Webtop: Realities in Designing a Web-Application Platform

David Heller
Documentum
6801 Koll Center Pkwy.
Pleasanton, CA 94566 USA
david.heller@documentum.com

Lisa Krenzelok
Documentum
6801 Koll Center Pkwy.
Pleasanton, CA 94566 USA
lisa.krenzelock@documentum.com

Julian Orr
Documentum
6801 Koll Center Pkwy.
Pleasanton, CA 94566 USA
julian.orr@documentum.com

Abstract
This case study focuses on the design process for a thin-client in a real world enterprise software environment, created for our own internal sales and marketing directives. This project became the basis of our biggest upgrade migration in four years. The project increased sales and earned our product industry awards, but was most successful for the paradigm shift in corporate culture; specifically, user experience has become a fundamental part of our development process, a challenge in most organizations. Developing this client as an interaction platform facilitated the design of applications based on the Documentum platform and increased client consistency throughout the Documentum product line.

Keywords
Web Applications, User Experience, Interface Design, Platform Design, Design Process, Document Management, Content Management, Enterprise Content Management (ECM), Enterprise Software, Web Content Management (WCM), Web Publishing, Workflow Management, Accessibility, Compliance, B2B, B2All, Documentum, Library Services, Thin-Client, User Interface, Interaction Design

Industry/category
Webtop (the product) is both a development platform and an Enterprise Content Management (ECM)

Figure 1: The application architecture stack of Webtop.

The combination of technologies comprising *Documentum 5* was chosen to match these customer requirements:

- Work across many application servers, database engines, server platforms
- Allow easy deployment, easy customization, cross-platform/browser distribution
- Use technologies that are widely used within our developer community

application that is used across vertical and horizontal industries in the enterprise among the Global 2000.

Project statement

Documentum is the leader in enterprise content management solutions. Our generic tools are meant to facilitate the management of digital content through its various business process states: creation, production, review, editorial, approval, audit, validation, and consumption. The tools also manage the relationships between various types of content through business processes, so that relationships are maintained from beginning to end, for the purpose of way finding as well as publication.

Consider the manufacture of an airplane and the digital content needed to expedite that process. In addition to the documentation associated with each subsystem, each part within that system has documentation associated with it as well, whether it includes specifications about the object or specific instructions on how to use it within the manufacturing process. The relationship between the subsystem and its parts is preserved while both documents go through the business process states associated with writing (review, editorial, and so on).

- For the last two years Documentum has been working on its latest innovation, a new development platform and application set that is called Documentum 5. At the core of this project is the development of a Java 2 Enterprise Edition (J2EE) application layer and a Java Server Pages (JSP) user interface layer. These technologies are used to build a DHTML (HTML, CSS, and JavaScript) visual display layer.

This would allow for the process states of any process to be accessible more easily throughout the enterprise. Our team's contribution was centered on the user interface. Originally, the backend portions of the product were managed separately from the front-end pieces. The backend (the platform) is known as the Web Development Kit (WDK), and the front-end application is called Webtop, which uses the UI components of the WDK.

Webtop is a fully functional, completely customizable, Web-based ECM application. The goals presented to the team for this project included:

- Matching the functionality of our desktop application by 90%
- Developing repeatable UI components for Webtop to be used in our other Web-based applications and thus, creating a model for UI convergence and consistency
- Creating a UI that is completely I18N, L10N, M16N, and Section 508 compliant
- Creating an engaging and flexible UI that excites our customers
- Creating a more usable interface that helps users be more immediately effective
- Creating an interface that had a "wow" factor, or at the very least would not cause a negative reaction

The project started in Q3 2000 and our first release was January 2003.

Project participants[1]

Design Team

David Heller, lead Sr. UI Designer
Lisa Krenzelok, Sr. UI Designer
Julian Orr, UI Designer and Usability Engineer

Engineering Team

one application architect
one managing engineer
one principal engineer
More than 10 engineers

Product Management

one product manager

Quality Assurance

one quality assurance manager
More than 6 other QA personnel

Other in-house stakeholders

Other client teams
Corporate Marketing
Sales and Consulting

Out-of-house stakeholders

System Integrators (Partners)

Documentum has a very large network of system integrators who implement the systems our customers buy.

Customers

Our customers are Global 2000 corporations that cross all vertical markets and horizontal needs. The end user is *not* their primary concern, though it is a growing one.

Users

We break our end users into 4 categories:

1. Administrators. Administer the system. Create custom objects, set user roles and privileges. Create workflow templates, administer workflows, create lifecycles. Administer security.
2. Managers. Administer workflow, assign tasks, create templates, contribute content, approve content, push object through the lifecycle.
3. Contributors. Create content, review content, edit content, use tasks, and push object through workflow.
4. Consumers. Read content, with minimal workflow and editorial requirements.

