Clip, Connect, Clone: Combining Application Elements to Build Custom Interfaces for Information Access

Jun Fujima [1*] *Aran Lunzer* [1] *Kasper Hornbæk* [2] *Yuzuru Tanaka* [1]

[1]Meme Media Laboratory
Hokkaido University
Sapporo 060-8628, Japan
{fujima, aran, tanaka}@meme.hokudai.ac.jp

[2]Natural Sciences ICT Competence Centre
University of Copenhagen
2100 Copenhagen Ø, Denmark
khornbaek@nik.ku.dk

ABSTRACT

Many applications provide a form-like interface for requesting information: the user fills in some fields, submits the form, and the application presents corresponding results. Such a procedure becomes burdensome if (1) the user must submit many different requests, for example in pursuing a trial-and-error search, (2) results from one application are to be used as inputs for another, requiring the user to transfer them by hand, or (3) the user wants to compare results, but only the results from one request can be seen at a time. We describe how users can reduce this burden by creating custom interfaces using three mechanisms: clipping of input and result elements from existing applications to form cells on a spreadsheet; connecting these cells using formulas, thus enabling result transfer between applications; and cloning cells so that multiple requests can be handled side by side. We demonstrate a prototype of these mechanisms, initially specialised for handling Web applications, and show how it lets users build new interfaces to suit their individual needs.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]: Graphical User Interfaces (GUI), Interaction Styles; D.2.12 [**Interoperability**]; H.5.4 [**Hypertext/Hypermedia**]: Navigation, User Issues

Additional Keywords and Phrases: customized information access, end-user programming, parallel exploration

1 INTRODUCTION

This paper describes a prototype that supports end-users in reducing the burden in certain kinds of information access. The prototype lets users create custom interfaces by clipping, connecting, and cloning elements from existing applications. It is currently specialised to the handling of Web applications, yet the underlying mechanisms are also suitable for applications of other kinds.

The motivation for this work arises from three kinds of challenge that are commonly observed in the use of applications for information access.

First, it can be laborious for users to find the interface elements that are of interest. For example, the input fields for a Web application may be on a page that also includes many other items; similarly, the page of results delivered by that application may include details that the user does not need. If the application is used repeatedly, even the effort of locating the relevant input fields and the relevant results can become a burden. We would like users to be able to create simplified versions of the applications they are using.

Second, users often want further information related to the results they have found. This may involve using results from one application as inputs to another. For example, in planning an evening out, a user may invoke one Web application to discover what films are currently showing, choose a film, see which cinemas are showing it, look up the address of a cinema to feed into a transport enquiry, and so on. Repeating all these steps for each of several films would be hard work; it would be preferable if the user could create connections that automatically transfer the data between applications.

Third, the results obtained from one request can have increased value when seen in the context of other results. For example, the film-goer might want to consider a variety of cinemas and screening times, in order to compare the cost and convenience of getting to each one. However, comparing results is difficult if each request causes the previous result to disappear from view. We would like to help users handle alternative requests side by side, so that it becomes easy to submit many requests and to compare their results.

This paper extends an earlier poster presentation [7] in describing C3W (Clip, Connect and Clone for the Web), a prototype that provides integrated support for addressing these three challenges. We show how C3W allows a user to create a custom information-access interface based on one or several Web applications, then use this interface to view and control many alternative requests side by side. Providing support for such side-by-side handling is the central theme of our work on Subjunctive Interfaces [11, 12, 14]; C3W represents a substantial advance in this work, being the first system to make subjunctive-interface facilities available for a whole class of applications that normally handle just one request at a time.

[*] The first author carried out much of the implementation reported here during a visit in 2003 to the University of Copenhagen, where the second author was then working.

UIST '04, October 24–27, 2004, Santa Fe, New Mexico, USA.
Copyright © 2004 ACM 1-58113-957-8/04/0010...$5.00.

Section 2 demonstrates our prototype and explains its capabilities. The facilities needed to generate this kind of system are discussed in Section 3, while Section 4 places our work in the context of prior research. The contributions are summed up in Section 5, where we also discuss how to move beyond the limitations of the existing prototype.

2 EXAMPLES AND EXPLANATIONS

Below we explain the main features of C3W through two examples.

2.1 A Useful Union of Two Applications

Suppose an investor in Japan wants to look up the stock price of US-quoted companies, but to see them in Japanese yen rather than dollars. Perhaps no single Web application offers such a lookup, but the investor knows that the CNN Money site offers a stock-price lookup in dollars, while Yahoo! has a currency conversion page that can certainly deal with dollars and yen. Furthermore, suppose the investor wants to see several companies' prices at the same time.

Figure 1 shows an interface, built using C3W, that fulfils these needs. We will now describe the three main techniques used in building it.

2.1.1 Clipping This interface was assembled from two previously created, simpler interfaces: one for the stock-price lookup and one for the currency conversion. Figure 2 shows how elements were initially clipped from the Yahoo! site; Figure 3 shows these clipped elements arranged by the user on what we call a C3Sheet, along with a second C3Sheet for the CNN lookup. All the cells on these sheets act as portals (in the sense of [18]) onto the Web pages from which they were clipped. Thus the top three cells on the currency-conversion sheet (on the left in Figure 3) retain their roles as input devices, so that specifying a new value in any one will cause a re-evaluation of the Yahoo! conversion and will update the bottom text field to show the new converted amount.

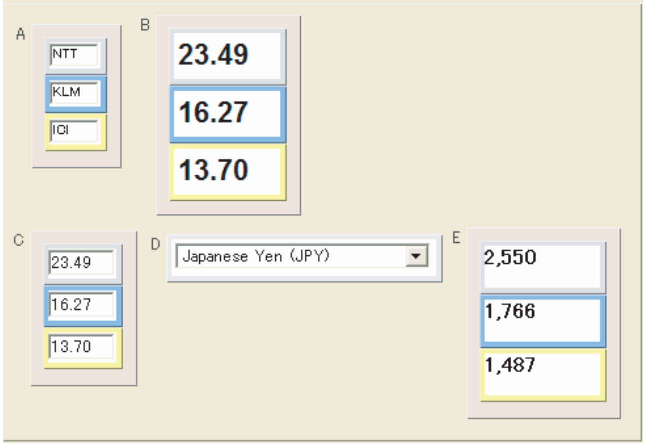
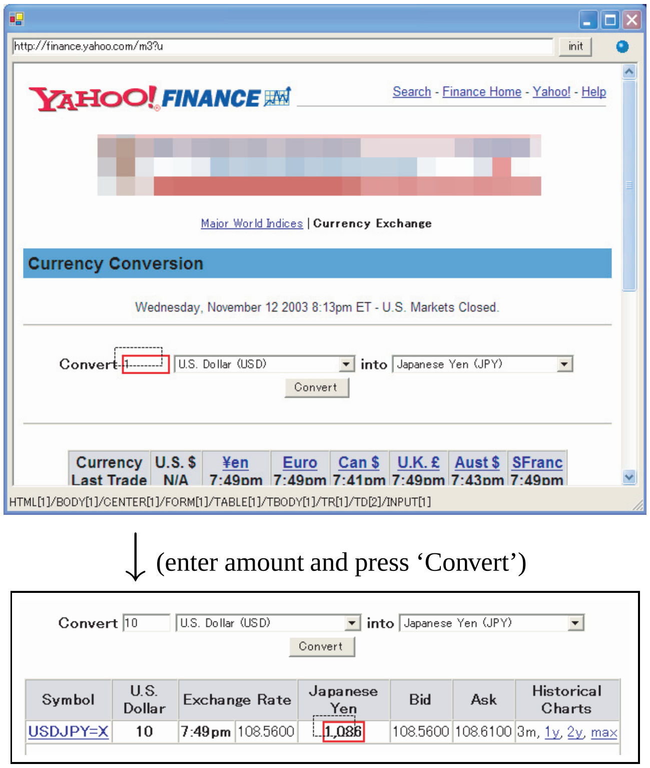


