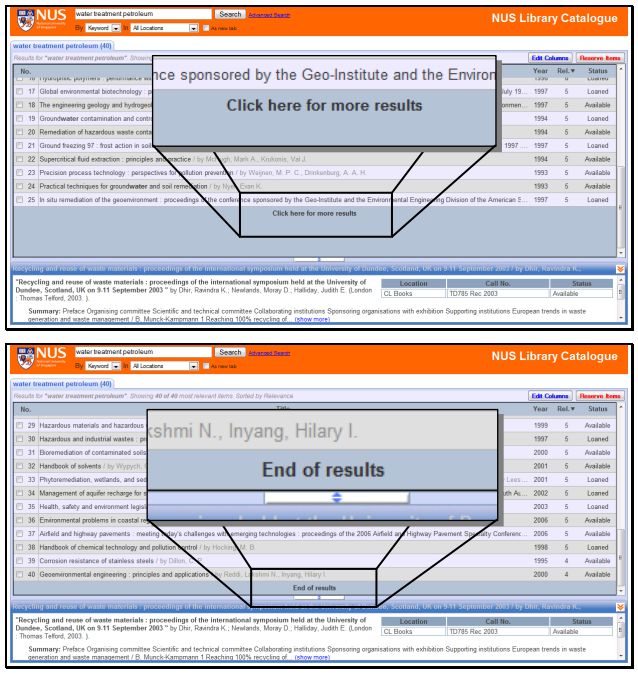


Figure 12: AJAX table with more results (top). AJAX table with no more results (bottom).

As the Overview pane can be resized, scroll bars need to be introduced when entries in the results list are obscured. As scroll bars are necessary to work with the dual pane, we do away with result list pagination although to simplify the locus of control. We see similar design considerations in other web UIs. The absence of pagination also avoids unnecessary navigation between pages, i.e. without a clear over-view of results that are already viewed [11]. Our UI uses AJAX to asynchronously fetch and append to the table, 25 at a time by clicking on the "Click here for more results" link at the end of the table as shown in Figure 12. This is in contrast to the "End of results" row seen at the end of the table when no more results are available.

The query information bar which is located directly above the query results table also provides another indication that only partial results have been fetched. The bar displays the full query string, the total number of results as well as current sorting order (Figure 13).

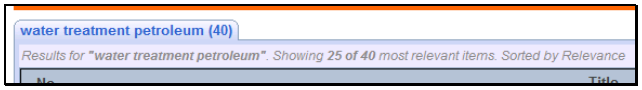


Figure 13: Query Information line displaying the full query string, shown and total number of results, as well as sorting order.

With a table that fetches partial results asynchronously, our UI faces a problem when sorting. In static OPACs, sorting results by some field translates to submitting the same query to the server but have the results returned in a different order. Sorting in a desktop environment, however, translates to sorting items that are in view and displaying them in a different order. For UIs such as email clients where all the available items are in view, there is no dilemma. For our UI, however, the set of displayed items are only a subset of the results. Two options present themselves: 1) we can sort the entire set of results or 2) sort only the items that have already been fetched. We believe that users would expect the latter because of how the UI presents itself similar to desktop UI tables.

Thus, our UI differentiates between how items are fetched and how they are sorted. By default, items are fetched according to relevance. Users may change this setting by performing an advanced search. This difference would mean that whenever the sorting order is different from the fetching order, the displayed results would be out of order every time more results are fetched (Figure 14). Whenever this occurs, the UI prompts the user to resort the table.

Figure 14: The UI needs to be resorted.

We believe that the customizations that we have provided for the tabular data display may support OPAC users in a variety of tasks. Again, we examine some use case scenarios.

Unknown-Item Searches

When users are searching or browsing for unknown items, users may want to customize the displayed columns. For example, it has been shown that humanities and science scholars rely on different attributes to determine the appropriateness of items [2]. Users who browse may also want to use the keyboard to navigate the results. With the Overview pane in focus (Figure 15), the up and down arrow keys navigate up and down the list. Pressing the Enter key will open up the Details pane if it was initially minimized. Pressing on the tab key changes the focus to the Details

pane (Figure 15). Pressing the Enter key here expands the Summary snippet to full text.