Project dates and duration

- Project started Q3 2000
- First public presentation of a UI made at our user conference September 2001
- A pre-alpha began in March 2002
- Our beta began in June 2002
- Controlled release was September 2002
- Release was January 2003
- Next versions are already in development. The overall design process is ongoing, and we have been able to contribute changes in each iterative release during the last two years.

[1] Executives, Technical Publications, Beta Management, and Project Management were all consulted with regularly, but played a secondary position compared to the rest of this team.

Our general Product Development Process (PDP) starts once a business requirement is identified and validated:

- A product manager writes the marketing requirements document (MRD).
- A product architect is responsible for the functional requirements document (FRD or functional spec).
- A lead engineer is responsible for a design (engineering design) spec, which is not to be confused with a user interface (UI) specification.
- A quality assurance engineer composes a certification requirements document (CRD) and a set of test cases based on the MRD and the functional specification.

Process

Documentum's history is built on old school server technology methodologies. As is often the case, user experience design was not a focus of our development efforts until recently. Also, we continue to be a very marketing driven company, with the attendant compromises that that might entail. Webtop was the first application where the User Experience Group (UEG) was given an opportunity to prove the value added by user centered design (UCD). As our first attempt at this process within our existing corporate PDP (see sidebar), we were limited by our own learning curve with these methodologies and by the education of and response by the non-design team.

Many of the espoused design methodologies begin at the business requirement definition phase. Our design process could not be said to have followed any one UCD technique. Our approach to design could be called user *oriented* design, in that we always sought to contribute to the greater good of all our users in our designs. A series of site visit interviews were conducted with local customers using a variety of our applications. The idea was to observe users according to the Contextual Inquiry method [2].

The structured results of our observations led to the creation of personae using the Cooper methodology[1]. Notes and personal experience were combined to create a family of personae that represented the users for whom we should be designing.

In tandem with this operation there began the creation of a modified version of a User Environment Design (UED). This allowed us to do "sanity testing" on the grouping of functionality before proposing a UI design.

Design from this point was primarily an internal process: logical grouping of functionality, designing behavior, laying out components. We attempted to be as conscientious as possible during the process, balancing the stated need of porting our desktop application to the Web, with the opportunities to push the limits of the user experience.

Communication among team members emerged as a critical issue. With the development team based in England and the Architecture/Design team based in California, a great deal of formal effort went towards developing communications protocols.

At this point the design process took on a new track; we would wait for the lead designer to come across a design impasse or wait for an existing design to be rejected by the project engineer or architect. The team would then assemble and the informal design process would begin. The team consisted of the four to five members of the group taking on different advocacy roles:

- UI design—the user and application consistency
- Architect—scope
- Product manager—timeline and marketing agenda
- Managing engineer—resources

When certain issues could not be resolved, we resorted to guerilla user tactics. These tactics included involving customers who came to our campus for training. We would bring paper prototypes down to the training rooms during breaks and do quick sessions with these users. The results helped us with some of the tougher and more urgent design decisions.

With this knowledge, we prepared a build of the product (very limited functionality) that we could preview at our upcoming user conference. The attendees of these conferences leaned heavily towards specific personae: administrators and developers. This validated our earlier assumptions and proved our methodology was indeed useful. When we had a more stable and feature-complete product we were able to run more usability tests to validate some of the design areas where we still had doubts.

Finally, we had a beta test period where a core set of customers was able to install the application, customize it ,and put their users in front of it. A beta manager tracked issues formally using our customer response systems. These were made available to the team.

The process was able to validate our previous design decision regarding the Streamline view [Fig. 6]. In the lab we saw different users choosing streamline over classic, and vice versa. We also found a visual design flaw in our page design. Having a page bar as a solid bar at the bottom confused our users into expecting an added frame under it (as exists as an option with our desktop application [Fig. 2]), so in the released product we removed it. (In tests since the final release, we discovered that the page bar is still required, but that its visual design needs to be less of a framing border.)

Solution details

How does the solution support the project requirements?

Introduction

To take on all the requirements in this study would be a massive project. As listed previously, there are many project requirements and too many specific goals.

Without going into too much detail, we hope to explore what it means to have so many requirements in a single interface. For example, where most applications can take the 80%/20% approach when doing design, we purposefully designed from the point of view of the edge case scenario, and not from the point of view of the least common denominator. While we created an out-of-the-box solution, it is a base rather than a final product. It can be used in its base form, but 90% of our customers will change the interface to suit their needs. This is a primary selling point of our products, and one of our goals was to make sure that customization happens as easily as possible.