Figure 1: A user-built interface that takes company stock codes as input (in the cells marked A), and finds their dollar prices (B) and the equivalent yen amounts (E). Cells A and B were created by clipping elements from a stock-price lookup application; C, D and E were clipped from a currency-exchange application. The user connected these applications by giving C a formula that copies the value in B. The user has also cloned cell A to deal with three cases in parallel, which has caused the automatic cloning of all cells that depend on A. Each set of clones uses the same layout and colour-coding, to emphasise their correlations – for example, the current dollar price of NTT is $23.49, and the current value of this many dollars is ¥2,550.

Note that the cells need not be clipped in the order in which they are encountered. In particular, if the user had only decided to wrap this application after having obtained some result, he or she could have clipped the result field first, and then gone back to the previous page to clip the parameters that led to it.

Each of the C3Sheets in Figure 3 works as a reusable application, that could be saved on a user's local disk or distributed to other users for their immediate use. This illustrates how Clipping enables the creation of simplified versions of an application.


init
YAHOO! FINANCE
Search - Finance Home - Yahoo! - Help
Major World Indices | Currency Exchange
Convert

http://finance.yahoo.com/m3?u

Currency Conversion

Wednesday, November 12 2003 8:13pm ET - U.S. Markets Closed.

Convert 1 U.S. Dollar (USD) into Japanese Yen (JPY)

Currency	U.S. $	¥en	Euro	Can $	U.K. £	Aust $	SFranc
Last Trade	N/A	7:49pm	7:49pm	7:41pm	7:49pm	7:43pm	7:49pm

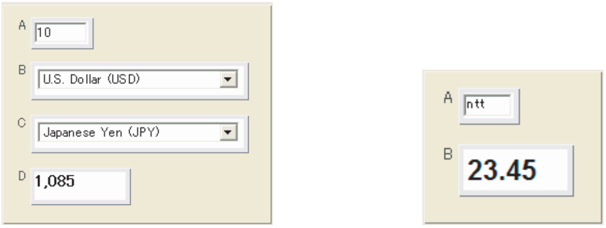
HTML[1]/BODY[1]/CENTER[1]/FORM[1]/TABLE[1]/TBODY[1]/TR[1]/TD[2]/INPUT[1]

↓ (enter amount and press 'Convert')

Convert 10 U.S. Dollar (USD) into Japanese Yen (JPY)

Symbol	U.S. Dollar	Exchange Rate		Japanese Yen	Bid	Ask	Historical Charts
USDJPY=X	10	7:49pm	108.5600	1,086	108.5600	108.6100	3m, 1y, 2y, max

Figure 2: Clipping elements from a currency conversion application. At the top is the default input page for the conversion, on view in C3W's specialised Web browser. Holding down the Alt key brings up a red rectangle that follows the mouse to highlight the region of the nearest surrounding HTML tag; this indicates which portion of the page will be clipped if a mouse drag is started. Here the user has started dragging the conversion input field, to drop it on a C3Sheet. After also clipping the drop-downs that specify the source and destination currencies, the user enters a sample dollar amount and presses the 'Convert' button. The lower part shows the clipping of the converted-amount text from its place on the result page.



A 10

B U.S. Dollar (USD)

C Japanese Yen (JPY)

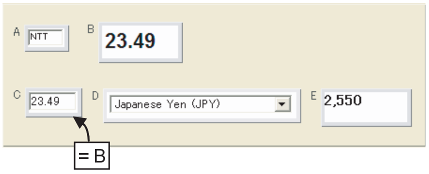
D 1,085

A ntt

B 23.45

Figure 3:
Left: C3Sheet with the elements clipped in Figure 2.
Right: C3Sheet for a stock-price lookup. The cells' different font sizes reflect the visual nature of the clipping operation; these are the fonts used in the elements that were clipped. However, each cell created from a Web-page element is in essence a fully functioning Web browser, so the user is free to adjust the font size.

2.1.2 Connecting The next stage in building the example in Figure 1 is to create a combination in which the two applications are connected. To do this, the user starts with another empty C3Sheet and repeats the clipping process – this time clipping the cells from the existing C3Sheets. Notice that there is no need to clip the source-currency selector if the desired currency is always the same; the user must simply ensure that the selector is set to that currency when the other cells are clipped. The cells may be laid out like this:



A NTT

B 23.49

C 23.49

D Japanese Yen (JPY)

E 2,550

= B

Here the upper two cells (A and B) come from the stock-price application, while the lower three are from the currency conversion. The user has already taken the final step necessary in connecting the applications together: establishing that the conversion-amount cell (C), rather than its previous status as a user input field, is to take its value from cell B. This is done by adding the simple formula '=B' to cell C. Now whenever B's value changes, as the result of a new stock-price lookup, C is automatically updated and a currency conversion is triggered. If the user wanted instead to convert the price of 100 shares, this could be done by making C's formula '=B*100'.

This C3Sheet also works as a reusable application, accepting a stock code in cell A and producing a converted price in E. It could be configured to re-evaluate automatically every few minutes to reflect the latest stock price and conversion rate. In addition, because it includes the destination-currency selector, it will perform conversions into whichever currency the user chooses; for example, a Danish investor might want Danish Krone conversions.

This illustrates how Connecting enables the automated transfer of results from one application to another.

2.1.3 Cloning The third stage in our example is when the user decides to track several companies' stock prices in parallel. The interface shown above can be extended to support this, by cloning the relevant cells.

Using a pop-up menu on cell A, the user asks for an additional scenario, and then one more. This causes A to be cloned twice, with the new instances appearing directly beneath the original and initially holding the same stock code. At the same time, cloning automatically takes place for every cell that depends on the stock code – namely the dollar price (B), the input to the currency conversion (C), and the conversion result (E). Each cell that is cloned shows its instances in the same layout, and with the same coloured borders. Thus when the user types a new stock code into any instance of cell A, it is easy to tell where the results for that stock will appear.

