

## Design Study 3: Dynamic Workspace for Document Assembly and Navigation

Larry L. Constantine  
Lucy A. D. Lockwood  
Constantine & Lockwood, Ltd.  
<http://www.forUse.com>

**Abstract:** *Assembling reports or documents by pulling together and organizing disparate content from a variety of sources is a common task in many different work situations. Standard components and interaction techniques supported by conventional graphical user interfaces make it possible to accomplish such tasks but hardly can be said to facilitate simple and efficient use. This design study reviews the design of a novel facility for allowing users at all levels of sophistication and skill to quickly and easily construct complex documents using resources obtained from numerous and varied sources. The resulting “dynamic workspace” was inspired by the way teachers actually assemble and sort notes and bits of information on their desks in preparation for creating lesson plans or other documents. Design tradeoffs in efficiency, ease of use, and screen real estate are discussed.*

**Keywords:** instructive interaction, document assembly, information management, navigation, usage-centered design, design innovation, user interface design, interaction design, usability, performance support

---

Learn more about usage-centered design at <http://www.forUse.com>.

---

### General Background

A browser-deployed performance-support system for K-12 classroom teachers presents many design challenges. An effective solution must be flexible enough to accommodate to an enormous variety of teacher working styles in many different work settings. At the same time it must be efficient and easy to use. (For more background on the application and project, see the first report in this series [Constantine and Lockwood, 2001].)

In usage-centered design [Constantine and Lockwood, 1999], the ultimate guide to good design is the nature and structure of the tasks to be supported. Innovation is neither sought for its own sake nor avoided in deference to a real or imagined legacy. Solutions to tough problems in visual and interaction design follow from an understanding of the nature of the work and how it is and can be carried out.

A key part of the work for many classroom teachers today is lesson planning. Every lesson over the course of every day must be planned with specific objectives, resources, and activities. Lesson plans vary tremendously in complexity and the activity of planning a lesson can range from pulling last year's plan from a file to assembling an entirely new plan from scratch. In general, plans are reviewed and updated with new material even if they have been used before. Because the typical classroom teacher has less than 15 minutes a day for lesson planning, performance-support software must be extremely easy to use efficiently. Knee-jerk design solutions, such as wizards or step-by-step dialogs, are unacceptable because the task itself is highly sophisticated and variable and because teachers are reluctant to give up their distinct individual approaches to lesson planning developed through long experience.

### **Document Assembly**

Lesson plans come in many forms and formats and the details of what goes into a given plan may vary from lesson to lesson. To create or revise a lesson plan, a teacher may need to pull together information and ideas from a variety of sources. A Web site for subject specialists might be consulted for up-to-date data. Lesson plans from other teachers might be scoured for ideas or for specific content to be incorporated in the new or revised plan. Quizzes, exercises, readings, student research materials, and a variety of other resources might need to be located and compiled from many different sources. Some of these materials might be used as-is, some might be adapted in some way, and others might supply quotes, illustrations, URLs, or other useful bits and pieces. Some material might itself be incorporated in one form or another in a lesson plan while other material might be gathered only for inspiration, review, or comparison.

In the absence of computer support, teachers might assemble such material on the top of their desks or on a table or other work surface. It is a hallmark of the process that the order in which material might be gathered and in which it might be used is completely unpredictable and dependent not only on the task and the teacher but also on the particular material located. Like Web surfing, the path through the process is non-linear and unpredictable. Another hallmark of the process is that for it to work well, all the material gathered at any point really ought to be visible and readily accessible throughout the process, which is why so many writers, teachers, and others involved in the creative transformation of information tend to spread out their research material over a wide area, with stacks and clusters arrayed around them on desks, tables, and even the floor.

We call this general process "document assembly." Document assembly is a common and recurrent task for knowledge workers and other professionals of many kinds. Indeed, it can be regarded as an archetypal task "pattern," generic activity common to many tasks in many areas of application.

Conventional software support for document assembly is not very good. Many programmers would argue that the standard tools of cut-copy-and-paste through the so-called clipboard supply all the functionality needed to solve the document-assembly problem. While this may be true from a strictly functional point of view, from the standpoint of ease and efficiency of use, these facilities leave much to be desired. As usually implemented, these facilities require the user to keep switching between visual interaction contexts, going to a source, copying, switching to the target, pasting, then back to a source again. For maximum efficiency, users must resort to the arcana of keystroke shortcuts, namely the ubiquitous and inconsistent `ctrl-X`, `ctrl-C`, and

ctrl-V. A surprising number of casual and non-technical users never employ these techniques.