Matching the functionality of our desktop application

This is actually a pretty lofty goal—create a thin-client to match the functionality of a thick-client.

Working in a cross-platform and browser environment, there are tremendous limitations: no consistent support for drag-and-drop, no hot-key support, no means of communicating between the browser and the desktop's operating system, and a limited number of input and navigation widgets at a developer's disposal.

We decided to match a desktop metaphor because the scope of the functionality that our application has to support required it. We needed to have a menu system in order to support our over-20-actions-per-object type. We also needed to have multi-object selection and act on those selections simultaneously.

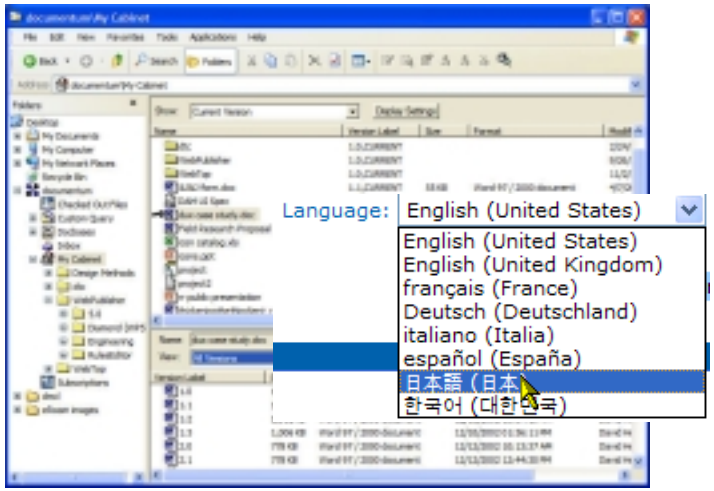

Figure 2: The desktop application that we based Webtop on.

Webtop was supposed to match the functionality of our "Desktop Client" (DTC; our thick generic application) by 90%. Our generic clients are three tools in one: a library services repository system, a workflow manager, and a business processes tool. They are also very customizable.

We fell short of this goal for two reasons: lack of resources and lack of technology. The former is a universal problem in software development. The latter is a particular result of the Web environment. A desktop application can have robust communication with the desktop, such as drag-and-drop between the application and the desktop, or use the clipboard functionality of the operating system. A Web-based system cannot yet do this without significant help from added applets, which then raise other issues.

The largest issue we found was that most of our users, at some point during testing, tried to use the Web browser's menu bar instead of the menu bar for our application. There are other browser components such as the back button and address bar that also distract users from our application.

Repeatable UI components we developed for Webtop to be used in other Web-based applications

Since our customers think of our applications as a development environment, they expect the same type of objective control they get in other programming environments. To this end, our application had to have repeatable controls and containers that were then themselves extensible.

One of the considerations is that the UI was one layer of a multi-layer system and often our ability to componentize was limited by the layers we laid on top of it, as well as the core server technology that holds everything together. We created such a robust system that our control applications, which are to be released using this platform, were able to upgrade their systems and port working applications in 25% the time it took them previously. This is probably our biggest selling point.

The effect on the design was very positive. We were able to force much more consistency throughout the application through these components than we were in previous versions of our Web-based application.

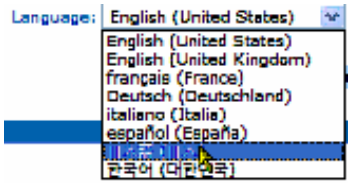

Figure 4: The list of languages are written in the language that the user would be selecting in. We initially had it in the language of the currently selected language, but found that if a user accidentally selected Japanese, for instance, they then couldn't find their way back to English.

Creating a UI that works internationally

Figure 3: Internationalized login screen

The components of our parent layer just made this part easier. There are no graphic text controls, so all text is live, and all text is built on top of a data-dictionary that gives an almost singular translation source. We ship in five to six locales; we are UTF-8 and Unicode aware. Our login screen demonstrates this interoperability very seamlessly.

Creating a UI that is accessible (508c standards)
(This requirement was added mid-way through the implementation process.)
Accessibility standards have been developed by the W3C (WWW Consortium) and dubbed 508. They include the use of "alt" attributes for images; the use of consistent and usable tab orders, especially when using frames like we do; and the use of proper labeling of data grids and navigation elements.

To attack this problem, we made an interface that is filled with text controls instead of image buttons, which helps. Through a preference setting, we add features that aid screen readers. We recently ran a test with blind people who were able to use the system faster than we could. The screen readers found flaws we couldn't find, but for the most part, they were a lot more satisfied than frustrated, which is a great success.