A cell that has not been cloned remains as a single instance, that participates in all scenarios. An example of this is the single instance of the destination-currency selector in Figure 1. Selecting a new currency will cause all scenarios to be re-evaluated – for example, one could switch all the conversions simultaneously to Euros. Alternatively, the user can specify that this cell should also have separate instances for each scenario; then each scenario could be given a distinct setting, for example to convert the NTT price into Yen, the KLM price into Euros, and the ICI price into Pounds.

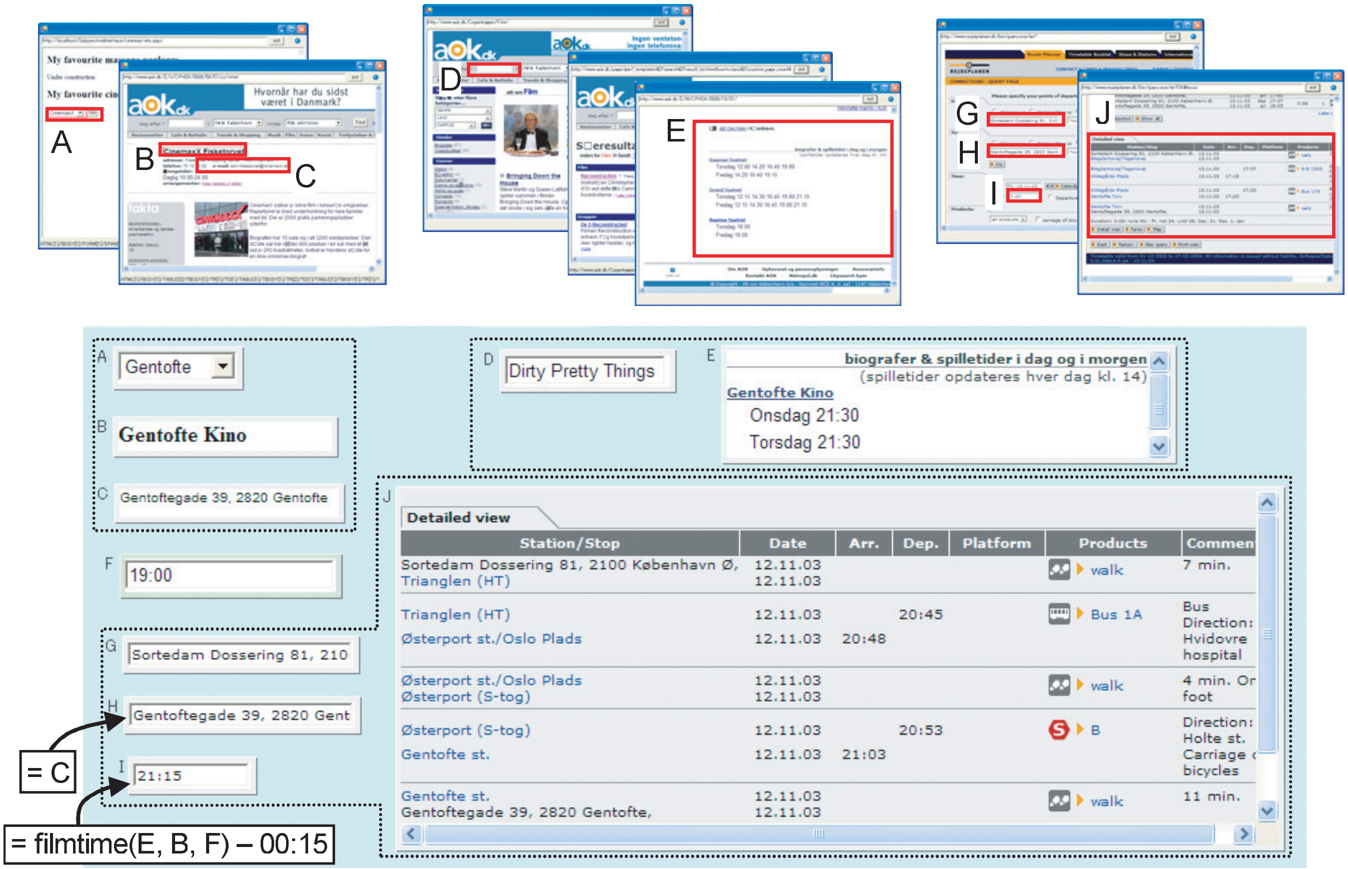
This illustrates how Cloning enables side-by-side handling of requests that normally can only be handled separately.

2.2 A More Ambitious Combination

We now present a more complex example. Consider someone who lives in Copenhagen and likes to make use of the city's many cinemas. It would be useful for this person to have an application that helps in planning a film outing. The C3Sheet shown in Figures 4 and 5 was built by clipping from three Web applications with the following facilities:

1. A personal page that lists favourite places and has links to pages describing them. It includes a drop-down selector listing cinemas by their familiar names; the user can select a cinema then navigate to a description page containing the cinema's official name and address.
2. The film section of a large Copenhagen events site, that includes a search box into which the user can enter the name of a film currently on release and obtain a page with a description of the film and links to further details. One link leads to a page detailing where and at what times the film is being screened.
3. A travel-planning site allows a user to specify start and destination addresses, and the proposed travel time (either for arrival or departure). The site then presents one or more recommended itineraries.

The user wants to be able to select a film, a cinema, and the earliest acceptable screening time; the application should then find the first satisfactory screening of that film at the chosen cinema, and submit an itinerary request from the user's home to the cinema, arriving fifteen minutes before the film starts. A combination that achieves this is shown in Figure 4. Figure 5 then shows a simple example of cloning: the user has created a further scenario to handle a second film name, and can now work with the two scenarios in parallel.



A: Gentofte

B: Gentofte Kino

C: Gentoftegade 39, 2820 Gentofte

D: Dirty Pretty Things

E: biografer & spilletider i dag og i morgen
(spilletider opdateres hver dag kl. 14)
Gentofte Kino
Onsdag 21:30
Torsdag 21:30

F: 19:00

G: Sortedam Dossering 81, 210

H: Gentoftegade 39, 2820 Gent

I: 21:15

J: Detailed view

Station/Stop	Date	Arr.	Dep.	Platform	Products	Commen
Sortedam Dossering 81, 2100 København Ø, Trianglen (HT)	12.11.03 12.11.03				walk	7 min.
Trianglen (HT) Østerport st./Oslo Plads	12.11.03 12.11.03	 20:48	20:45		Bus 1A	Bus Direction: Hvidovre hospital
Østerport st./Oslo Plads Østerport (S-tog)	12.11.03 12.11.03				walk	4 min. Or foot
Østerport (S-tog) Gentofte st.	12.11.03 12.11.03	 21:03	20:53		S B	Direction: Holte st. Carriage c bicycles
Gentofte st. Gentoftegade 39, 2820 Gentofte,	12.11.03 12.11.03				walk	11 min.

