The Office 12 User Interface System

Jensen Harris, Office User Experience Team

Draft 2: December 14, 2004

Office 12 is centered on a new user experience which enables users to be more successful finding and using the rich functionality in our products. Early feedback shows that users are delighted by their new-found ability to browse and discover new capabilities in our applications. Much of the power in our products remains undiscovered today, buried behind layers of UI noise fashioned over two decades of feature growth. Data collected through SQM show that most people only use a handful of basic features, despite their desire to achieve more with our products. ***Our key opportunity is in exposing the power that's already present in Office, unlocking the years of innovation that are just waiting to be discovered.***

This document details the philosophy behind and details of the new Office User Interface System. It is designed to give you both an overview of the components of the system and how they work together as well as a detailed look at how each piece works individually. Although this document is complete, it in no way communicates the depth of the system as well as actually using it.

Office Today: Powerful, but Underrated

The current Office UI system of menus, toolbars, task panes, and dialogs has been suitable for providing access to the wide variety of features in Office. At the same time, most people, even those who use Office for hours each day, don't feel comfortable using our product. They have dug out little paths of productivity—5 or 10 simple features that everybody needs to use and perhaps another 5 that someone showed them how to use or they've read about in a book or on a web site. SQM data indicates that most people use a small number of "standard" features and a few of "secondary" features. This, despite years of adding features which solve customer problems to our products. Why don't we see broad adoption of these features?

Competitors have an easy job; as long as the powerful core of Office functionality is never discovered, they can implement 10% of our features and sell it at a fraction of the price. Customers, most of whom can't tell the difference, have started to migrate to these cheaper alternatives. Yet, we know our product is leaps and bounds more powerful and would allow customers to create fantastic documents and achieve better results with less time—if they only discovered the possibilities of Office.

The major reason users don't discover the power of Office is the sub-standard user interface in today's product. Designed for the limited, command-oriented world of programs introduced nearly 20 years ago, the user interface has failed to advance despite the improvements to nearly every other subsystem of computer interaction.

Figure 1: Word 2.0 (1988), Word 2003 (2004): What other part of using a computer has remained stagnant for so long?

Word from two decades ago is a totally different application from the Word of today. Word 2.0 supported around 100 commands, and the menu/toolbar structure did a good job of communicating the power of the application. There were only a few menus more than one level deep; a quick browse across the top-level menus revealed 95% of the program. The scope of the program was eminently understandable and running it even today feels in some ways like a breath of fresh air. You quickly achieve a sense of mastery over the program.

Word 2003, by contrast, contains about 1500 end-user commands[1]. Word 2.0 contained two toolbars; Word 2003 contains 30 top-level toolbars and another 25 task panes, in addition to several times as many commands in the menus. Where 1980's Word was svelte and graceful, 21st century Word is a bloated pig of mis-organized and forgotten functionality. We've made two decades of progress in improving the capabilities of our word processor, but the UI, bursting at the seams, was designed to handle programs of a different, simpler era.

Users have no idea where to start exploring our programs. Think of all of the places functionality is scattered: many levels of menu hierarchy (don't forget to explore both the "short" and "long" menus), legions of toolbars in all different places scattered throughout the product (some on and some off, some "rafted" and some floating over your document), the stack of task panes. We never clean up after ourselves; we just add and add and *add and add…*

There's no way to create a coherent mental model of the Office applications because they are too complicated and too strewn with incongruities. Two decades of arbitrary design decisions by developers and program managers reflect generations of user interface philosophy and values all combined into a single Frakendesign. It's no wonder that today's users learn to use a few features and shy away from exploring more of the product; we haven't created an environment conducive to end-user success.

Today's UI isn't "good enough"—it's terrible. There's nothing magic about menus and toolbars—both models have significant usability and discoverability problems that are inflamed severely by the fullness of them.

While the "founding fathers" of Office were right about the applicability of menus and toolbars to basic tasks, in practice, they don't work very well for the powerful and varied functionality in today's products. The only thing menus have going for them is their familiarity—but it is this very familiarity which causes users to overlook new features and ignore the potential of our products. It's only through the grace of a best-of-breed feature set and a great core set of functionality that we've stayed competitive. But it's no wonder that it's almost cliché to hear customers say "nothing in Office has changed in the last 10 years."

The time is now to cash in on the incredible potential of our applications. The parade has marched on in some many other areas of technology; why must Office—so crucial a part of so many people's day—be left behind?

Office: The Future is Now

The new UI framework for Office provides a system of user interface controls, visual design, and behaviors that is optimized for today's feature-rich programs. Widely applicable to a variety of different program tasks, the new user experience helps users discover and use the powerful functionality already in our products. It creates the runway for another decade of innovation on the Office platform.

What would our original architects have designed if they had anticipated the scope and breadth of functionality in today's product? Assuredly, it would not be what we have today. In that

[1] The total number is closer to 1900, but that includes all of the AutoShapes, cursoring functionality, and a few debug commands.

same manner, we need to look ahead and anticipate the kind of growth our products will experience over the next decade—to imagine the ways people will want to work moving forward, not just right now.

The new UI is not simply a single control or concept. It is a web of controls, design philosophies, and interactions, calculated from inception to work together. Converting an application to the new UI is not about porting menus and toolbars to some other control or adding some new pane to the screen. The new UI is a philosophy and a set of tools that need to be applied to an application's soul. More an art form than a science, each program can use the new UI to communicate to the user what it is best at. The new user experience leverages the advantage of our depth in program management and design skills and doesn't lend itself as easily to the kind of hollow cloning currently being exercised by our competitors.

It is important to the future of Office that users start to feel a sense of mastery about our products. Office is perhaps the most complex and powerful tool that the average person uses in their day-to-day life. Certainly more powerful and multi-purposed than a car, a television, or kitchen appliances, Office is used by most workers in a longer and more intimate way than pretty much anything else in their life. Any strides we make towards improving this customer relationship, through making our capabilities more transparent, will result in unbridled enthusiasm from our faithful.

The new user experience will get a new generation of users excited about Office; the UI's ability to put the user in control *is* the feature. This is our chance to start fresh and surprise the world with what they can achieve—with out help.

The Office Design Philosophy

The new user experience was created and validated against a cohesive set of design philosophies. Research done by usability and design indicated that a user experience adhering to these principles would provide a very positive customer experience. As a result, we've consistently validated the plan of record against these tenets to ensure that we are true to our core design values.

- The user's focus should be on the content, **not on the UI**. Help the user **work without interference**.
- **Reduce the number of choices** presented to a user at any given time. Increase the user's sense of mastery of the product by **contextualizing as many commands/properties** as possible. Reduce the command space by **eliminating** redundant or seldom used features.
- **Increase efficiency**. A small gain in "scope of features used" isn't worth a significant loss in "efficient use of the features."
- **Don't try to solve all problems the same way.** Any user experience that simply maps the existing command structure to a new UI paradigm won't be an improvement; it will just be different.
- Give features a **permanent home**. Prefer consistent-location UI over "smart" UI.
- **Straightforward is better than clever.** Office has a long and rich legacy; any successful UI innovation must feel familiar to the user.

We believe that by designing a system of UI that follows these principles, we will restore end-user enthusiasm about the product. We will help people create fantastic documents and share stunning presentations. We will enable a more efficient, yet familiar, user interface which will save people time and help them get more done. Users will gradually discover more of the product, accomplishing things they didn't think was possible in Office. We will cement the worth of our product in customer's mind as a premium offering—one differentiated from the decrepit clones threatening to lower the bar of what's possible.

We will create a product that is fun and simple to use.