Create a more usable interface than its predecessor
Quality of the user experience increased tremendously based on customer response during testing, demonstrations, focus groups, and other customer gatherings. There are probably three key areas where user experience was exponentially increased for users: ability to do multi-select, a better menu system, and an additional Web-like interface.

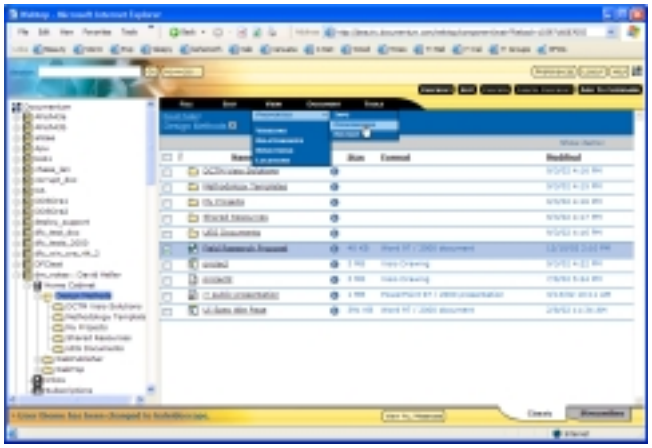

Figure 5: Webtop Classic View

Previously, in our Web-based application, we had a very small set of inline actions. The user had to go to a screen where all available actions were located. This required a refresh of the screen (round-trip to the server) in order to even see the list of possible actions. Now, using a DHTML system, we have all actions, both enabled and disabled, available for users.

Where it made sense, we added controls that allowed users to select multiple documents. Checkboxes are located next to each line-item, and actions for those items, when enabled, are in either a toolbar for quick and persistent actions or in menus.

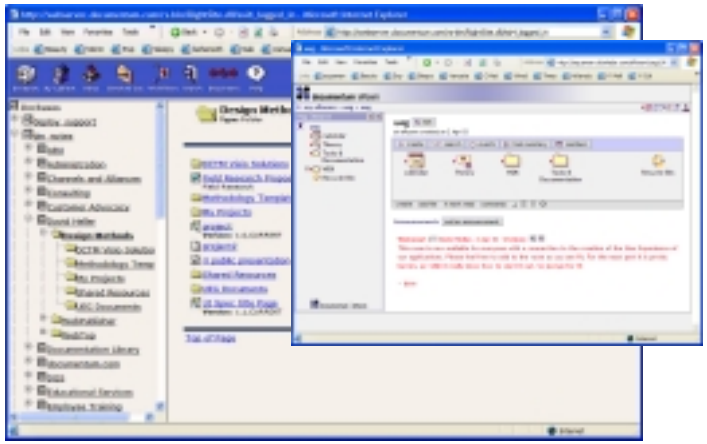

Figure 6: Intranet Client: Previous Web-based thin-client

Having disabled actions visible is an important aspect of the system—it gives users cues to the system's logic, it creates a consistent set of markers that users can use to help location recall, and it teaches users about functionality in the system they might not have been aware of. The most important part of the menus is a user's access to actions without requiring the system to go back to the server and causing delays and frustration.

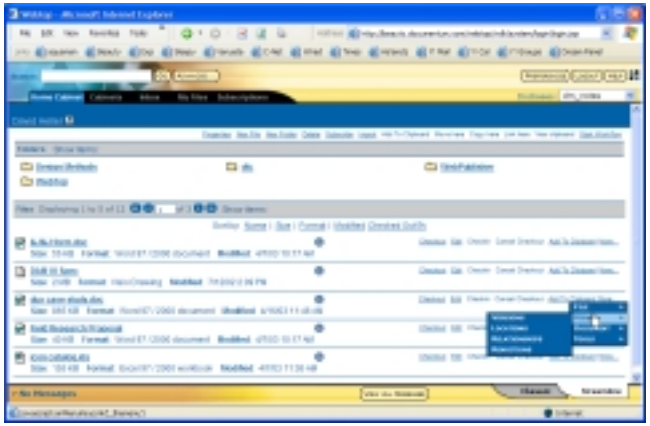

Figure 7: Webtop Streamline View

Based on a combination of customer and executive requirements for wanting a "Webby" version of the application, we created an alternative view of our product. Our Web-like interface, which we call "Streamline," takes our "Classic" view, with all the same functionality, and displays it in a Yahoo-like directory structure, which can only use inline single-select actions. The Streamline view also has a narrative display of data, as opposed to the data-grid of Classic view. A comparison to the Detail and Tile views of Windows XP can be made.

Merely through a simple designer-initiated change in the UI, we shook the sales and marketing team's view of the value of the product.