Unlike the top of a desk or a work table, the clipboard that carries information to be copied or moved is itself (normally) invisible and only holds a single item. (The multi-object, dockable clipboard now supplied by Microsoft is slowly evolving into a useful tool for document assembly, but still misses the mark in many ways.)

## **Design Objectives and Constraints**

The overall design objectives for this project—flexibility, simplicity, efficiency of use, and ease of learning—apply with particular salience to document assembly because the facility is one that will be used throughout the system to many different purposes. An awkward, obscure, or inefficient solution could markedly reduce the usability of the entire system.

We were particularly keen to make efficient use of the limited screen real-estate available for an application running inside a browser window. Increasing the challenge was the fact that, in many of the targeted schools, older computers with only SVGA (800x600) resolution were common. We had also already concluded that window-management overhead and context-switching needed to be minimized for efficient use.

Based on our analysis of teacher tasks in document assembly, we concluded that an effective facility for this purpose needed to be (1) persistent, (2) visible, (3) compact, and (4) instructive. In addition, it needed to be intelligent in the sense that it should make reasonable assumptions in the interest of simple and efficient operation.

By persistent, we meant that the facility needed to be available at all times from any part of the system. By visible, we meant that the facility and its contents should appear on the screen. By compact, we meant that this persistent and always visible facility had to take up an absolute minimum of screen real estate consistent with its function. Instructive interaction refers to our approach to assuring that the function, use, and behavior of even novel features can be correctly guessed by typical users on first encounter.

Overall, anything that a teacher might reasonably do with notes and papers assembled on the top of a desk ought to be as readily accomplished with our new scheme. This includes such things as rearranging materials, discarding them, clearing the desk, and the like.

## **Design Tradeoffs**

Limited screen real-estate versus the need for visibility guided us toward an adjustable workspace that could be collapsed or expanded as needed. An effective workspace would need to hide or reveal itself as appropriate while otherwise staying out of the way. Ideally, the workspace would hide and reveal its contents automatically as needed. In addition, it should keep all contents visible in as little space as possible.

It soon became clear to us that there were many conditions under which the user might want to return to the source of any information to review the context, for example, or to extract additional material. This suggested to us the idea of overloading the workspace to allow it to serve as a means for navigating among a number of selected documents.

Efficient operation meant that the workspace had to perform its support functions with the least number of actions on the part of the user. At the same time, it was

important that the workspace serve as a tool supporting the user rather than imposing a logic or process of its own.

## Dynamic Workspace

The dynamic workspace we designed is quite simple in concept. We took as our point of departure the inadequacies of the conventional Windows-style clipboard, bit-by-bit extending the concept to support the document assembly tasks more effectively. The result has elements in common with extended clipboard features as well as with the “dock” in Apple’s OS X for the Macintosh, which it anticipated.

The original design for the workspace as documented in engineering specifications is shown in Figures 1 and 2. The workspace serves as a holding bin for files, documents, images, pointers, fragments (what Microsoft calls “scraps”), or any other object known to the application. It also functions as a document launcher, allowing the user to return repeatedly to any open document or page of current interest. In effect, it combines within the application frame the functions of both the Windows Task Bar and the desktop itself, giving the user complete command in a freeform environment allowing for completely flexible workflow.

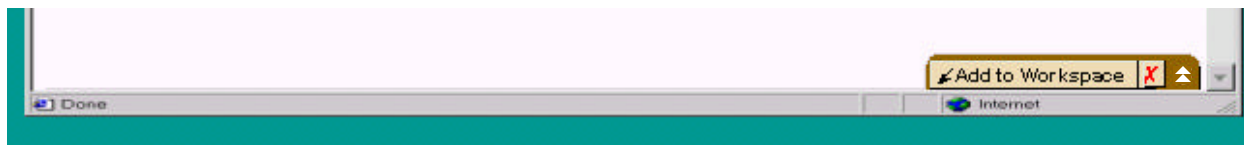


Figure 1 – Design for a dynamic workspace, shown closed.