The pages that follow outline the details of the new user experience, explaining each part of the system and how it works together. Each of these elements together comprises what we call the *Office 12 User Interface System*.

The Office 12 User Interface System: Detailed Design

The Ribbon

The heart of the new UI is called the "ribbon." The ribbon is a region at the top of the screen that presents the main set of commands in a highly browsable manner. Each application is split into a set of tabs which represent the main functionality groups of the program. Much like a well-organized restaurant menu splits the selections into sections (Appetizers, Salads, Beverages, Desserts), the ribbon exposes the organization of the application by section (in Word's case, perhaps Writing, Page Layout, Form Letters, Reviewing, etc.)

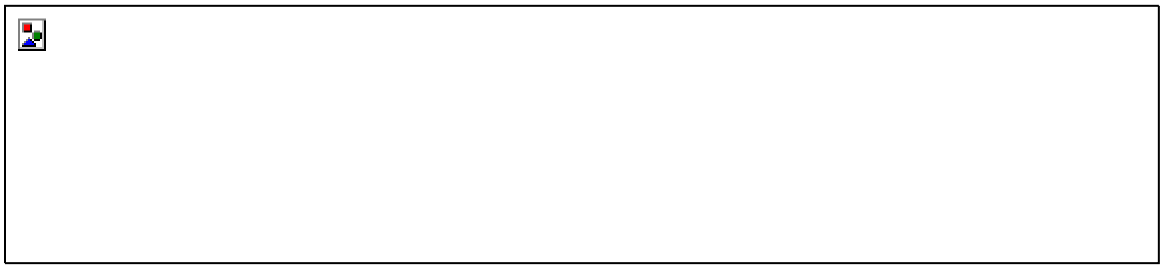

Figure 2: The first tab of the Word Ribbon. Designed to look familiar to today's Office user, the "Write" tab brings forward the best parts of today's UI.

Within each section (called a "tab") is the collection of commands which comprise the tab. The ribbon presents the commands within a section modelessly, giving you the much of the benefit of toolbars today—specifically, one-click access.

The ribbon is the single home base for command UI in Office. One of the reasons that today's Office user doesn't feel a sense of mastery of the product is the sheer number of places command UI is scattered: short menus, long menus, hierarchical menus, visible toolbars, toolbars accessible through View->Toolbars, toolbars accessible only through Tools->Customize, popup toolbars, task panes in the task pane list, task panes that only come up during certain scenarios, etc.

If you wanted to see what is possible in Word, where would you look? You'd have to look in at least 8 different places in the UI, weaving through hierarchical menus and task pane "stacks", making it nearly impossible to create a cogent mental model of the product's capabilities. On the other hand, the ribbon is the single, central point for command UI in Office. Everything that's possible to do in the product is in the ribbon[2].

In most cases, commands are labeled in the ribbon. We know from experience that most users can't decipher 16x16 icons into the features they are designed to represent. Although a few icons (Bold, Center, Save) are very well-known and don't need labels, most commands are not recognizable from the icon alone. This leads to "tooltip surfing" today where users start from the left-most toolbar icon, pauses over it to get the tooltip, and then move carefully over each toolbar icon to the right reading the label for each one. Worse, many users just ignore most of the toolbar buttons and use menus to access functionality beyond the top 5-10 features.

Because menus are so much more inefficient than toolbar buttons, however, it often takes 5 or more clicks to perform an action that would take only one click if the user knew which toolbar button represented the feature. By labeling items wherever possible, the ribbon keeps users from time-wasting "tooltip surfing" and gives efficient access to most of the feature set of

[2] There are, of course, infrequently-used commands that remain in the command well, but these are not used by more than a handful of super-users.

the application.[3] There's no longer a disconnect between the browsable way of finding commands (the menus) and the efficient way of using one (the toolbars.) It's all rolled up into one place.

The ribbon supports a variety of 2D layouts that communicate priority and hierarchy to the user. Unlike a toolbar or menu, which is a flat list that could be generated by computer, the ribbon is laid out by hand. Big buttons show the most important commands, while little buttons grouped together show a relationship of minor features designed to work together. Commands within a tab are grouped into "chunks," which provide a visual hierarchy that makes the ribbon even more browsable. The rich layouts possible in the ribbon mean that UI design can be more of an art than a science.

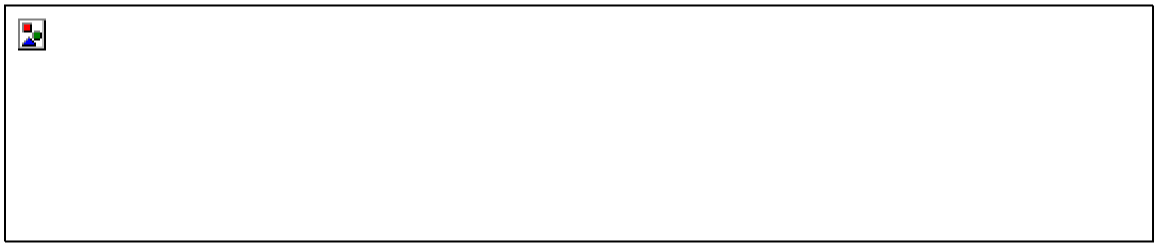

Figure 3: The Page Layout tab of the Word ribbon—important features are more prominent.

Contextual Tabs

An important part of the philosophy behind the new UI is contextualization. Today's UI shows all the commands possible in the application, regardless of whether or not it is possible to use a command in a given application state. There are some minor exceptions, such as a few toolbars that come up on selection, but these are the exception rather than the rule.

In the new UI framework, commands that operate on an object are not available unless the object is selected. When an object is selected, additional tabs become available in the ribbon called "contextual tabs." These tabs provide the UI necessary to manipulate, edit, and format the object. When shown, the contextual tabs work just like normal tabs; users can easily switch between core tabs and contextual tabs, and the same rich ribbon layouts and controls are available in each. When the user deselects the object, the contextual tabs disappear.

Examples of objects that will have contextual tabs: *tables, pictures, text boxes, shapes, charts, WordArt, equations, diagrams, PivotTables, headers/footers, etc.* Anything that feels like an object on the canvas and shows selection will have contextual tabs.

[3] The Outlook team found that, by adding labels to nearly all toolbar buttons in Outlook 98, they were able to increase usability substantially. Users that only used menus in Outlook 97 started using the toolbars in Outlook 98 and time on task decreased substantially.