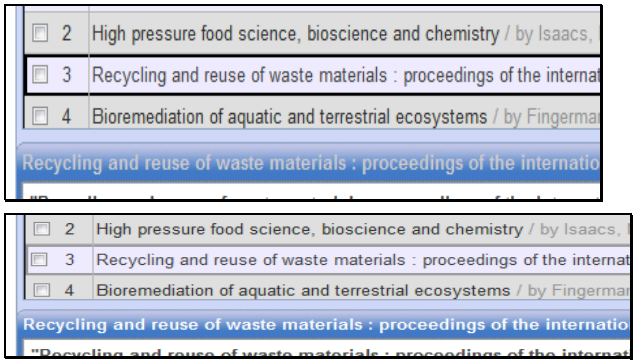


Figure 15: The Overview pane is in focus (top). The Details pane is in focus (bottom).

Known-Item Searches

When users are looking for known items, they may want to easily access the attributes that they recognize, e.g. if the user is looking based on known author names, the user may want to display the title and authors in separate columns. If the year of publication is known, the user may find it helpful to sort the results according to year.

TABS AS HISTORY MECHANISM

Online interfaces with static architectures normally do not implement history mechanisms as the browser's mechanism suffices. Some interfaces, however require more explicit mechanisms to make the interface more usable to users. Many interfaces have adopted breadcrumbs to keep track of history. Breadcrumbs however, work well for hierarchical structures or when trails are long. In contrast, data logs from our local OPAC shows that a typical session lasts for only two user turns.

Static OPACs typically implement their history mechanisms as a drop down list of past queries (e.g., linc.nus.edu.sg). While users can switch to and fro between any past queries, the drop down list does not allow the user to readily view all past queries at the same time. This affordance is important as searches within a single session typically search for items in the same context, e.g. due to query refinements. The Aqua Browser from Medialab Solutions (aqua.elibraryhub.com) on the other hand, is an OPAC that displays all queries entered in a session as a single horizontal list with arrows on either side to scroll when the list becomes too long. Unlike a simple drop down list, this allows users to see several past queries in the same view. Unfortunately, the list behaves much like a history panel as clicking on any past query will erase all future queries in the timeline. In unknown-item searches where users may not know exactly what they are looking for, switching back and forth between query versions may be essential to aid comparisons.

Amazon's A9 search engine (www.a9.com) allows users to search the same search terms using multiple search engines and quickly switch back and forth between all the results. This is achieved by displaying all the results on the same page with each in its own column. The columns displayed can also be easily added and removed using a checkbox list. While A9's multi-column layout aids direct comparisons between results, we feel that it adds too much clutter to the UI and places too much focus on interface navigation rather than the task at hand.

Our UI uses a tabbed interface as a history mechanism as well as to aid such comparisons without introducing much cognitive bandwidth, i.e. although not all results can be seen at the same time, the response time to switch between tabs is fast. Tabs also only require very little screen real-estate. Dedicating too much space to handle past queries would be a waste of real-estate given query histories are short.

Tabs have been used to group and organize workflow in a variety of systems, e.g. web-browsers, office applications. However, no OPAC has ever integrated a tabbed interface for organizing search queries to our knowledge. Our tabbed interface provides users with a clear overview of all submitted queries in the current session. As the information seeking process may involve several stages of query expansion and rephrasing, tabs provide an effective way to organize these queries transparently in the UI, without the clutter of multiple web-browser windows. Tabs also allow users to quickly switch to past queries without having to rely on the web-browser history mechanism which was not previously practical given static web technology. Using dynamic web technologies allow the UI to cache and prefetch the query results.