Figure 4: Using three Web applications in combination to provide travel information for film outings. The cell relationships established by clipping from the pages shown as thumbnails at the top are as follows: selecting a cinema in the list in cell A triggers navigation that places the cinema's name and address in B and C respectively; entering a film name in D causes E to hold a list of that film's Copenhagen-area screenings for today and tomorrow; cells G, H and I serve as start address, destination and arrival time for a travel enquiry that gives an itinerary in J. Cell F is an independently created cell in which the user specifies the earliest desired film start time. The applications have been connected by giving cell H (destination) a formula that copies the cinema address from C, and cell I (arrival time) a formula based on a user-written function for finding in cell E the first screening time relevant to cinema name B and time F. Any change to the cinema choice in A, film name in D, earliest start time in F, or starting address in G will trigger re-evaluation leading to a new itinerary.

The two examples described in this section illustrate the potential capabilities of C3W. In the following section we provide some details of C3W's implementation, as guidelines for practitioners wishing to develop similar systems.

3 IMPLEMENTATION REQUIREMENTS

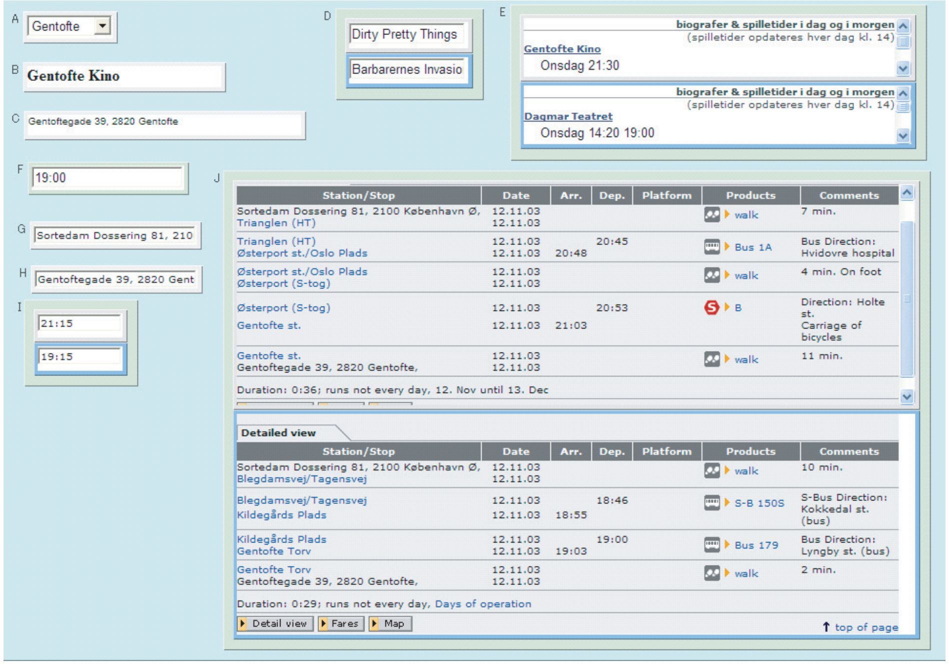
Many software platforms and toolkits could be used to implement a system supporting the operations shown above. C3W, which primarily addresses Web applications, is built on the PlexWare platform (K-Plex, Inc., San Jose and Tokyo; www.kplex.com). Before we illustrate how clipping, connecting and cloning are supported in C3W, we will explain briefly some implications of choosing this platform.

PlexWare is an implementation of the IntelligentPad meme-media architecture [22, 21]. In IntelligentPad, information and processing components are represented as 2D visual objects called pads. Each pad holds its state data in named slots, which also form its interface for functional connection to other pads. By the direct-manipulation pasting of one pad onto another, a user defines a child-parent relationship between them, and can also connect one slot in the child to one slot in the parent. This simple form of linkage is enough to enable pad compositions to act as compound multimedia documents, or hierarchically structured applications. The compositions are also easy to decompose – again by direct manipulation – in order to be reused or re-edited.

Within PlexWare, many powerful Microsoft controls – such as Internet Explorer (IE) and Excel – have been encapsulated as pads. Through our collaboration with K-Plex we had access to the source code involved in such encapsulation, and were able to modify this code to extend the capabilities of the encapsulated controls. In addition, though not described in detail in this paper, we were able to implement mechanisms for extracting and reusing elements within general pad compositions, enabling us to explore how non-Web applications may also be included in C3Sheet constructions.

3.1 Clipping

The activity that we call clipping is the fundamental mechanism by which a user extracts elements from existing appli-



Gentofte

Gentofte Kino

Gentoftegade 39, 2820 Gentofte

Dirty Pretty Things

Barbarernes Invasio

biografer & spilletider i dag og i morgen
(spilletider opdateres hver dag kl. 14)
Gentofte Kino
Onsdag 21:30

biografer & spilletider i dag og i morgen
(spilletider opdateres hver dag kl. 14)
Dagmar Teatret
Onsdag 14:20 19:00

19:00

Sortedam Dossering 81, 210

Gentoftegade 39, 2820 Gent

21:15

19:15

Station/Stop	Date	Arr.	Dep.	Platform	Products	Comments
Sortedam Dossering 81, 2100 København Ø, Trianglen (HT)	12.11.03 12.11.03				walk	7 min.
Trianglen (HT) Østerport st./Oslo Plads	12.11.03 12.11.03	 20:48	20:45		Bus 1A	Bus Direction: Hvidovre hospital
Østerport st./Oslo Plads Østerport (S-tog)	12.11.03 12.11.03				walk	4 min. On foot
Østerport (S-tog) Gentofte st.	12.11.03 12.11.03	 21:03	20:53		S B	Direction: Holte st. Carriage of bicycles
Gentofte st. Gentoftegade 39, 2820 Gentofte,	12.11.03 12.11.03				walk	11 min.

Duration: 0:36; runs not every day, 12. Nov until 13. Dec

Detailed view

Station/Stop	Date	Arr.	Dep.	Platform	Products	Comments
Sortedam Dossering 81, 2100 København Ø, Blegdamsvej/Tagensvej	12.11.03 12.11.03				walk	10 min.
Blegdamsvej/Tagensvej Kildegårds Plads	12.11.03 12.11.03	 18:55	18:46		S-B 150S	S-Bus Direction: Kokkedal st. (bus)
Kildegårds Plads Gentofte Torv	12.11.03 12.11.03	 19:03	19:00		Bus 179	Bus Direction: Lyngby st. (bus)
Gentofte Torv Gentoftegade 39, 2820 Gentofte,	12.11.03 12.11.03				walk	2 min.