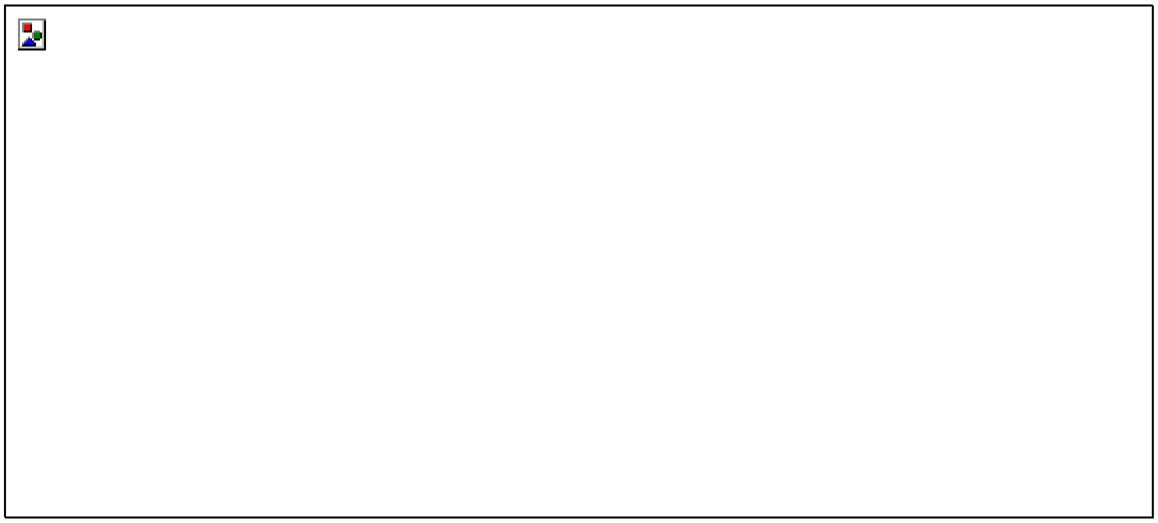

Figure 4: Table contextual tabs in Word

The key advantage of contextualization is a reduction in the effective command space the user has to evaluate at a given time. Most of the commands in an application are object-based. By only showing them when the right kind of object is selected, the number of commands to learn and browse in the core tab set is drastically reduced. Why show 100 disabled table commands when they won't work until a table is selected anyway? People can more easily create a mental model of the app when it feels like it has a manageable size. By never showing all of Excel on the screen at once, the UI feels responsive and relevant. By contextualizing based on object, you enable a mental model based on what feels like a physical relationship: a picture on the screen is drawn like a *thing*, and it makes sense to people that there are tools *designed to work with that thing*.

There is a significant relationship between the Insert tab in the Office authorizing apps and contextual tabs. Insert is the only tab that exists in each of the authoring apps; this is because it is so fundamental to the model. Most of the objects in Insert have a contextual set of tools that operate on that object; in fact, inserting an object navigates the user to the set of contextual tabs for that object. It's important that the Insert tab be enticing because when a user browses the core tab set of an application, we want them to see the power of the objects that application controls. We want a PowerPoint user to browse the core tabs of PowerPoint and say "wow, PowerPoint has tables and charts and pictures and sounds and movies and diagrams" without showing them an additional 1000 commands. In this way, the power of the applications is communicated without sacrificing the simplicity and browsability of the core tab set.

Three-Stage Formatting

Contextual tabs support a model of user interaction we call "three-stage formatting." We have observed that users consider formatting objects in three parts: choosing an overall design, modifying that design visually, and finally tweaking properties of the object at a fine level (if necessary.)

Each set of contextual tabs support this workflow. When a user first inserts an object, they are taken to the first tab of the contextual formatting ribbon for that object. This tab allows a user to select from a gallery of pre-designed styles which help them create a great look for the object. The user then can switch to the second tab of the tab set, which allows the user to change elements of formatting through individual galleries of possibility (such as adding different shadows, fill patterns, or border styles.)

Once the user has customized the look of objects using all of the visual tools available, they may still want to make a few advanced tweaks (such as setting the size of a picture to an exact number of pixels.) These tweaks can be accomplished through the ribbon or (in some cases) through the dialog launchers available at the bottom of the second-stage chunks. In this way, the formatting capabilities presented by the contextual tabs maps directly to the way end-users like to make their objects look right. UI to format at each stage is consolidated into a single experience; users no longer have to go on a goose hunt in the UI to find style-based and fine-grain formatting options.

Dialog Boxes

Although less central to the UI than in the current design, dialog boxes will continue to be part of the Office user interface. Dialogs remain are an excellent way to tweak a number of advanced settings at once, as well as being an ideal mechanism for soliciting user input. In most cases, if a feature requires typing into a number of edit boxes, it should be located in a dialog box instead of in the ribbon. For instance, setting up a complex multi-level sort in Excel is best done through a dialog box because there's a clear flow: type the search criteria for each level of the sort and then and click the Sort button to close the dialog and apply the sort. While one can imagine exposing cool "sort templates" and also quick ascending/descending sort on the ribbon, a modal task like setting up and applying a complex sort is well-suited to a model dialog box.

There are many dialog boxes today that provide more advanced versions of commonly-used features. For example, the Font dialog in Word is a superset of the more commonly used *Bold*, *Italic*, and *Underline* toolbar buttons. However, in today's UI there's no relationship between the toolbar buttons that provide quick access to a feature and the menu item that leads to the dialog box version of the feature.

Each chunk in the ribbon can have a "dialog launcher" that provides a formal link between the efficient, modeless command UI in the ribbon and the more advanced dialog box version of the functionality. The dialog launcher should be used in a chunk whenever there is dialog box that maps cleanly to its basic functionality. In most cases, this means that the dialog box should have the same caption as the chunk which launches it. This formalization of the connection between simple and advanced functionality helps make the product feel richer but also more manageable—there are fewer concepts to learn (only one UI place for font styles), but more functionality available to the user when they want it.

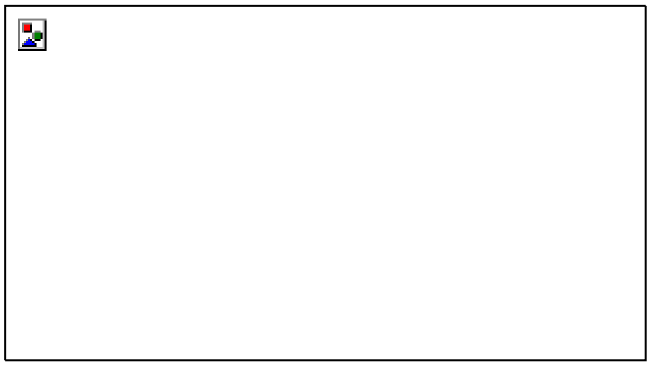

Figure 5: The Number chunk's dialog launcher leads to the Number dialog in Excel

Many other dialog boxes that don't map cleanly to chunks may still be launched from the ribbon, but they will be exposed as commands and not using the dialog launcher mechanism.

The Quick Access Toolbar

The Quick Access Toolbar ("QAT") provides access to the top few commands that are needed throughout the entire program, regardless of which tab the user is on. Users might be frustrated if they had to switch tabs just to save their document or to undo their last action. What tab should Undo be on? Really, it's so fundamental to the usage of the product that it should be always available. The QAT allows us to specific a set of commands that are always one click away from anywhere in the application.

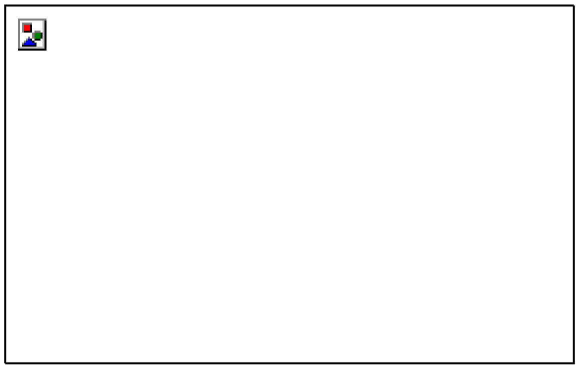

Figure 6: The Quick Access Toolbar in Excel

The QAT must not be a dumping ground for features we want to advertise in each new version. Extreme caution should be used when proposing a new QAT icon. One metric which has been proposed[4] is that no command shall be added to the default QAT until it has existed in Office for at least 3 versions. While the letter of that law isn't precisely necessary, the spirit of it is correct. The QAT real-estate is extremely valuable, and each icon we add to it increases the perceived complexity of the product—particularly if the icon isn't already well-recognized.

The QAT is also one of the primary places for end-user command customization in the product. If there are a few commands that a user wants one-click access to from anywhere in the application, they can add it to the QAT. If a person wants to build a large customized QAT full of many icons, the ribbon will move down to enable this user to build their own custom toolbar of controls above the ribbon.