Figure 16 shows our tabbed interface with two tabbed queries. When the cursor hovers over a passive tab, its background changes into that of an active tab. Its font, however, only changes when the tab is clicked to make it active. This design maintains a one degree difference between a passive and hovered tab, and between a hovered and active tab.

Figure 16: An active and passive tab (top). An active and hovered tab (bottom).

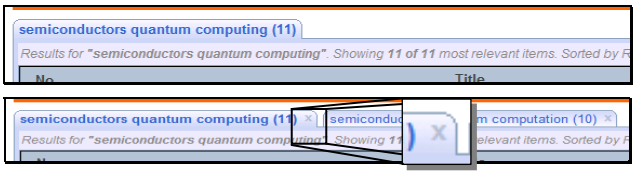


Figure 17: The UI with a single tab and no close tab icon (top). The UI with multiple tabs and a close tab icon available on each tab (bottom).

Another difference in the tabbed UI is between displaying a single or multiple tabs. While the tabs benefit from the close tab icons as a further visual cue we have removed the icon with a single tab to prevent users from accidentally

closing the tab, especially when users have yet to discover the tabbed interface (Figure 17).

EMBEDDED SUGGESTION BAR

A 1998 survey indicated that a major usability problem with OPACs is in finding appropriate keywords for search [10]. However, the more time users spend in thinking of keywords means more time spent focusing on the system and the search itself rather than the task motivating the search in the first place. We believe that spelling suggestions as well as phrasal and morphological expansions can help users find the appropriate keywords, especially for users who have little knowledge in the search domain.

Users are becoming more and more familiar with user assistance modules thanks to web search engines. However, the most popular OPACs in 2005 [3], i.e. systems from Innovative Interfaces, Endeavour Information Systems, and SirsiDynix do not even provide any form of user assistance modules. In fact, not many OPACs have incorporated user assistance modules into their UIs and those who do, only provide spelling suggestions (www.gapines.org).

One UI that provides multiple suggestions for web search is Google Suggest from Google Labs. Unlike typical UIs where the suggestions are provided after the query has been processed and results returned, Google Suggest lists the suggestions as the user types in a query in the search box. While this design allows for different kinds of suggestions to be listed together, we believe that OPAC users may benefit more by seeing the results returned from the intended query before assessing suggestions.

Thus, our UI displays suggestions in a single, uniform suggestion bar embedded inline and floating in the query results table, i.e. it does not scroll along with the table. This placement introduces very little clutter to the UI and is directly visible with its proximity to the results table. In an early iteration of our design, we placed the suggestion bar at the top center of the UI, just above the table column headings and row of tabs (Figure 18). Preliminary testing results, however, show that all of our participants did not even notice it. When asked, users commented that immediately after the page has loaded, they directed their attention directly to the first row of results, effectively missing the suggestion bar which was at a distance separated by the row of column headings and row of tabs.

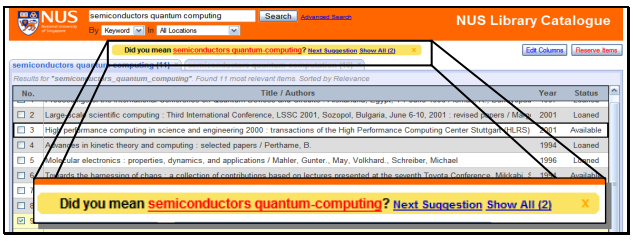


Figure 18: Suggestion Bar unnoticed by users.

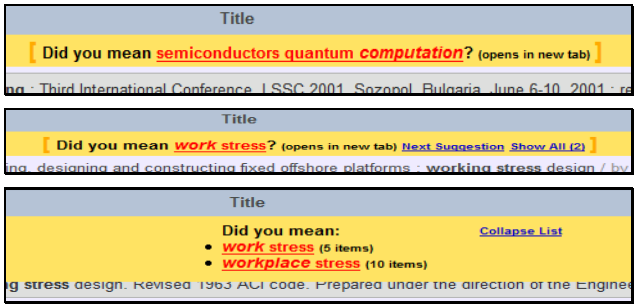


Figure 19: The Suggestion Bar with one suggestion available (top), with more than one suggestion available (middle), and expanded with more than one suggestion available (bottom).