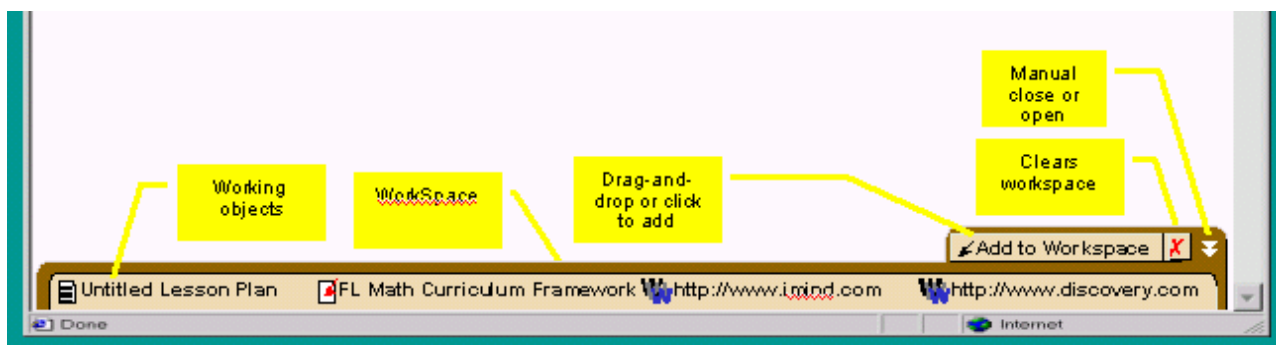


Figure 2 – Design for a dynamic workspace, shown open.

The workspace, which is always present, effectively links all the views and subsystems of the complete application into a seamless working environment for the user. It enables the user to keep track of and move quickly among any number of resources without window-management overhead. It enables easy and flexible assembly of materials from a variety of resources into any view or document within the application.

### *Instructive interaction*

The workspace is a non-standard control with non-standard behavior. In order to make it completely self-teaching, we used the techniques of instructive interaction. Instructive interaction is a body of both established and novel techniques that promote correct use of new features by first-time users. For example, visual bevels give the workspace the appearance of a tray or document to which the user might be able to drag objects, which is reinforced when the user drags something to it. In addition, although the workspace supports standard Windows editing keystrokes, it also has

visible controls for adding and deleting items, and these are augmented by explanatory tool tips.

Animation is also used to clarify behavior. When the workspace opens, it rolls upward, progressively revealing its contents. When users collect items by adding them to the workspace, they see the workspace expand to accommodate them and then see them inserted. The resemblance of objects within the workspace to documents in a file view or on the Windows desktop encourages users to try appropriate operations, such as double-clicking to launch or open or dragging to insert. The fact that the workspace remains in all views, encourages users to consider it as a holding bin or way station for information, even without being told of this potential. The obvious ease of adding or removing items further encourages experimentation with its use.

### ***Appearance and Behavior***

The workspace itself is located across the lower margin of the browser window, making itself available as needed and appropriate, but occupying a minimum of screen real estate. Its normal state when empty is closed, so that only its “tab” shows, revealing controls for adding items, deleting, and for opening or expanding the workspace.

The power of the workspace is in the subtle sophistication of its behavior, although its sophisticated operation is designed to be completely natural and largely transparent to the user. The Workspace can be “opened out” or closed manually, but it also does so automatically as needed. In the open state, it expands to accommodate as many items as are added to it, up to a preset limit of a certain number of rows. The workspace automatically opens out whenever a new item is added to it or on mouse-over after a short delay tweaked to avoid unintentional expansion; it closes again on the mouse leaving the area (after another brief delay) or on some other user action.

Items can be added to the workspace by any of the conventional Windows interaction idioms, including (a) selection followed by clicking on the Add to Workspace button, (b) copying to the clipboard followed by clicking on the Add to Workspace button, (c) drag-and-drop to the workspace, even if closed, or (d) cut or copy followed by shifting focus to the workspace and then a paste operation. In addition, an entire document can be added to the workspace by implicit operations described below.

The workspace is tailored for simple and efficient operation based on built-in understanding of the nature of the supported tasks and the information being handled. If a fragment of text is added to the workspace, a pointer to its source is invisibly attached so that the user can return to where the information originated. If a picture on a Web site is selected and dragged or otherwise added to the workspace, it is copied along with an invisible URL linking it back to its source.