Results-Oriented Design

One of the key design philosophies behind the new UI is our belief in "results-oriented design." Today's UI is optimized around exposing commands to the user. We've broken up features into discrete steps and we give names to those steps and then stick them on a menu or toolbar. Occasionally, when we believe strongly that the commands need to be executed in a specific order, we organize them into a wizard. More recently, when we think no one can understand how the commands work together to form a feature, we stick them in a huge task pane full of paragraphs of explanatory text.

While in some cases, access to individual commands is the right user model, the new UI is designed to facilitate "results-oriented design," in which we present the user with choices of how a series of commands would affect a document and then apply that result all at once. The user doesn't even have to learn what sequence of commands would be used to create such a great result. The user is suddenly able to produce significantly better output than they believed they knew how to achieve.

[4] By Clay Satterfield

Visual designs immediately come to mind, in which we show the user great table styles or combinations of picture effects and they get great results without having to know what features conspired to produce that result. But an equally important application of results-oriented design is in designing non-visual features that help a user achieve a difficult result in an easy way.

For instance, setting text in Word to flow around a picture is beyond the capability of most users today. To do so properly requires using a number of commands in the right order: set the Z order, set the right alignment option, set the anchor and padding properly. Although Word has a powerful set of features around text-wrapping, the very complexity created by having such a plethora of options makes simple text-wrapping difficult. Through results-oriented design principles, we could show the user samples of how their document would look wrapped in several common ways and let them pick the version that shows text wrapped around the picture the way they envision it. The user is empowered to use a complex set of features without knowing exactly what they did.

Another example is conditional formatting in Excel. Today, it's a set of rules and formatting options which together comprise a power-user feature rarely touched by end-users. By rethinking the feature with a results-oriented approach, we can envision presenting the user a number of smart choices of the ways they might want to look at their data (*out of range numbers red, low numbers blue gradient to high numbers red, etc.*) Suddenly, the user can harness the power of conditional formatting without knowing how to make manual rules. As a side benefit, the user can more easily tweak the formatting rules that have been applied without having to figure out how to start using the feature from scratch. Before long, they're using the full potential of the feature.

Galleries and Live Preview: Supporting Results-Oriented Design

The "gallery" is a new type of control designed to be the visual manifestation of results-oriented design. Conceived hand-in-hand with the ribbon and new contextual UI, galleries make it easy to show rich representations of the results of features. The gallery is exceptionally visual and represents the beginning of the transition between features being represented only as terse strings or 16x16 icons.

The gallery is the control that makes results-oriented design work in practice, in part because of its great flexibility. It supports many different layouts, incorporating rich graphics with adaptable text styles to communicate feature results to the user. The gallery can be hosted inside a ribbon tab, dropped down from a control in the ribbon, accessed through context menus, and even used inside of dialog boxes. The gallery supports rich layouts of content—it can be arranged in a grid or a menu-like layout, and the contents of a gallery can be filtered to help sort through many design choices. Because of the visual richness of the control, it can blow up thumbnails to a larger size as you mouse over them, making the overall experience feel responsive and fun.

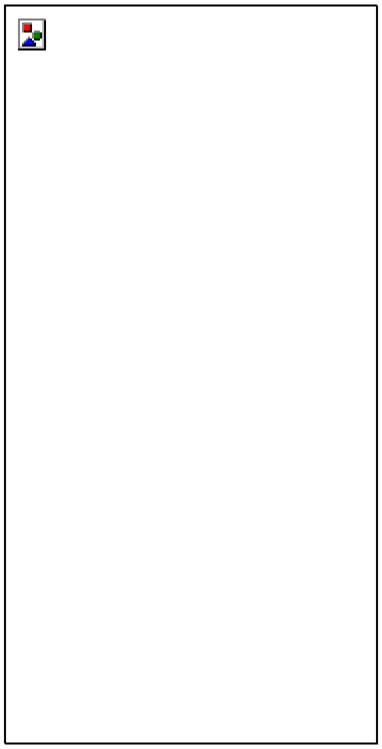
Figure 7: Rotate Picture gallery

In addition, the gallery supports "live previews" in a number of situations. There are two kinds of live previews: "thumbnail" and "in-document." Live previews show more than just a graphically designed representation of what a feature will do—they actually enable showing the exact result of the feature, either as a thumbnail in the gallery or, in some cases, by actually updating the document while the user mouses over the choices available. For instance, to rotate a picture, you could calculate the angle and direction of the rotation (say, 90 degrees clockwise) and then try it out to see if you calculated right. Or, using live preview thumbnails, we can show a gallery of the different ways your picture could look after rotation, and you can choose the one you wish.

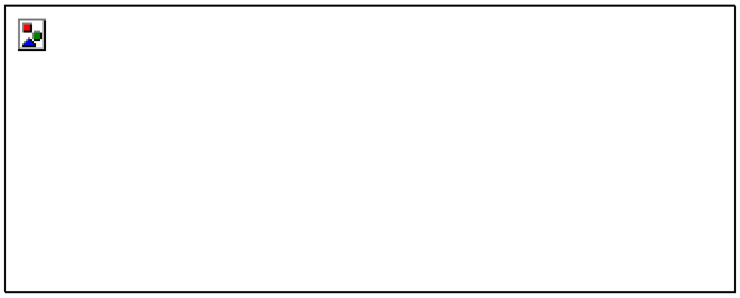
Figure 8: The “Insert Chart” gallery in Excel

In-document live previews work well for showing the results of applying design styles to objects in the document. For instance, no thumbnail can do true justice to what a table might look like after applying a specific visual style; instead, live in-document previews can provide instant feedback on what the table would look like as the user mouses over each style in the gallery.

Super Tooltips: Bridging the Gap Between Functionality and Help

Most toolbar buttons have a lot of metadata implicitly associated with them: the button’s name, the keyboard shortcut, the menu item associated with it, a description of what it does, and the help topic that tells how to use it. Getting this information from a command today, however, is nearly impossible. In the new UI, “Super Tooltips” bridge this gap, integrating quick access to information about commands directly from their location in the ribbon.

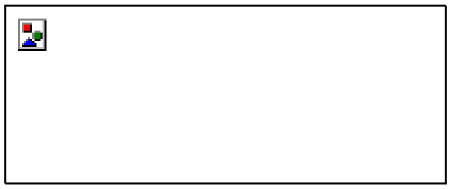
Figure 9: Super Tooltips link a feature to information about using the feature

Not only can we help users understand how and why to use a feature, Super Tooltips allow us to link to auxiliary information, such as training or a help topic, directly from the command itself. Users no longer have to discover the name of a command, open up the Help window, and type in the command name—the link is built into the UI. As the new user experience reveals more of the richness of Office to users, the additional in-context information will help people start on the right foot with unfamiliar functionality.

Scaling the Ribbon: Managing Variable Display Sizes

The ribbon is designed to take advantage of large displays while working better than today’s UI even on compact tablet screens. This is a big improvement from the way we design UI today. In the past, Office has generally targeted a single “base” screen resolution, most recently 800x600. All UI is designed to fit into this base resolution without any degradation. Because of localization concerns, in reality the English UI needed to fit into about 30% less space. Any UI that didn’t fit into the required space was relegated to an overflow menu or required unwieldy horizontal scrolling.

The ribbon, in contrast, is not designed for any particular “base” resolution. Each chunk of functionality is designed in several different sizes; as screen resolution decreases, more of the small versions of the chunks are used. But the ribbon doesn't just scale downward—as you move to a larger screen, the ribbon takes advantage of the space by showing larger versions of the chunks. In many cases, this means that someone with a very large monitor will see more gallery selections at once and, in some cases, more efficient versions of commands laid out at the top level. While someone on an 800x600 monitor might have to scroll to see more than 5 table design styles, a customer running at 1400x1050 might see 15 at once.