In our current design, when there are suggestions, words in them that are different with the current query will be highlighted with a slightly larger italicized font (Figure 19). When there are no suggestions, the suggestion bar simply does not appear in the UI.

Figure 19 shows the Suggestion Bar in its three possible states: single suggestion, collapsed multiple suggestions, and expanded multiple suggestions respectively. Again these three states show the dynamicity of the bar which has been made possible by dynamic web technologies. When there is a single suggestion, clicking on it will open and load the results in a new tab. Here, we are leveraging the tabs as a display mechanism rather than opening a new window. The suggestion is also accompanied by the text "(opens in new tab)" to promote the discoverability of the tabbed interface. When there are multiple suggestions, clicking "Next Suggestion" cycles though the suggestions one by one, while clicking "Show All" displays the full expanded list. Here, all suggestions are displayed in an unordered list accompanied with their hit counts, i.e. query previews are integrated within the suggestion bar. Clicking "Collapse List" collapses the list back into a single row.

DEVELOPMENT PROCESS AND EVALUATION

In the development process of our UI, several design and test cycles have been executed. Our first prototype was evaluated in an expert review session of 10 people. Results from the review led to the second prototype which was tested on a small scale of 5 users. Subsequently, our UI went through another cycle with the summative evaluation.

To test the UI, we have used slices of relevant results for sample queries taken from our current, static Innovative Interfaces OPAC. A total of 30 undergraduate students aged 20 – 25 years old, ran four query scenarios each to test various usability aspects of the UI features, including discoverability and adoption, e.g. the query "microsoft windows history" invites the user to conduct research for a college term paper. We began each session with an overview of the project and some ground rules to the evaluation before proceeding to give the users the first mock query scenario to work out with the prototype. We conversed freely with the users during the sessions and asked them to think aloud to articulate their actions as they interact with

the UI. The four scenarios took around 20 minutes. This was followed by a five minute interview of the users' thoughts and feedback of the UI. We captured the screens and videotaped the users' interactions for all of the sessions. Figure 20 is a still from one of the video recordings.

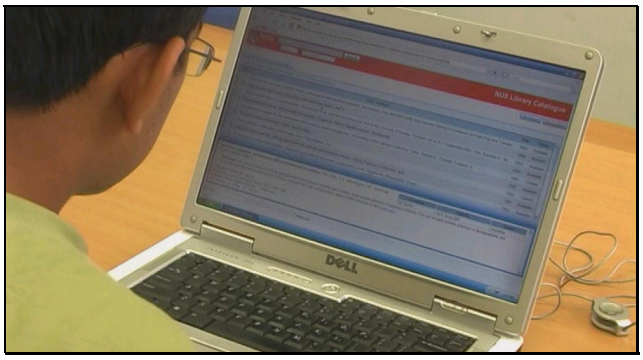


Figure 20: Still from user session video recording.

Our implemented UI and the summative test cases used are accessible from http://opac.comp.nus.edu.sg/. From our evaluation, we were able to see that the waterfall testing method resulted in significant changes in the four areas.

Overview + Details Panes

A static UI would display the details pane only after clicking on an item. On the other hand, our UI uses dual panes. For a smoother transition into the new Overview + Details interaction model, we initially wanted the Overview + Details model to behave like the classic paged model by default and only change to the dual pane model on user customization. This was done by having the Details pane minimized at start and restoring it to maximum size whenever an item is selected. This way the UI can simulate the behavior of static OPACs and only switch to the dual pane model after the user explicitly resized the partition.