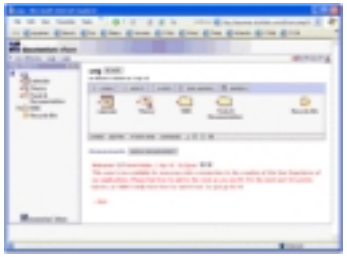

Figure 10: eRoom screen using the same skin treatment of font and color as the eRoom skin in [Fig. 9].

Another difference is that in our Classic view, the primary form of navigation is an explorer tree model. The Streamline model is based on a bread-crumbing and drilldown system. Originally, our personas suggested that a novice user would prefer Streamline and an advanced user would prefer Classic. However, there are efficiencies in both systems and there is no correlation between comfort with technology and a user's choice of navigation mode. Lastly, the Streamline view separated out container objects (cabinets, folders, categories) from the list of leaf objects (files, content). By treating containers differently from leaf objects, we helped users focus. In Classic view, the ability to focus on either leaf or container is achieved through a filter.

Raise customer perception that we take usability seriously as a company

From a customer's perspective and, for that matter, an analyst's perspective, "What is usability?" To be quite honest, it is "lookin' good" that matters and not much else. We had to then improve the surface look of Webtop.

An early technical requirement for the system was to have the ability to change skins of the application. Many of our customers wanted to use their own brand with the application. When we did testing on the first skins we created, customers were excited about the new functionality but not the application itself. A graphic designer created a set of skins that were much more engaging. Once these skins were added to demonstrations of the product, there was a dramatic, positive response to the application. The glitzier new look had to be supplemented with more usable skins that were less distracting.

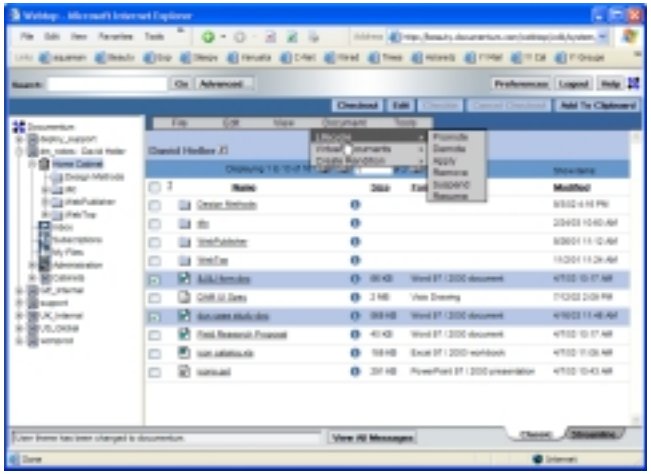

Figure 8: Our first skin never got the same response as our new default skin (see Figure 5).

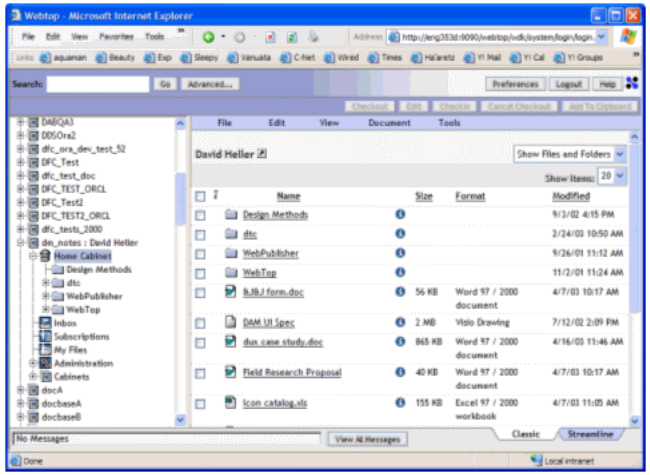

Figure 9: Our eRoom branded skin. It is both a new brand and with less "glitz" than the default skin (see Figure 5).

Addressing the reality of a cross-browser, cross-platform environment without any GUI applets, while at the same time trying to match a desktop behavior set, is very challenging.

What were the elements of the design strategy?
There were several elements to our design strategy, each addressing various pieces of the application and platform that we were designing.

Be an app first before being a Web app
There was actually debate about this component of the strategy. Many felt we shouldn't hide from our "Web roots," instead, possibly expose them proudly. This segment eventually was molded into our Streamline view [Fig. 7]. But at the core of our design is an attempt to be a desktop application that happens to be asynchronous, and that runs in another desktop application.

What does it mean to be an "application"? In this case we went with the 80/20 approach to expectation. Users expect a Windows Explorer interface that gives them the added functionality of our library and business process services. We looked to Windows GUI conventions for assistance, but they themselves are so inconsistent that it was often up to us, through our own heuristic analysis, to come to our own conclusions. Still, the menus, toolbar and explorer tree were all designed with this in mind.