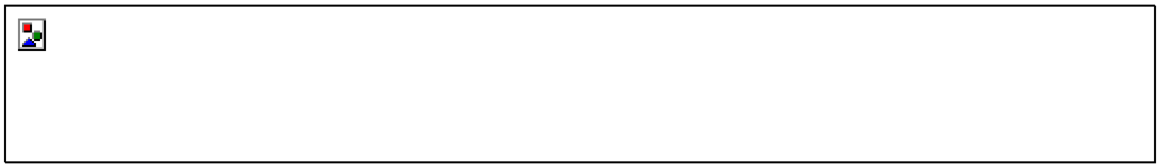

Figure 10: Chunks are designed in several different sizes

Because the chunks are designed with the different sizes in mind, we maintain a baseline usability benchmark across all screen sizes. Today, if a button is scaled off of a toolbar, it loses its position and relationship to the commands around it and is pushed into a totally different kind of control—a menu. The ribbon, by contrast, always keeps the position of controls in relative locality—space is saved by removing text labels and moving to more efficient (but less evocative) control layouts. And, we ensure that the UI is more future-proof by taking advantage of the screen space afforded by displays much larger than is commonplace today.

The Floatie: Super-Efficient Access to Frequently Used Commands

Because the ribbon is modal, there are many times in which the user isn't in the first tab of the application. For instance, when reviewing a document in Word, a user is likely using primarily the commands in the Review & Comment tab of the ribbon. But perhaps they're inserting comments and formatting the text in those comments over and over again. *Insert Comment, tab switch, Bold, tab switch, Insert Comment, tab switch, Italic, etc.* This is what we call a "command loop" and, if common, it would be an infuriating part of using the ribbon day-to-day.

Although most command loops can be fixed simply by us designing the content of the tabs smartly so that commonly used-together features are located concurrently, one class of features is so frequently used as part of other tasks that it would be hard to make work well with the ribbon alone: text formatting.

The solution to the need for quick, on-the-fly text formatting from many places in the product is an on-object formatter called "Floatie."

Figure 11: The text selection Floatie

On-object UI has been used in Office in the past primarily to help showcase rarely used or hard-to-find commands in the product.[5] Floatie does the exact opposite: through minimizing necessary mouse movement to use it, it provides the most efficient possible way of using the top few commands in an application via the mouse. Floatie is designed to work with the ribbon to

[5] Examples: Paste Recovery, AutoCorrect options, Text Import in Excel

minimize single-command tab switches. Mouse users can match the efficiency of keyboard gurus by using the on-object Floatie UI to perform frequently repeated formatting.

When selecting text, there are only a few possible things you could be doing with it: delete/typeover, drag to another location, cut/copy, or (most commonly) formatting it[6]. Floatie takes advantage of this by "hinting" a small toolbar near the mouse cursor on selection. Nearly anything the user does other than use Floatie dismisses it: move the cursor away, hit the scroll wheel or a key, press a mouse button. But if the user moves the cursor towards Floatie, it comes into full view and offers the most common formatting options: bold, italic, color, center, etc. In this way, users are actually more efficient at using the top formatting commands in the product regardless of what tab of the ribbon the user is on.

In addition, Floatie can be customized by the end-user to include any other commands they need quick access to in a formatting situation: *format painter, subscript, double-strikethrough, etc.*

Context Menus

One of the most interesting steady changes in user behavior over the last 10 years has been the emerging popularity of the context menu. Thought to be mostly a power-user feature in Windows 95 and hardly known about at all in the end-user community, users have gradually become more and more comfortable using context menus for everyday actions. In fact, in Office usability studies just over the last 5 years, we've seen a jump from most intermediate users never clicking the right mouse button to some users trying to right-click *everything*, before they even glance at the menus or toolbars.

The success of context menus in today's UI is due in part to the fact that they're a scoping mechanism. Unlike menus and toolbars which blindly show all the commands in the product regardless of what object you're using, context menus scope the command set to a reasonable list of 10-15 that are likely to work with the object that was clicked.

This positive aspect of contextualization is formalized everywhere in the new UI framework: in the modality of the ribbon, in contextual tabs, and in the Floatie. As a result, we can go beyond simply using context menus for discoverability and look at how they might be used to increase efficiency as well.

The new context menu is designed to be the more efficient way of exercising features displayed in the ribbon. Each set of contextual tabs in the ribbon is designed with a context menu version of those tabs in mind that allow you to perform the most common tasks for a particular object. Thus, to apply a single effect to a picture, or to quickly rotate it or change its brightness, the user may choose to use the context menu instead of activating the contextual tabs. Additionally, context menus for objects have a special item at the bottom which navigates you to the contextual tabs, formalizing the relationship further.

We know from SQM data that the top things context menus are used for today are clipboard actions: Paste, Copy, and Cut (in descending order, with Paste having a very large advantage over the other two.)[7] As a result, the new context menus formalize the importance of these commands, putting Paste as a big button directly next to your mouse cursor. This rearrangement of commands based on SQM is an example of a relatively small detail that will make the product feel efficient and polished, as if Office is working with you.

[6] People also sometimes select text absent-mindedly to mark their place when reading, and the Floatie design does take this behavior into account as well.
[7] In terms of percentage of overall mouse clicks in context menus Word: Paste: 28.4%, Copy: 17.0%, Cut: 6.6%.

Figure 12: Galleries and live previews work on context menus, just like in the ribbon

Because, like the ribbon, the new context menu also can host galleries, people can learn to use a feature once and use it the same way everywhere. There's no sense that the context menus are the "advanced" way of doing something—the same very visual design language (including results-oriented design and live previews) are supported in the context menus just like they are in the ribbon. In fact, the galleries are only built once and used everywhere they're needed.

Current Office context menus emphasize taking the user to dialogs to perform tasks; the new context menus promote working directly from the context menu. Many of today's context menus are simply lists of dialogs associated with an object. Right-click on a word in Word and the menu consists of "Font…", "Paragraph…", "Bullets and Numbering…" and other links that open dialogs. The problem is that most users don't have to be familiar with the font dialog in Word, and opening the dialog is a very inefficient way to achieve what the user likely wanted: a small change to the font properties of the selected word. The new context menus actually let you work directly through context menu whenever possible, showing feature galleries instead of lists of dialogs.

Floatie in the Context Menu

Another unique aspect of the new context menus is the way in which Floatie was designed to work together with it. There are a number of commands which are ill-suited for today's type of context menu. One example is "Brighten Picture", which the user probably needs to click multiple times in order to get the picture looking right. Using that command through the context menu would be frustrating because it would take several clicks to use the command each time: *right-click, move the cursor over the command, click it, (menu disappears), repeat…* Another type of command that doesn't well in context menus today are simple formatting commands, in which you are likely to want to change several parameters at once (perhaps bolding the text and changing it to 12 pt.)