To inform the users of the new interaction model, we detected whenever the user repeatedly goes back and forth between item details and results list to trigger a tooltip that launches a tutorial. In our initial testing, however, none of the users noticed the tooltip. We also observed that users actually had no problems in using the new interaction model. The model was just not easily discoverable. In fact, users suggested that the feature should be enabled by default. Thus, instead of repositioning the tooltip, we have enabled the feature for our summative evaluation. Subsequently, we observed that users indeed had no problems using the feature. From the four scenarios, we observed that the adoption rate of the feature was 61% for all four scenarios. However, if we only consider the scenario where research or browsing is involved, the adoption rate is 84%.

Tabular Data Display

Although by default the results are sorted by relevance, we initially did not display this column to simplify the display. From 27 users who were asked to sort the results by some column, 18 still clicked on the column heading even though there were no visual cues in the original prototype designs. 6 used the Edit Columns button, while 3 users had difficulty discovering the feature. When we asked the users to add/remove some column, out of the 28 users, 23 correctly found the Edit Columns button while 5 users tried to bring up a context menu by right-clicking the mouse instead.

We observed that most users had no problems in locating how to sort or edit column headings. The issue was more on discoverability. As such, we modified the UI after the evaluation to include more visual cues. We displayed the Relevance column by default to ensure the triangle icon was visible in the UI. The iterative testing cycle also helped to move the Edit Columns button closer to the column headings. It was initially positioned two rows away.

At first, we did not introduce any difference in design between the "Click here for more results" and the "End of results" rows. Although this feature was not essential to the completion of the scenarios, we observed in many sessions that users tend to scroll down to the bottom of the results list and quickly scroll back up again once the scrollbar stops moving. On introspection, we believe that the caption "Click here for more results" was not visible enough to draw the user's attention in the earlier design rounds. We thus introduced some negative space following the "Click here for more results" text. This effectively shifts the last row of the table further up to draw more user attention to the text.

Tabs as History Mechanism

Users were very satisfied with the tabbed interface as it was easier for them to navigate back and forth between queries while having a good overview over all the queries at the same time. Many identified the tabs in our UI with the tabs in the new Internet Explorer version 7. Only 5 users did not notice the tabbed interface, while 12 users only realized after there were two tabs open. The remaining 11 users had no problems identifying the tabbed UI.

Embedded Suggestion Bar

Users also had no problems identifying the suggestions and seemed very familiar with the idea. Most users identified them with Google's spelling suggestions while others referred to them in various ways: tips, more correct phrasings, more intelligent query versions, similar search topics, better keywords, or as suggestions. The discoverability of the suggestion bar was also acceptable in the final design, after the embedding of the bar directly in the query results table. We designed two query scenarios to specifically evaluate the suggestion bar. Users encounter one suggestion in the first scenario and two suggestions in the second one. Out of 30 users, 12 did not notice the bar in the first encounter. However, only 3 in 2 encounters and only 1 in all 3 encounters failed to notice it.

We initially incorporated a "close" icon in the suggestion bar we tested with the 30 participants. We have since removed the icon as users did not seem to be distracted by the presence of the bar as they carry on their task. None of the users closed the bar in scenarios involving the suggestions. In fact, because our initial design removes a sugges-

tion once the user has clicked it, many users wondered how to bring the suggestion bar back. As such, the UI no longer removes suggestions from the bar even after the user has clicked on it.

In our observation, we have also noticed that the suggestion bar has altered the search behavior of some users, e.g. while some users realize the appearance of the bar just after the page loads or just after the tab loads, those who did not will consciously look for the suggestion bar once they determined the item they are looking for was not found in the results list. This was observed in 15 of the 30 users tested.