Consistency
Consistency is one of our strengths. While developing this product in tandem with our suite of core-controlled applications, we had to constantly iterate new behavioral and visual patterns. Here is an example. What happens when you click on an object name link? What do you expect to happen? Is it true in all cases? Is it really what you want all the time? We opted to ruin the instant gratification approach to usability and exchange it for long-term learnability. Every time a user clicks an object name that is a link they view the content of the object.

Other areas where we wanted to be consistent were in the creation of widgets: choosers, repeating attributes, property dialogues, create object wizards, etc. We tried to focus on these core elements and worked together to make a general set of guidelines as opposed to designing each instance completely separately.

Give the user information
When it comes to complex software that is based on enabling users to complete tasks as part of grander business processes, you have to give the user information in order to make decisions quickly. This is one area that we feel we can use some definite improvement, not just in implementation, but also in design. For us, this becomes apparent in the designs of error management and metadata visualization. During object listing we have enough space to handle this, but in places where we are doing object selection and need more space for GUI controls, we have not designed well how to put enough of the information on the screen so a user can make an informed selection.

Modality done without pop-up windows
Applications have pop-up dialogues. They are a natural part of our desktop application universe. We accept them gladly. However, in the Web world they are so scorned that it is considered heinous to accept them even on a small scale. We decided that our system would not have any pop-up dialogues.

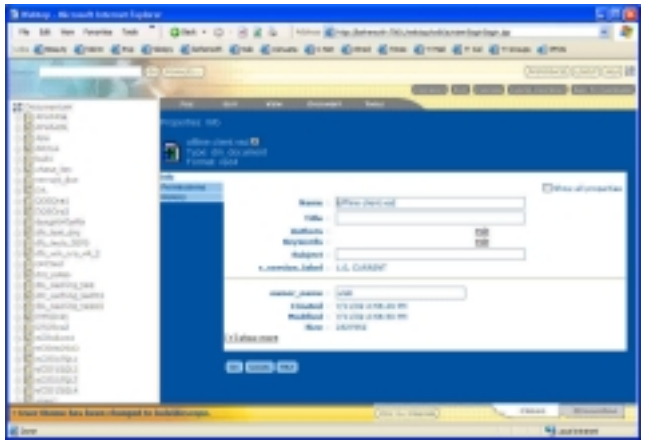

Figure 11: Properties (modal): Navigation is grayed-out.

The reasons behind this decision are technical as well as user-based. Regarding the technical reasons, the best way is to address them in a Q&A session, beyond the scope of this paper. The design issue comes down to the “app inside an app” problem. Every new window gets its own taskbar item, and that confuses users. We also felt that in the Web world, users aren’t used to Web interfaces spawning dialogue windows, and there was a lot of argument over what could be deemed a dialogue and what was primary focal area. We couldn’t find anything in the literature to support our positions for creating the windows, so countering the technical limitations as well as the conservative approach was difficult.

We created modality through DHTML. We wanted to prevent navigation in places that would cause an implicit cancellation of an action. We wanted to make actions finite in scope so that a positive or negative action was always made explicitly.

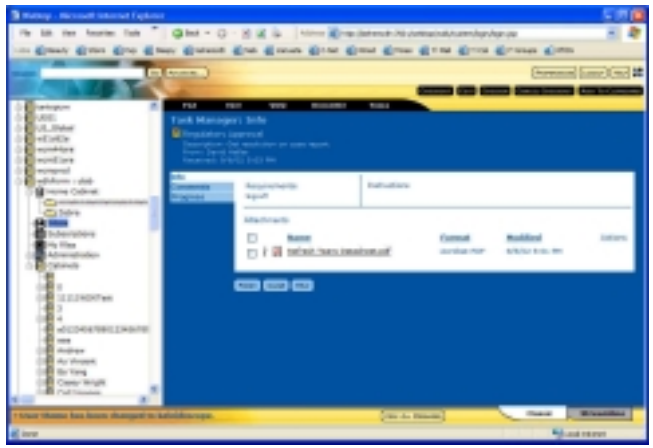

Figure 12: Task (non-modal): requires acting on objects.

Because the contents of a task [Fig. 12] include objects that need to be acted upon similarly as if they were in a standard file list, the container could not be modal.

Describe what is unique or convention-setting about the user experience?

What is convention-setting here is the creation of an application that is also a platform for other applications. Using the component architecture and visual vocabulary that was designed in Webtop, we have now built four other controlled applications based on what is displayed in Figure 13. This primary goal forced our design in directions where it might not have gone. We had to turn the 80/20 rule on its head and design towards complexity, while never ignoring the more simple and probable uses of the system as a whole.