To improve this situation, many new context menus have a Floatie embedded in them. This looks to the user like a simple toolbar included within the menu itself. When the user clicks on a command in this Floatie section of the toolbar, the rest of the menu disappears like normal, *but the Floatie stays up as a small Floatie toolbar.* By turning the context menu into a Floatie, it is possible to use several commands at once in a very efficient manner. Additionally, this marriage of Floatie to context menu makes it possible to extend the usage of Floatie to situations in which selection is not a good trigger (such as in the Excel grid, where you always have a selection). New context menus, therefore, can host in a reasonable way many high-usage commands not usually found in context menus today.[8]

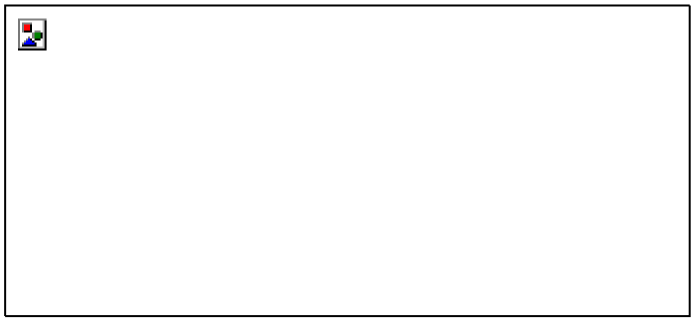

Figure 13: Part of the context menu turns into a Floatie

Once the context menu has turned into a Floatie, it behaves just like a normal Floatie—it fades away as you move your mouse away, and all the normal triggers dismiss it. The user only has to learn the way Floatie works once.

Managing Object Selection

A problematic stumbling block for many Office users today is that in many situations, it's difficult to select the right object. One example of this is in a table; unless you are very adept at using the clunky backwards arrow mouse cursor or the "plus in a box" icon, it's hard to correctly select a cell vs. the text in a cell vs. a row vs. a column. Add a picture inside of a cell and the problem becomes even worse.

Selection is also hard whenever there's a stack of objects on top of each other, such as a chart or a complex diagram. Clicking on the right part of the chart to format the way you want can be very challenging. Although Chart specifically tries to deal with this shortcoming via an unlabeled dropdown on the Chart toolbar with a list of selectable parts in it, it's so unique that most users do not benefit from it. In other cases, the "mode" of a specific object might be hiding command you're looking for. For instance, when Word has tagged a misspelled word with the red squiggle, the context menu no longer allows you to format the word—you have to ignore it or add it to the dictionary in order to get back to the context menu you're used to.

The new context menu provides a selection changer that helps to solve many of these problems. When you right-click an area, the most appropriate context menu pops up. At the top of the context menu is listed the piece that you clicked on—this is called the "selection changer." Clicking on the selection changer reveals the list of other objects or selections that are "in range" and allows you to correct your selection and reveal the tools for the new selection. For instance, I can right-click text in a cell and use the selection changer to select the row and reveal all the tools used to manipulate the entire row.

[8] Some of these high-SQM features are indirectly linked today, such as the fact that you could get to Bold through the Font dialog, but the efficiency of the new model coupled with it working just like the ribbon and the text selection Floatie makes it a huge win.

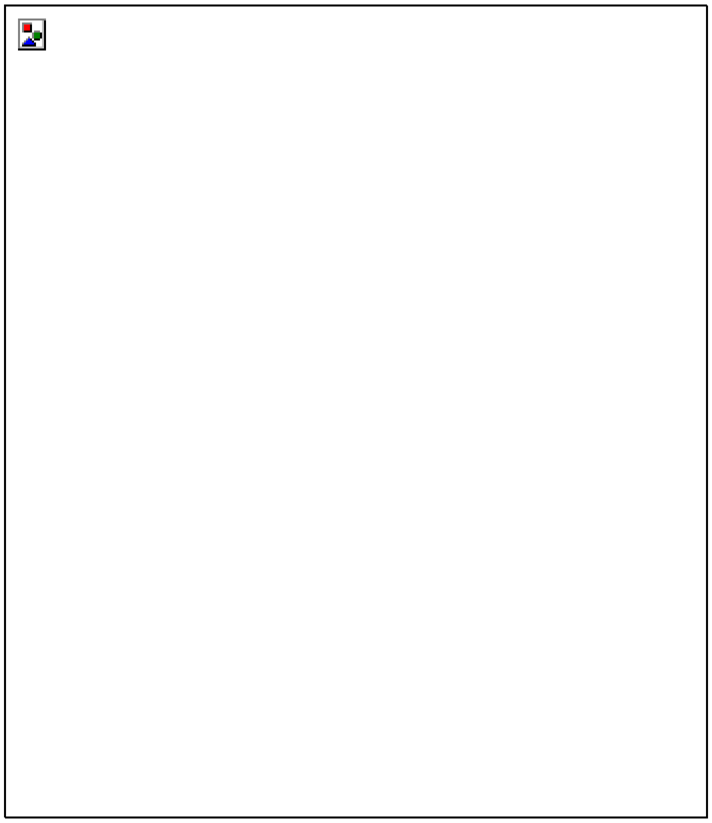

Figure 14: A picture in a table—selection changer makes it easy to select what you intend

Although we should continue to improve selection in Office to disambiguate objects and to clarify where you need to click to activate each part of an object, the selection changer gives the user the power to explicitly take control of selection. This helps to enhance the user's sense of mastery over the product and reduces occurrences of the user feeling out of control.

Task Panes

Task Panes were added to the Office XP product as an outlet for UI to help address the fullness of the menu and toolbar structure. Due to a lack of strict usage guidelines, however, Task Panes created a number of problems for the user experience of Office. Bloated with text and often interrupting the user's work, Task Panes quickly became yet another rock users had to look under to try to discover the functionality of the product.

One of the most often-heard customer complaints is that Task Panes come up without being requested and cover up their document space. Many features cause Task Panes to come up automatically; however, in only a few cases is the Task Pane really critical to the use of the feature. Users feel frustrated that this pane keeps coming open and covering their document. Task Panes have become just another source of perceived bloat—yet another way to get things done.

This is exacerbated by the design of Task Panes themselves, which creates an environment conducive to poor feature design. Because there's a huge rectangle of space to be used for a single feature, PMs have become experts at using all of it. Features that could be represented in 1/10th the space are instead paragraphs of explanatory text mixed in with a few controls here and there. PMs have gone ever further and designed all sorts of ways to squeeze even more text into these poor panes, turning them into clunky tabbed dialogs or pseudo-wizards. Most of these designs are ineffective and confuse the user.

Even if all of the Task Pane features were successful, though, the design of Task Panes themselves stifles their usability. Task Pane features are all contained within a single window, and only one can be visible at once. This leads to jarring instances in which features fight for the Task Pane window. A Word power-user who wants to use both the Styles and Formatting and Office Clipboard panes is out of luck. Even if I just want to use one feature, such as the Office Clipboard, other Task Panes automatically jump to the top of the stack and cover up the one I'm using based on context. Overall, Task Panes provide a poor platform for productivity-oriented feature design.

Task Panes are going away in the new user experience and being replaced by "Task Pane Lites" (TPLs). TPLs do not host command functionality—that's the purpose of the ribbon and related UI. Instead, TPLs are used when necessary to show auxiliary content to be used along with the document. The "Watch" window in Excel and the "Reveal Formatting" pane in Word are good examples; it makes total sense to use the content of these panes while working on your document. Panes that are oriented around command or galleries of choices (such as Getting Started, Protect Document, and Slide Design) are moving to other kinds of UI—in these cases, dialog boxes, the File menu, and the ribbon.

TPLs never appear automatically. The end-user chooses when to bring them up and when to close them. They are part of the frame around the document, and the user needs to be in control of this space. In addition, because every TPL lives is a separate pane, users can choose to bring up several at once. If I want to use the Office Clipboard and Reveal Formatting at the same time, I can do that—the features no longer fight for screen real-estate. The user is finally in control about what UI they want to see and when.

The Office Window Frame

The redesigned Office window frame pulls together a number of disparate tasks into a well-organized nerve center. More than just a simple status bar, the window frame provides consistent access to important document functionality such as view switching and zoom. Add that to a customizable version of the peripheral information and task status provided by a standard status bar, and the window frame is more than simple status: it's an important part of the overall user experience of the product.