Scaffolding Web towards Desktop Paradigm

One of our main concerns in the study was that the new UI is very different from the static OPAC UI. We did not want the UI confuse users who approach it from either the static web or the desktop paradigm. We were afraid the UI might generate too much user expectation from the new interaction model, i.e. users might expect much more functionalities typical of a desktop environment from the UI. However, from the 30 users, only 8 users showed such an indication. Out of the 8 users, one tried to open a new tab by dragging a row from the results table, three tried to resize the column widths while five tried to right click on the UI to bring up a context menu. Thus, the majority of users did not have any problems with the new UI and its scaffolding. We received very positive feedback from users that the new UI is more aesthetically pleasing and that the features are helpful in navigating and assessing the search results. The new UI was very much preferred over the old UI at http://linc.nus.edu.sg, which is an OPAC from Innovative Interfaces, one of the top four most popular OPACs in 2005 [3].

CONCLUSION

We have presented a redesign of the online library catalog using current dynamic web technologies. While our new design touches many disparate aspects of the UI design, we have focused on four areas where the increased interactivity directly benefits typical information seeking tasks of library patrons.

While our redesign greatly benefited from iterative design principles, we have not yet deployed the UI on the full library metadata and have yet to report on a longer-term evaluation and rate of user acceptance. Despite this limitation, our evaluations to date suggest that these will not be a large problem as the redesign has been well-received by testers and that expectations from the legacy UI did not hamper its usability. As such, we believe the design choices made in our catalog redesign will inform the design of other information search interfaces, as to how to best incorporate dynamic web technologies.

FUTURE WORK

Current work focuses on deploying the UI by populating the server with the necessary record metadata to replace the legacy catalog. We are utilizing the open-source Lucene search engine to integrate modern, keyword-based information retrieval technology in our catalog redesign. Our target in the continuation of this project is to integrate other desired features by both library staff and patrons into the catalog (e.g., RSS feeds and aggregators for alerts) [12].

ACKNOWLEDGEMENTS

We thank the student volunteers of the National University of Singapore, especially members of the WING group who have helped with the user testing at many stages of system development. We also would like to acknowledge the cooperation of the NUS Libraries staff for making the OPAC and session logs available to us to study.

REFERENCES

1. Allen, B. Recall Cues in Known-Item Retrieval. J. Amer. Society for Info. Science. 40(4). 1989. 246-252
2. Bates, M. J., Wilde, D. N., & Siegfried,S. An analysis of search terminology used by humanities scholars: The Getty Online Searching Project Report Number 1. Library Quarterly, 63, 1 – 39. 1993
3. Breeding, M. Libraryjournal.com: Tables for Automated System Marketplace 2005. Retrieved March 30, 2007, from http://libraryjournal.com/article/CA512806.html. 2005
4. Buyukkokten, O., Garcia-Molina, H., Paepcke, A. Accordion summarization for end-game browsing on PDAs and cellular phones. *Proc SIGCHI conference on Human factors in computing systems*, 2001, 213-220
5. Dumais, S. T., Cutrell, E., Cadiz, J. J., Jancke, G., Sarin, R., & Robbins, D. C. Stuff I've Seen: A system for personal information retrieval and re-use. *Proc SIGIR'03*, 2003
6. Ellison, M. Secondary windows in online help: What do users really make of them? Usability Interface: Newsletter of the STC Usability SIG. 7, 3. 2001
7. Hearst, M., English, J., Sinha, R., Swearingen, K., and Yee, P. Finding the Flow in Web Site Search. Communications of the AC. 45 (9), 2002, 42-49
8. Lergier, R., & Resnick, M. A framework for evaluating user strategies in internet search and evaluation. 7th Conference on Human Factors and the Web. 2001
9. Nielsen, J. Why frames suck (most of the time). Retrieved March 30, 2007, from http://www.useit.com/alertbox/9612.html. 1996
10. Rousseau, G., Jamieson, B., Rogers, W., Mead S., & Sit, R. Assessing the usability of on-line library systems. Behavioral & Information Technology. 17(5), 1998, 274-281
11. Spool, J. As the page scrolls. UIE eye for design. Retrieved January 18, 2007, from http://www.uie.com/articles/page_scrolling/. 1998
12. West, J. What We Want: An OPAC Manifesto. Retrieved March 30, 2007, from http://www.librarian.net/opac. 2005