Duration: 0:29; runs not every day, Days of operation

Figure 5: The composite Web application shown in Figure 4, after the user has cloned cell D. Cloning has been propagated automatically to all the cells that depend directly or indirectly on D: the screening times (E), derived arrival time (I) and the itinerary (J). The user may now wish to experiment by entering alternative film names in either instance of cell D, or could change some other input parameter and obtain new timings and itineraries for both films simultaneously.

cations for use in a custom interface. Any implementation of clipping requires detailed cooperation between the existing applications and the interface-construction environment. The necessary components are as follows:

1. An interaction mechanism by which the user identifies an element that is to be clipped, and installs it as a cell in the custom interface.

   Our support for clipping from Web pages is implemented by cooperation between an IE pad, with some extensions that we added, and a C3Sheet object. The extended IE supports the extraction of any Web-page region that can be expressed using an HTML-Path [23]. An HTML-Path is a specialisation of an XPath expression; it can specify any self-contained HTML tag (such as an input field, table row, or even a whole document), and also supports regular expressions for defining portions of text nodes. Once the user has steered an interactive highlight to the desired region of the page, that region can be dragged and dropped onto a C3Sheet to define a new cell.

2. A visual representation for the custom interface and the cells within it, where each cell can exhibit the behaviour of its source region in the original application.

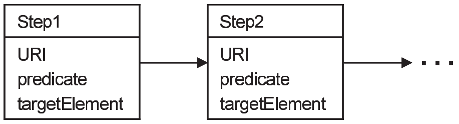
   Earlier we mentioned that each cell in C3W is in essence a fully functional Web browser – a Web-browsing pad – that works as a portal onto the region of the page from which the element was clipped. Therefore the cell has the interaction behaviour that the element would normally have within the full Web page: input fields, menus, buttons and the like all work as usual, as do display elements.

3. A way of capturing the dependency relationships between separately clipped cells, so that changes in input values result in re-evaluation of results.

   The most complex aspect of clipping is the capture of dependency relationships between cells. Cells in C3W are typically extracted from navigations that span multiple Web pages, and carry detailed information enabling the C3Sheet to determine their mutual relationships and to re-execute the navigation when necessary.

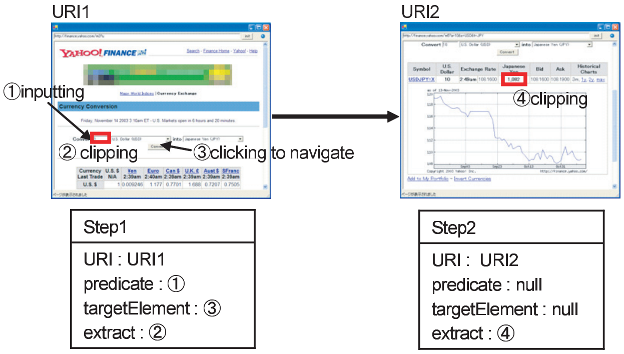
We now give further details of the processes that support clipping in C3W.

3.1.1 Capturing the Dependency Relationships Between Elements As the user navigates using our browser, a navigation path of the following form is recorded:



The path is a sequence of steps representing encountered documents. A step consists of the URI of the document, a 'predicate' specifying any conditions that the user has specified on that document, and 'targetElement' specifying the interaction that took the browser to the next document (if any). The predicate conditions correspond to values specified in elements such as INPUT or SELECT; the targetElement is typically an A (anchor) or FORM. All these attributes are recorded using HTML-Path expressions, as described above.

When the user clips elements from the documents, the path is augmented with ‘extract’ attributes. The following figure shows how navigation actions within the Yahoo! currency-conversion application contribute to the recorded path, and a specific example of the attribute values recorded when the user clips the entry field for the starting-currency amount on the top page, puts the value ‘10’ into that field, presses the Convert button (to submit the form), then clips the converted amount on the result page:



Step1
URI : URI1
predicate : ①
targetElement : ③
extract : ②

Step2
URI : URI2
predicate : null
targetElement : null
extract : ④

Values
URI1 : http://quote.yahoo.com/m5
URI2 : http://quote.yahoo.com/m5?a=10&s=USD&t=JPY
① : HTML[1]/BODY[1]/CENTER[1]/FORM[1]/.../INPUT[1]/@value = "10"
② : HTML[1]/BODY[1]/CENTER[1]/FORM[1]/.../INPUT[1]
③ : HTML[1]/BODY[1]/CENTER[1]/FORM[1]
④ : HTML[1]/BODY[1]/CENTER[1]/P[4]/TABLE[2]/.../TD[2]/B[1]

Whenever a user drops a new element on a C3Sheet, it brings with it the full multi-step navigation path describing how it was extracted. The C3Sheet compares this path against the paths for existing cells, and thus determines their dependency relationships. In our Yahoo! example the conversion-result cell becomes a dependant of the currency-amount cell, with a single navigation step between them. If a currency-selection field were also extracted as a cell, the result would be a dependant of that cell too.

Note that this analysis of navigation paths is not affected by the order in which the cells are created. This is why, as mentioned in Section 2.1.1, a user need not decide in advance which navigations he or she will want to reuse, but can do so in retrospect – i.e., after having seen some initial results. In such a case the user could clip the desired result elements, then backtrack to earlier pages and clip the input elements. By the same token, C3W can support clipping from a branched navigation – for example, pursuing two different links from a film-information page to clip both actor information and local reviews.

3.1.2 Replaying Derivations Later, when a new value is supplied for some cell that is acting as an input, C3W must replay the appropriate navigation steps to obtain the results for any dependent cells. For this it uses an instance of Web browser that it holds behind the scenes, iteratively executing the following procedure for each navigation step:

1. Set up any conditions specified by predicate attributes.
2. Set up any values connected to input cells; propagate any values connected to output cells.
3. Access the element specified by the targetElement attribute, to jump to the next step. This jump goes to a URI that depends on the user’s input values at the time of replay, and may therefore override the URI attribute value originally recorded for each step.

Note that, in contrast to the formulas specified by users to connect applications, the details of the dependencies between input and result cells extracted from a given application are typically hidden. To help users understand these hidden relationships it would be possible to create a dummy formula for each result cell, revealing which other cells it depended on and perhaps some clues about the kind of derivation involved (e.g., whether it is a Web navigation or a pad-based calculation). These dummy formulas could also contribute to a global mechanism for helping the user to trace dependencies through the sheet.

Although the specifics for clipping from other kinds of application are different, the facilities required are equivalent. For example, if the user is creating cells by clipping pads from within large IntelligentPad composites, then instead of a Web navigation path each cell includes a precise description of its pad’s position within the hierarchical structure of the source composite; instead of holding a Web browser behind the scenes to re-evaluate the navigation, the C3Sheet must maintain a working copy of the relevant parts of the composites.

3.2 Connecting

Connecting refers to applying formulas to cells, so that derivations normally driven by direct user input can instead be driven by changes in the values of other cells.

Formulas in the C3W prototype are handled by a hidden instance of Excel spreadsheet. A C3Sheet cell with label x corresponds to the cell at location $x1$ on the Excel sheet – i.e., the cell labelled C is mapped to C1, and so on. When the user adds a formula to a C3Sheet cell, the formula is first scanned for references to other cell labels, these references are converted to spreadsheet cell locations, and the derived formula is then attached to the appropriate cell on the spreadsheet.