### ***Implicit selection and smart operation***

We wanted the user to be able to simply and efficiently collect a series of pages or whole documents as well as selected portions of these as desired. This requires making the workspace smart in the sense of matching its actions to the most likely intentions of a user. For example, if a user goes to add to the workspace without having selected any part of the visible document, the workspace should not chastise the user with an annoying ping or a time-wasting modal error message; it should do something reasonable.

We spent a considerable amount of time figuring out what was reasonable based on the task model and then devising a simple set of rules for the needed behavior. In the case where nothing is selected in the current document, the reasonable assumption is that it is the document itself that is of interest, hence clicking on the Add to Workspace adds the entire document or page to the workspace. The exception is when the last user operation placed something on the clipboard and the clipboard has not been inserted into the workspace, in which case the contents of the clipboard are added to the workspace.

In either case, the first-time user is clued into these actions by visible feedback consistent with our design philosophy of instructive interaction: the workspace opens if not already open, and the appropriate title and representative icon are shown being added to the workspace. In this way, a user can quickly collect a whole series of files or Web pages of interest just by finding each of them and clicking on the Add to Workspace control.

Throughout the application, tool tips and other mouse-over effects are used to provide cues for beginning users as well as additional information. Dragging an object to the workspace opens it, indicating drop acceptance. On mouse-over of an object within the workspace, details are displayed, including the full name or title of the item (first line of a text fragment), type of item (lesson plan, text, figure, Web object, etc.), file details or URL as relevant, and thumbnail “print-preview” image. (Such a thumbnail image, though no doubt valuable to the user in making sense and keeping track of the workspace contents, proved too difficult to implement in the first release.)

While a single-click on an object in the workspace selects and highlights it, a double-click on an item within the workspace opens the object—or its source if it is a fragment—and changes the view to that source. Thus, once a collection of items has been assembled, the user can quickly move around among just those items simply by double-clicking on them from the workspace. To enhance the value of the workspace for moving quickly among documents of interest, lesson plans within the system are automatically added to the workspace as a complete document when opened by the user.

As shown in Figure 2, objects within the workspace appear as small icons indicating the type of object along with names or titles. Anything the user can select and drag or copy to the workspace can be held by it. In addition, entire documents of any type known to the software can be held in the workspace. Each row of the workspace will hold up to a certain number of objects, depending on the available horizontal space in the browser window. Additional objects wrap to the next row, and the workspace opens another row to keep all objects visible. Objects are actually displayed within a grid of fixed cell size, so names or titles are truncated with appended ellipses as needed.

Items occupy locations within the workspace, so a drag-and-drop to a particular position or a paste with the cursor positioned within the workspace leaves the object at that spot, moving the other contents to the right and down as needed. Likewise, items within the workspace can be rearranged by drag-and-drop or cut-and-paste, making it a completely flexible working area for organizing the material of interest. A drag-and-drop to the frame or tab of the workspace or a paste from the clipboard or an “Add to Workspace” without the cursor positioned within the workspace simply places the object in the next available location to the lower right within the workspace.

## Remarks

The dynamic workspace is a simple mechanism for facilitating a common task: document assembly. Although the particular version of it described here was designed to serve the specialized needs of classroom teachers working in the particular environment of a performance-support application for classroom information management, other adaptations of the general concept are easily devised. In fact, a fully general-purpose mechanism is not difficult to envision.

The most recent versions of Microsoft's extended clipboard approach this capability but with somewhat less efficiency and elegance. If the user must continually move a floating clipboard tool out of the way or must manually open a docked clipboard, the task of document assembly is made more difficult. Although the Windows Task Bar allows for keeping multiple source documents open, integration of document navigational features with a workspace not only enable users to move easily among documents of interest but also to return to the source of materials collected. Anyone who has compiled a set of quotes and forgotten to keep track of where some of these came from knows what a challenging task it can sometimes be to recover this information even mere minutes later.

Recognizing document assembly as a generic collection of tasks and designing a common mechanism for supporting it offers the potential for making the everyday work of many of us much easier in the future. Sooner or later, somebody will do it. We're ready to help anyone who is ready to try!

## References

- Constantine, L. L., & Lockwood, L. A. D. (1999) *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Reading, MA: Addison-Wesley.
- Constantine, L. L., & Lockwood, L. A. D. (2001) "Design Study 1: Active Table-of-Contents Control for Content Navigation and Customization."  
<<http://www.forUse.com/articles/designstudy1.pdf>>

**Learn more about usage-centered design at <http://www.forUse.com>.**