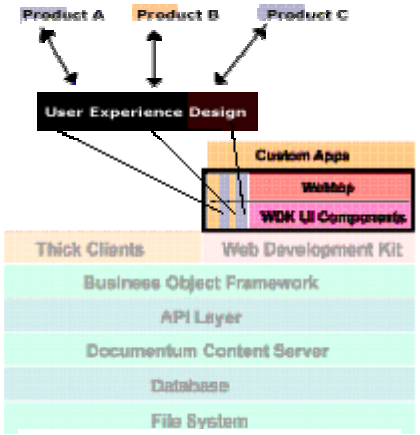

Figure 13: Centralized design of the new platform allows all applications to inherit one coherent design.

What were the constraints of the solution?
Time, budget, resources, technology, and disparate goals between groups all played a significant part in creating constraints. We would add to this an engineering team that was eight time zones away, that had its own culture, and where the engineers were traditionally designers.

The combination of improper staffing and inadequate budget was probably most responsible for constraining the visual design. We do not have an in-house visual designer. We did eventually hire a visual designer to do a few of the skins we released the product with.

Time also drove this as well as other factors that differentiate design from implementation. There was a lot of hope that went into the design, but that hope was often squelched by real world deadlines that the engineering team had to meet. In the software industry, releases are tied to quarterly reports. Thus, this often forced a one to two month push on our deadlines. This caused stricter triage sessions where scope gets greatly reduced from original expectations.

The other place where time, and probably resources, hit us is in upfront pre-design research. We are lucky to get any time to do this kind of work. The team is usually on three to four projects each, which means there is no “off” time where such research can be conducted.

More importantly, we are a new group, and so no time is available for standards development. We have no standard recognized visual vocabulary for all groups, and there is little time for the creation of a standard methodology for doing research—lab or field. What research we are able to stuff into our available calendars is very rough, and we take what we can from it. Karen Holtzblatt said at CHI 2002 in Minneapolis that it is better to get any time with a user than none, so we did just that.

The real problem is that our efforts are often not coordinated into the project plans of the other teams. A change request is almost always turned down, relegating usability testing and other measures to second release requests instead of into current project timetables. While most of the groups are unaffected by this type of delay, it affects the design group greatly because we are always left feeling like we didn’t do the best we could have done.

Technical Constraints
Client Environment: IE 5.5+, Netscape 6.2.3 (no 7 support), Java VM (MS or Sun), OS (Windows 32bit, MacOS 9.2 [only Netscape, only Sun Java VM]), Screen Resolution-800x600 (no horizontal scrolling for default views).

Application Constraints: No unbounded display sets (pagination required for display sets over 50 objects), 508 (accessibility), and internationalize and localize. There are no GUI applets (Java, Flash or ActiveX), but only one applet that is responsible for writing registry information to the desktop from the server.

How was business and culture affected?
The biggest effect to our business was to shock our customer and user base. They are still in shock from seeing the new look-and-feels we showed them upon release. To truly understand this effect, you would have to see what our other applications look like [Figs. 2, 6].

Awards and analyst recognition:

The Association of Image and Information Management (AIIM) 2003 Conference: Best in Show; Best Enterprise Content Management Suite.

Transform Magazine Product of the Year 2002 (Jan 2003): "With the September release of Documentum 5 … system functionality was placed behind cleaner, role-based interfaces."

Giga Information Group: "Documentum 5 brings significant enhancement ease of use and ease of access."

Another major effect was the creation of a real visual platform. We now have a new development business model centered on this component architecture, which allows all future applications of the company to be built on it. This has meant that designing the second level applications is that much easier, because we already know the visual language, as do our engineers. On the negative side, we assume that this new framework is appropriate for all of our applications regardless of user requirements. This means we cannot expand on or change our component architecture because we rely on the existing framework to reduce total "cost" of development.

What was the feedback/user response?

Feedback has been both anecdotal and research oriented. We hear from the field from our sales and consulting groups all the time, and we had a release preview at our recent user conference.

Users have been thrilled on so many levels with the new application. But the answer here is more complex. Since our customers come from both desktop and Web applications, their responses differ. The Web-based customers love the new interface because it gives them many more features than the predecessor, it is considerably more intuitive to them, and they also have a better platform with which they can extend their current applications. The desktop users, when first presented with the new interface, are a bit slower to warm up because they are focused on their losses as opposed to their gains. Over the long term, though, Webtop is a lot more stable and runs a lot faster than its desktop cousin, and users figure this out fairly quickly.