A cell on a C3Sheet always corresponds to some clipped unit of HTML, including its enclosing tag. However, in transferring values between the C3Sheet and Excel a distinction is made between two types of cell: those whose essential content can be characterised as a simple character string, and those that have a more complex, typically nested, structure. For the former category, Excel only handles the inner content string. This applies not just to simple textual-result cells and input fields, but also to cells whose main function depends on a single string property – such as the HREF property of an A (anchor) tag, or the SRC property of an IMG (image) tag. Thus for simple elements a user can write simple formulas, such as ‘=B*100’ for passing a stock-price result into a currency-conversion input, while complex elements can be handled using Excel’s full macro-language capabilities – for example to realise the specialised string-handling for the ‘filmtime’ function shown in Figure 4.

In the current prototype, the distinction between simple and complex cell types is made on the basis of hard-coded rules. However, realising that there will be exceptions, in the future we may make it a user-selectable option. Meanwhile we are considering what kind of built-in function library could help users build formulas for complex structure analysis, and are also examining how to incorporate interactive, direct-manipulation techniques that will let users identify elements within structured data without having to write code.

3.3 Cloning

The cloning facilities are based on our work on Subjunctive Interfaces [11, 12, 14]. Computer users often want to compare alternative scenarios, but few applications provide good support for the side-by-side viewing that would facilitate such comparison; equipping an application with a subjunctive interface allows users to set up multiple scenarios, then view and control those scenarios in parallel.

C3W's support for parallel viewing and control of scenarios is derived from the widget-multiplexer approach [13]. Using widget multiplexers involves reserving a dedicated region of the computer screen for each widget that appears in the normal, single-scenario application interface. As the user creates additional scenarios, each multiplexer displays simultaneously, side by side, how its widget would appear in each scenario. In general, if there are n scenarios then each multiplexer displays n copies of its widget. The exception is when a widget would have identical appearance in all scenarios, in which case only a single display is needed. All multiplexers lay out their scenario-specific displays in the same way; by the principles of 'small multiples' visualisations [26], this helps users to locate all the information that comprises each scenario.

The cloning facilities of C3W work as follows:

- The user can create a new scenario by cloning some chosen input cell. This introduces an independent calculation passing through all the cells on the sheet. However, because formula-based derivation is uni-directional, only the chosen cell and those cells that depend on it need to prepare and show multiple displays. The other cells still have the same value in every scenario.

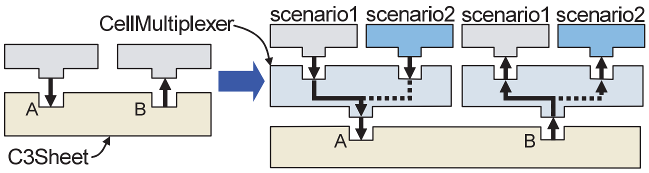


Figure 6: The use of CellMultiplexers to support cloning in a single-layer composite pad. On the left is the basic composite pad; on the right is a setup for two scenarios. The child pads connected to slots #A and #B have both been cloned, and the clone pairs are connected to the parent through separate cooperating CellMultiplexers. When the CellMultiplexers are handling scenario 1, the parent is connected (indirectly) to the scenario-1 instances of its children; when handling scenario 2, the other instances are connected.

- Entering a new value in one display of a cloned cell will only affect the scenario corresponding to that display. Entering a new value in an uncloned cell will affect all scenarios simultaneously.
- When multiple scenarios are on view, the user can choose to clone a previously uncloned cell so that it has separate displays for each scenario. Any cells that depend on the newly cloned cell, but were not cloned before, will now be cloned.
- The user can delete a scenario. This removes the clones associated with that scenario from all cloned cells.

The fact that C3W is implemented on the PlexWare platform as a set of cooperating pads allows us to support multiplexing in a particularly straightforward way. Each cell is a pad, that is a child of the C3Sheet pad and has a slot connection to a cell-specific slot within the parent. For each cloned cell, we introduce between the cell and the parent C3Sheet a specialised pad called a CellMultiplexer. The CellMultiplexer is responsible for creating and deleting clones of the cell, and for time-multiplexing the connection between a cell's various clones and the C3Sheet. Figure 6 illustrates the principle of a CellMultiplexer, showing two child pads that are each cloned to support two scenarios.

The various CellMultiplexers within a single application (i.e., attached to the same C3Sheet) coordinate their displays so that the clones belonging to a given scenario all have the same colour of border, and the scenarios are laid out in the same way. The colouring and layout policy is a simplified version of that described in [14]. As seen in the examples in Section 2, cell clones in C3W have the same size as their original cell, which imposes a practical limit on the number of clones – especially for large cells, such as the itinerary in Figures 4 and 5. We are working to improve this and other aspects of the usability of the cloning facilities.

4 RELATED WORK

4.1 Automating Access to Web Applications

Clipping elements from Web applications relates to the highly active research area on automated extraction of information from the Web.

4.1.1 Recording and Replaying Web Interaction Much usage of the Web is repetitive; users often want to revisit pages that they have accessed before. Many projects start from the observation that bookmarks, the basic mechanism for supporting revisitation, are not enough. Hunter Gatherer [19] is one such project, supporting users in collecting not just links to pages that they want to revisit, but copied portions of those pages. The copies can include text, images, links, or other active components such as forms; the active components retain the behaviour that they have within the original Web pages.

One benefit of being able to copy just selected regions within pages is that the user can construct a simplified interface, free from the clutter of surrounding elements that are irrelevant to that user or to a particular instance of use. This was one goal in the development of WebViews [6], in allowing users to create customised views suited to small-screen devices, and also of WinCuts [20], which lets users create live copies of arbitrary sub-regions of general windowed applications.

WebViews is not only a system for replicating components of Web pages; like the WebVCR [1] system that it extends, its key feature is so-called smart bookmarks that point not to a single location within the Web, but encapsulate a series of Web-browsing actions. Smart bookmarks can therefore be used to record hard-to-reach Web pages, that have no fixed URIs and can only be reached by navigation. However, the decision to make a smart bookmark must be taken before the user starts the navigation that is to be recorded; this is in contrast to C3W's support for extracting elements retrospectively, which we have not found in other published work.

When a user records a smart bookmark, he or she can indicate if some field in a form (e.g., a password) is to be requested at playback time, rather than stored with the bookmark. However, in essence the job of a smart bookmark is to arrive at the destination Web page and to deliver to the user's display either the whole page or some chosen parts thereof; intermediate pages are not seen. In this respect C3W offers greater flexibility, since the user can clip arbitrary regions of any page encountered during the course of a navigation, either to reveal intermediate results or to capture inputs. And because each clipped region is treated as a portal [18] onto the original Web page, inputs to be used during replay can be based on arbitrary form widgets – including menus, checkboxes and the like – rather than just textual input fields.