The current "view" of the document and its zoom level are relatively independent of the task you're trying to get done in the app. In some cases, there are natural affinities (such as Page Layout to Page Layout View), but in most cases, a user is just as likely to do a task in several different views or zoom levels. Because the ribbon is modal, it doesn't lend itself to hosting view switching and zoom within it. It would be annoying to have to keep switching to a "View" tab when what the user is really doing is reviewing or writing.

Zoom is hosted in the window frame as a more visual feature than before; instead of being a place to type a number, it's an interactive slider that resizes your document on the fly as you drag it. This reinforces the feeling that Office is more visual and responsive.

Figure 15: View switching and zoom in the status area of the window frame

View switching has been co-located with zoom, as both sets of commands are about changing the way the document looks on-screen for easier reading or editing. Simple buttons continue to exist for one-click view switching in all apps, but a richer menu that controls not just the document view but the state of the window frame itself is hosted next to the simple buttons.

The status area of the window frame has been cleaned up, trading such pointy-headed acronyms as REC, TRK, and EXT for more user friendly information such as a running word count. Users can customize the status area to show many different kinds of peripheral information about their document—mostly information that would require opening up a dialog box to get today.

Office System Features

Many of the most valuable features in Office aren't about the document authoring experience at all; instead, they are about all the things you can do *with* a document: share it, protect it, print it, publish it, send it… Today there is nowhere in the UI where a user can see all of this value in one place; document-level features are mixed in with authoring features. As a result, we get little "bang for the buck" on many of our advanced Office System features—the very features which should be differentiating us from our competitors.

The new Office UI brings together the capabilities of the system into a single entry point in the UI: the File menu. This has two major advantages to the product. First, it helps users find these valuable features (which in many cases, users never would have found before.) Second, it simplifies the core authoring scenarios by allowing the ribbon to focus on creating great documents. The new File menu presents a task-oriented approach to managing and sharing your documents; many of the oldest, hardly-used features have been removed from the product, enabling the File menu to be simple and straightforward to understand.

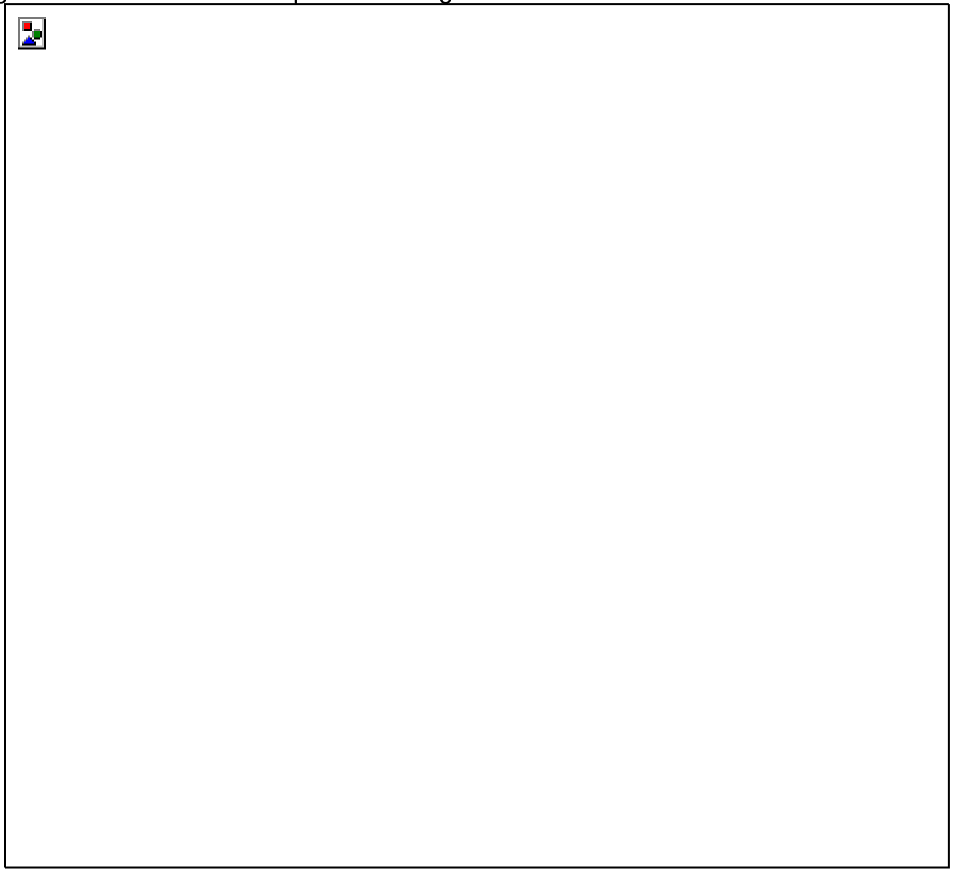

Figure 16: The redesigned File menu showcases Office System functionality

The File menu is also the new launching point to application settings, which have been redesigned to separate out and highlight the most common end-user options. Many common feature requests actually already exist in today's Tools.Options, but because of the devastatingly complex design (rows and rows of tabs each filled to the brim with unexplained checkboxes), many users open Options once and then close it forevermore. We've mixed extremely technical options[9] in with what should be simple end-user preferences, making it impossible to find the good stuff. The new design highlights the settings that make a difference, leaving the ones that are just clutter to the experts.

Figure 17: Streamlined options highlight the most useful settings in each application

On Organizing the Applications

One can't separate the task of creating the fundamental design language and control set of a new UI framework from that of actually organizing the applications to take advantage of that framework. Both parts walk hand-in-hand, feeding each other—the controls reflecting the reality of the applications and the applications benefiting from the richness of the experience. Progress in the UI framework can only be measured by the quality of the design of the applications. As a result, along with the growth in the control set has evolved design guidelines for using those controls effectively. The place where the most work had to be done is in developing a philosophy for organizing commands into the ribbon.

One of the first tenets of organizing commands in the ribbon is to let the soul of the application come through. A spreadsheet is fundamentally different from a presentation application. While some of the decisions we've made in the past based on consistency have had good results, the downside of consistency is that all of the UI ends up optimized for the least-common denominator. PowerPoint is a program in which drawing is one of the most important

[9] One of my favorites in Word is the option that lets users choose different ways to password-protect a document, such as "Diffie-Hellman", "DH SChannel", and "RSA and AES Prototype Cryptography." It also conveniently lets our faithful end-user choose an arbitrary key length for this encryption…

things you do; in Excel, it's mostly an afterthought. Yet in our current UI, they both share the exact same UI for drawing. The ribbon gives us a chance to rethink what our applications are *about*—an important piece of the command structure is communicating to the user what the application is good at doing.

Where consistency is possible and the trade-offs aren't major, we should embrace it. But we should revel in the differences between the products as well, because in those differences lie much of the unique power of Office. The new UI doesn't confuse *familiarity* with *sameness*. There's a difference between *consistent* and *identical*. One of the reasons people today miss so much of the power of Office is because the apps *are* so similar; the best-of-breed features that give us the overall advantage are tucked behind a veneer of needless similarity.

We do aim to make the Office family feel familiar; if you understand the design language in one app, you can use any app. Functional consistency to the user means that we've made design decisions in a consistent manner. It is critical that users can transfer knowledge and experience from one Office experience to another, but the consistency is inherent in the fundamentals of the design itself. Consistency in itself does not mean that all the commands need to be presented identically.