For both types of users, one thing is true—they seem to treat Webtop as if it is a desktop application, trying to use interactive behaviors like drag-and-drop and right-click that don't usually exist in standard Web-based applications. The appearance of a navigation tree and menu and toolbars hints at the robustness of a desktop application, and users seem to have greater expectations of the system. This expectation is not as great when users are in "Streamline View."

What was the impact on you or on the end user?

User Experience Group

The impact on us has been tremendous. The main impact has been that the company has a new respect about and understanding for the importance of user experience. It is the "proof of the pudding" that we needed to get on the menus of our executives.

The process was also the beginning of creating a new corporate culture where design is singular across applications. Because this is our new platform for development, it means that the same group that controls the design of this platform controls the design of all the applications made with this platform. We now have a new, centralized control as the User Experience Group that we have never had before.

End Users

The impact to end users is harder to judge. There are so many types of end users for our systems that the gauge is very different for each one. Also, because the system is recently released, end results won't be known for quite some time. Rollout still takes a long time, and many of our customers have strict implementation

procedures, so while sales are affected immediately, the end user doesn't see the product for awhile.

Results:

How did you measure success?

Sales

The most important measurement of success is increased sales, or at least increase win rates when pitted against competition. In this regard we are seeing good results, where customers are actually citing our UI as a main contributing factor as to why they are considering, or choosing, Documentum over the competitor.

Reduced development time

One of our core applications, Web Publisher, was able to build its first release using the same platform as Webtop, in record time. With this we are able to release a new product line, have it look and act with the previous product in record time, even while a big bulk of the production was done with parallel code lines.

Testing with other applications

Even before Webtop was released, we were building our internal applications on its platform and had beta and usability tests on those applications. Our beta survey for our Web Publisher application, for example, showed that customers saw a tremendous improvement between this application and its predecessor. Usability testing and focus groups also had similar results, where most tests ended with the phrase, "I need this yesterday. When can I get it?"

What insights were gained?

Internal Environment

Instituting design standards at a small but growing enterprise software developer is slow going, but it does progress. It is good to design idealistically, but reality often wins out in the end. Learning to pare down and compromise designs in certain areas in order to meet resource and marketing requirements is a key and valuable skill for any designer. There are key moments where you realize, looking at a user in a lab, that it is just impossible to be all things to all people.

Communication is key. Better documentation is needed to make projects go better—especially bridging the gap between interaction design and final implementation. Have non-design allies. Get feedback from individuals who can then advocate with you for your design before publishing specifications to an entire team.

Customer vs. User

Never underestimate the power of the word "Internet" or "Web." Too often our design needs are determined by Internet hype and not by the needs of the customer or the end user. There is a big difference between a customer and an end user. Their concerns are not at all the same and are sometimes in conflict with each other.

What a sales person needs to make a sale isn't always the best thing for a customer or a user. A sales person needs something that flashes in a demo. If it doesn't have a bell or a whistle here and there, they don't have talking points to focus on for their sales pitch. Functionality is also determined by common customer checklists, more so than by what would help real users

accomplish known tasks and make business processes more efficient.

What is repeatable, and what would you do differently?

Repetition isn't always a goal. We are evolving our process, which means change is our focus at the moment. We should spend more time documenting our designs to improve efficiency in production. The goal here is to get our designs done to our expectations with as few iterations during development as possible.

What are the next steps in the work, anticipated growth, and development of the study's concept?

We took the back end of this system and made a componentized system of it. While the attempt was made to truly componentize the front end, we really didn't succeed. On a similar note, we didn't give the visual layer of our front end enough control so that builders of skins could have more variety and actually improve the information display layer. The next revision is meant to do the latter, and, hopefully, the revision after that will take on the former.

The development of a GUI guide is a major necessity. This will help communicate with engineering and quality assurance. We are revamping some key components due to customer feedback and due to additional functionality being proposed by other divisions. We also have acquisitions that are adding functionality to our corporate library of resources, and these need to be assimilated into the Webtop/WDK framework.

Acknowledgements

Applications

Outlook was a key application for us. Its combination of views, navigation methods, data types, and object manipulation model was a good match for our needs. Yahoo was the biggest influence in terms of two areas: streamlined navigation style and its Web-based applications. Hotmail, as well, was also influential.

Theories & Theorists

Alan Cooper, Nathan Shedroff, Karen Holtzblatt, Jesse James Garret, Donald Norman, MS Inductive Interfaces Guidelines.

References

[1] The Inmates Are Running the Asylum, Cooper, Allen & Saffo, Paul; Sams—1999.

[2] Contextual Design, Bayer, Hugh & Holtzblatt, Karen; Morgan Kaufman—1997.