For presenting results, too, the portal approach has some benefits. For example, in the cinema interface shown in Figures 4 and 5, the carefully designed itineraries provided by the travel-planning application can be viewed in full detail. However, if the user wants to use the contents of such a semi-structured display as a source of values for further processing, he or she must typically write a complex extraction function – as exemplified by the 'filmtime' function in the same application. In such a case, it may be preferable for the system to include some automated facilities for formalising the semi-structured content.

4.1.2 *Formalised Extraction of Content*

Many researchers are working on how to extract structured data, suitable for storage in databases or processing by agents, from Web pages that were designed for human readers; [9, 10] are recent surveys of this area.

Lixto [3] and InfoBeans[2] are two research systems that, like C3W, aim to define what data is to be extracted by generalising from example selections made by a human user. The goal is a definition that will extract equivalent information from Web pages with different contents, and perhaps with different structures – because information providers on the Web often make changes to their pages (such as adding or removing menu items, or slightly altering how items are grouped) that, while causing no difficulty for human readers, can easily confuse a formal system. The HTML-path expressions used by C3W are not robust against such changes in page structure. The XPath expressions used by WebViews include additional conditions that help to make the extraction more robust, and the system also uses various heuristics to work around common kinds of structural change. The scripts used by InfoBeans can incorporate context, landmark and characterisation conditions to increase their robustness; if a script still fails to locate suitable content, the system includes TrIAs (Trainable Information Agents) that can start a dialogue with the current user to obtain further example-based training.

However, no amount of heuristics or training can guarantee to find the intended elements in the face of arbitrary changes. Here again, the fact that C3W encourages the display of intermediate results rather than hiding them in a processing pipeline can be of some benefit. For example, if some change in a Web resource causes values that are normally numeric to be replaced by strings, our portal-based clipping at least increases the likelihood that the user will see that something has gone wrong, rather than being puzzled by (or, worse, failing to notice) the outcome of strange derived values flowing through an unseen pipe.

4.2 Passing Information Between Applications

In C3W, separate Web applications can be connected with the help of formulas on input elements. Other systems for working with Web applications offer alternative approaches: Lixto provides a visual patch-panel mechanism; InfoBeans allow the user to define 'channels' for each InfoBox component, which are then used to connect them; earlier work on wrapping Web applications as pads within an IntelligentPad environment [8] allowed the applications to be connected using the pads' slot-based communication mechanism. In Snap [17], a system supporting construction of customised visualisations, users connect visualisation components by using dialog-box mechanisms to set up parameterised queries between them.

The cell-and-formula approach of C3Sheets is based on the spreadsheet model; other well known spreadsheet-based systems include Forms/3 [4], Information Visualization Spreadsheets [5], and C32 [15]. In general, such projects selectively adopt or abandon various features of what could be called the traditional spreadsheet. The use of formulas in C3Sheets, for example, conforms to the tradition that each formula determines the value of just one cell. On the other hand, our implementation is not alone in breaking free of the restriction that cells should contain only text or numbers, or the idea that cells should be positioned and named according to a tabular grid.

4.3 Handling Multiple Scenarios in Parallel

As explained in Section 3.3, the cloning mechanisms in C3W reflect our interest in supporting the viewing, comparison and update of alternative scenarios. This applies not only to the kind of information exploration addressed in this paper, but also in design, such as when investigating the influence of image placement on the layout of a document, or in simulation, such as when testing how alternative population growth scenarios would affect a country's economy. But such comparison is seldom well supported; Terry and Mynatt [24] speak of the 'single-state document model' of most of today's applications, and the implicit barrier that this imposes on a user who wants to explore and compare alternatives. In addressing this issue for simple design tasks, Terry et al. [25] demonstrate an interface that lets a user set up and work with multiple designs in parallel, although their interface is limited in its support for viewing scenarios side by side.

In the domain of Web browsing we have found little work supporting interactive comparison of scenarios. One isolated example is the Comparative Web Browser (CWB) [16], an interface that offers comparison and synchronised browsing between similar Web pages. However, CWB displays the pages in full, which makes it less scalable than the C3W approach in which the user is able to clone just chosen cells, and where those cells may have been taken from several Web pages.

Relative to our own previous work on subjunctive interfaces, a key advance embodied in C3W is that the widget multiplexers themselves are generic; they can handle any client that is a cell, and in C3W a cell can contain any clipped region of a Web page. This makes it the first system to offer subjunctive-interface facilities for an entire class of applications.

5 CONCLUSION

We have introduced C3W, a prototype that supports users in creating custom interfaces for accessing information through Web applications. C3W exemplifies three mechanisms that together help users to overcome inconveniences and restrictions imposed by the original applications, as follows:

1. Clipping: By drag-and-drop manipulation, a user can select and extract input and result elements from the pages of a Web application. Placing the elements on a substrate (a C3Sheet) turns them into cells that work as portals onto the original Web pages; cells containing clipped input elements support user input, and cells containing result elements display the corresponding results. Thus the user can create a compact, reusable interface that excludes unnecessary features of the Web application.

2. Connecting: A single C3Sheet can hold input and result cells for multiple applications. The user can define formulas for input cells, so that they obtain their values from the contents of other cells. By defining formulas that refer to the result cells of other applications, the user can create connections between applications that were not originally designed to work together.

3. Cloning: The user can set up multiple scenarios – i.e., different settings for the inputs, leading to different results – to be shown in parallel. Each cell displays, side by side, its contents in the various scenarios. So even if the original application could only handle one scenario at a time, the user can efficiently explore and compare the different results available through that application. This also works for applications that have been connected using formulas.

As demonstrated in this paper, even the current prototype lets us build useful interfaces for our own needs, as well as helping us to explore the wider promise of this approach. That said, there are several areas to be addressed in going beyond this stage of development.

First, the existing C3Sheet implementation has many interface features that should be improved. We are considering how to incorporate strategies found in other spreadsheet-like interfaces, such as for partial automation of cell sizing and layout, for more informative labelling of cells (including the ability for a user to provide clarifying annotations), and for assistance in creating formulas. We are also working to improve the usability of cloning operations, and the scalability of the cloned-cell displays.

Second, our support for Web applications can be improved by incorporating techniques demonstrated in other projects, such as for enhancing the robustness of element identification in the face of changes to the underlying pages, and for capturing a broad range of user actions across applications of different types. Content-formalisation techniques can be applied to help users extract richer data types from Web-page elements for use in formulas.

Third, the broader benefits of the C3Sheet approach depend on supporting not just Web applications and IntelligentPad composites, but other kinds of application for information access or derivation. Obvious candidates for inclusion are local database applications, spreadsheets, and statistical packages. Their integration, through implementation of the facilities outlined in Section 3, is likely to be achievable through the use of existing APIs and object-broker services.

All the above development directions will naturally need to be backed up with user evaluations. In the first instance we are working with colleagues who rely on combined use of Web applications in the course of their (non-computing) research; our hope is to confirm that using C3Sheets will benefit their work, and to improve our understanding of the facilities that must be made available.

Thus we regard the work reported in this paper as initial steps towards facilities that, while not technically hard to achieve, could make a profound difference to how people can benefit from the many potentially cooperating applications available to them.

6 ACKNOWLEDGEMENTS

We are grateful to the many reviewers who provided helpful comments. In our examples we used a stock-price application provided by CNN (money.cnn.com/markets/), currency exchange from Yahoo (quote.yahoo.com/m5), film lookup from Alt om København (www.aok.dk/Copenhagen/Film/) and itinerary search from Rejseplanen (www.rejseplanen.dk/).

REFERENCES

1. Anupam, V., Freire, J., Kumar, B., & Lieuwen, D. Automating web navigation with the WebVCR. In *Proceedings of WWW 2000* (May 15–19, Amsterdam, The Netherlands), ACM, NY, 2000, pp. 503–517.

2. Bauer, M., Dengler, D., & Paul, G. Instructible information agents for Web mining. In *Proceedings of IUI '00* (Jan 9–12, New Orleans, LA, USA), ACM, NY, 2000, 21–28.

3. Baumgartner, R., Flesca, S., & Gottlob, G. Declarative information extraction, Web crawling, and recursive wrapping with Lixto. *LNAI 2173* (Sept 2001), 21–41.

4. Burnett, M., Atwood, J., Djang, R. W., Gottfried, H., Reichwein, J., & Yang, S. Forms/3: A first-order visual language to explore the boundaries of the spreadsheet paradigm. *Journal of Functional Programming 11*, 2 (Mar 2001), 155–206.

5. Chi, E. H., Riedl, J., Barry, P., & Konstan, J. Principles for information visualization spreadsheets. *IEEE Computer Graphics and Applications 18*, 4 (July/Aug 1998), 30–38.

6. Freire, J., Kumar, B., & Lieuwen, D. WebViews: accessing personalized Web content and services. In *Proceedings of WWW 2001* (May 1–5, Hong Kong, China), ACM, NY, 2001, pp. 576–586.

7. Fujima, J., Lunzer, A., Hornbæk, K., & Tanaka, Y. C3W: Clipping, Connecting and Cloning for the Web. In *Alternate Track Papers and Posters of WWW 2004* (May 17–22, New York, NY, USA), ACM, NY, 2004, pp. 444–445.

8. Ito, K., & Tanaka, Y. A visual environment for dynamic web application composition. In *Proceedings of HT 2003* (Aug 26–30, Nottingham, UK), ACM, NY, 2003, pp. 184–193.

9. Kuhlins, S., & Tredwell, R. Toolkits for generating wrappers – a survey of software toolkits for automated data extraction from Web sites. *LNCS 2591* (2003), 184–198.

10. Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., & Teixeira, J. S. A brief survey of web data extraction tools. *SIGMOD Record 31*, 2 (June 2002), 84–93.

11. Lunzer, A. Choice and comparison where the user wants them: Subjunctive interfaces for computer-supported exploration. In *Proceedings of INTERACT '99* (Aug 30–Sept 3, Edinburgh, Scotland), IOS Press, Amsterdam, The Netherlands, 1999, pp. 474–482.

12. Lunzer, A., & Hornbæk, K. Side-by-side display and control of multiple scenarios: Subjunctive interfaces for exploring multi-attribute data. In *Proceedings of OZCHI 2003* (Nov 26–28, Brisbane, Australia), IEEE Computer Society Press, Los Alamitos, CA, 2003, pp. 202–210.

13. Lunzer, A., & Hornbæk, K. Widget multiplexers for side-by-side display and control of information-processing scenarios. In *Adjunct Proceedings of HCI International 2003* (June 22-27, Crete, Greece), Lawrence Erlbaum Associates, Mahwah, NJ, 2003, pp. 91–92.

14. Lunzer, A., & Hornbæk, K. Usability studies on a visualisation for parallel display and control of alternative scenarios. In *Proceedings of AVI 2004* (May 25–28, Gallipoli, Italy), ACM, NY, 2004, pp. 125–132.

15. Myers, B. Graphical techniques in a spreadsheet for specifying user interfaces. *Proceedings of CHI '91* (Apr 27–May 2, New Orleans, LA, USA), ACM, NY, 1991, pp. 243–249.

16. Nadamoto, A., & Tanaka, K. A comparative web browser (CWB) for browsing and comparing web pages. In *Proceedings of WWW 2003* (May 20–24, Budapest, Hungary), ACM, NY, 2003, pp. 727–735.

17. North, C., & Shneiderman, B. Snap-together visualization: Can users construct and operate coordinated views? *International Journal of Human-Computer Studies 53*, 5 (Nov 2000), 715–739.

18. Olston, C., & Woodruff, A. Getting portals to behave. *Proceedings of InfoVis 2000* (Oct 9–10, Salt Lake City, UT, USA), IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 15–26.

19. schraefel, m. c., Zhu, Y., Modjeska, D., Wigdor, D., & Zhao, S. Hunter Gatherer: Interaction support for the creation and management of within-web-page collections. In *Proceedings of WWW 2002* (May 7–11, Honolulu, HI, USA), ACM, NY, 2002, pp. 172–181.

20. Tan, D. S., Meyers, B., & Czerwinski, M. WinCuts: manipulating arbitrary window regions for more effective use of screen space. In *Extended Abstracts of CHI 2004* (Apr 24–29, Vienna, Austria), ACM, NY, 2004, pp. 1525–1528.

21. Tanaka, Y. *Meme Media and Meme Market Architectures: Knowledge Media for Editing, Distributing, and Managing Intellectual Resources*. Wiley-IEEE Press, 2003.

22. Tanaka, Y., & Imataki, T. IntelligentPad: A hypermedia system allowing functional compositions of active media objects through direct manipulations. In *Proceedings of the IFIP 11th World Computer Congress* (Aug 28–Sept 1, San Francisco, CA, USA), North-Holland/IFIP, 1989, pp. 541–546.

23. Tanaka, Y., Kurosaki, D. & Ito, K. Live Document Framework for Re-editing and Redistributing Contents in WWW. In *Proceedings of EJC 2002* (May 27–30, Krippen, Germany), IOS Press, Amsterdam, The Netherlands, pp. 247–262.

24. Terry, M., & Mynatt, E. D. Recognizing creative needs in user interface design. In *Proceedings of C&C 2002* (Oct 13–16, Loughborough, UK), ACM, NY, 2002, pp. 38–44.

25. Terry, M., Mynatt, E. D., Nakakoji, K., & Yamamoto, Y. Variation in element and action: Supporting simultaneous development of alternative solutions. In *Proceedings of CHI 2004* (Apr 24–29, Vienna, Austria), ACM, NY, 2004, pp. 711–718.

26. Tufte, E. R. *Envisioning Information*. Graphic Press, Cheshire, CT, 1990.