A common misnomer is that we intend for users to spend the same about of time in each tab of the ribbon. This isn't the case; although each application has a different balance, on the whole we expect most users to spend the majority of their time in the first tab.[10] It's no coincidence that the first tab of the applications maps to a lot of the common functionality on the Standard and Formatting toolbars today—this is what helps the new design feel familiar to users despite the deep changes. We've seen again and again in usability that people can adjust quickly to the new UI in part because the first tab of each application lets them be productive right away.

End-User Customization

End-user customization is an important part of the new user experience; we will offer the usual several powerful options to place specific, often-used commands in a more prominent place in the UI.

Much of the impetus behind customization today is a desire to co-locate functionality that is frequently used into a primary place in the UI. For example, the PowerPoint team analyzed screen shots of customized PowerPoint 2002/2003 layouts from MVP's and other people who design slides for a living. What they found was that the customizations made to the UI were by-and-large very similar across each of the screens. Office consistency dictated that secondary drawing functionality—such as Z-order and object alignment—was placed deep under fly-out menus. While that makes sense perhaps for Excel or Word, in PowerPoint these commands are very important and these power users reacted to our unoptimized UI model by manually customizing these commands up to the top level.

We believe that the need to do this kind of wholesale customization of an entire group of commands will be rarely necessary in the new UI. Between the task-orientation of the new UI and the focus on showing the right command set for each app (vs. a rigid consistent approach across all apps), most of these common customization scenarios from today will be solved simply via the new UI organization. All the tools you need for manipulating the style, layout, and order of objects will be available at once, out of the box. Users will no longer need to create their UI manually.

[10] Excel, which is optimized more for efficiency than the other applications, has over 90% of current mouse clicks accounted for in the first tab.

The majority of end-user customization in the new UI model will be driven by the user's desire to increase their efficiency when using certain individual commands (as opposed to groups of commands). A potential inefficiency inherent in the design of the ribbon is what we call a "command loop." A command loop is when a person uses a set of commands over and over again in sequence but those commands are not co-located on a single tab. In most cases, these will be solved by making sure that our command organizations truly reflect the functionality used together within the apps. In rare scenarios, however, a user might need to promote commands to the top-level of the UI simply to avoid switching tabs over and over. For instance, a paralegal editing a brief in Word might need to constantly switch between inserting a footnote and using Format Painter to correct the formatting of the footnote. Because these commands are located on different tabs, an additional mouse click would be required to use each feature (assuming the user didn't learn the keyboard shortcut.) This paralegal can improve her efficiency by adding the Insert Footnote command to one of the two places in the UI: the Quick Access Toolbar or the Floatie.

Although we believe that most users will not need to customize the UI much, the Quick Access Toolbar can grow to allow the user to put as many commands as they wish in it—if necessary, the ribbon will move out of the way to allow for additional customization space. The customizability of the new UI along with the improved functionality organization, better keyboard accessibility, and advertisement of keyboard shortcuts, will help create a powerful but responsive user experience, suitable for both beginners and power users.

Using the Keyboard Today: Shortcuts vs. Accelerators

Although some people use only the mouse to interact with command UI, power users and a growing set of novices have learned to use the keyboard as a more efficient way of getting things done in Office. There are two kinds of keyboard command access in today's UI: keyboard "shortcuts" and "accelerators."

Keyboard shortcuts are single keystrokes to perform common commands, such as **Ctrl+B** for Bold, **Ctrl+S** for Save, and **F12** for Save As. Keyboard accelerators, by contrast, are a way of navigating the menu structure with the keyboard using "mnemonics" (underlined letters.) A user can press **"Alt + A + I + T"** to Insert a Table or **"Alt + F + S"** to Save. Although accelerators are not as efficient as shortcuts, they work well in today's menu based UI because they don't require explicit memorization—you can figure them out as you go. In order to actually save time vs. the mouse, however, many users have memorized a few of them.

There are a few downsides to keyboard accelerators. One, as our menus have become more deeply hierarchical, additional keystrokes have been necessary to achieve common actions. Worse, keyboard accelerators only work for menus and not toolbars—but the efficient versions of features we build are on the toolbars, not the menus. As a result, starting a simple numbered list in Word takes a single mouse click on the toolbar but takes eight keystrokes.[11] Trying to use the keyboard to "click" the numbering toolbar button is even worse, requiring a minimum of 17 keystrokes[12].

Using the Keyboard Tomorrow: The New Keyboard Access System

The new Office user experience includes an innovative, ultra-efficient, and easy-to-learn keyboard access system that works hand-in-hand with the ribbon to provide a great alternative to using the mouse. It builds on the best of the current system, extending it further to reduce the number of keystrokes necessary to use most commands.

[11] Alt + O + N + Alt-N + Tab + Right Arrow + Enter
[12] Alt + Ctrl-Tab (3 times) + Right Arrow (12 times) + Enter. (Assuming you are using a high-resolution monitor and haven't changed the size or position of any of the default toolbars.)

Keyboard shortcuts will continue to work as today in the new UI. Many people have memorized these shortcuts and there's no reason to move away from them. We will continue to support customizable keyboard shortcuts in Word as in today's UI. To help more people learn these keyboard shortcuts, we will display them in tooltips throughout the new UI.

The concept of keyboard accelerators is extended by the new UI to provide a low-keystroke way of using the keyboard to execute commands or browse galleries without required memorization of the keystrokes. When a user presses **Alt**, a character appears next to each command, gallery item, or navigation button in the ribbon. These characters are the equivalent of mnemonic underlines today; pressing the displayed character executes the action. The benefit of this is that all commands—regardless of if they have text labels or are totally visual, are represented by a character. Because navigation between ribbon tabs are represented by numbers and commands are represented by letters, the new keyboard system takes advantage of the "modes" of the ribbon. When a user is on the *Page Layout* tab in Word, for example, every command in that ribbon is only a single **Alt + Char** keystroke away. Commands in other tabs are only an additional keystroke away.

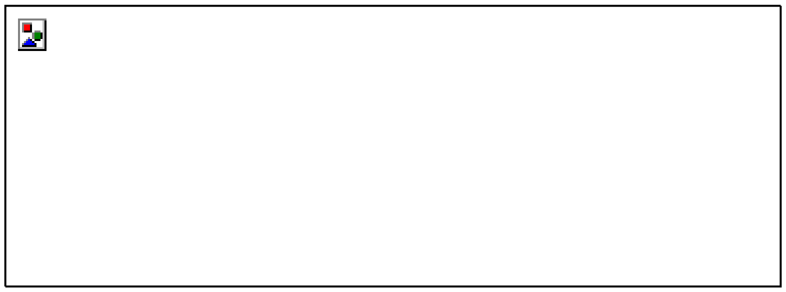

Figure 18: Character overlays show the user what letter to press

In this way, keyboard access to ribbon functionality is uniformly more efficient than accessing that some functionality via the keyboard today. As a bonus, the new model is very easy to teach (like "paint by numbers") and builds on the familiar accelerator model of today. It also lends itself equally well to people who like to memorize keystroke combinations and never have them change and people who will use the keyboard to "navigate" the UI based on what's displayed on the screen.

For those users who have heavily invested in memorization of today's keyboard accelerators, a users can choose to activate the old keyboard accelerators. This will not work with new features and even with some old features, but it will help bridge the transition period in certain vertical markets (such as financial services) in which employees are trained to use the product with no mouse whatsoever. Over time, though, we expect these users will see the benefits of the highly efficient new keyboard system and switch over to that.

From the Perspective of Power Users

As we make the product more approachable and less mysterious to the average person, there is always the concern that we might alienate the "power user." After all, these users have already invested the time and energy to become experts at using Office. They pride themselves in being part of an elite crowd of super-formatters and ultra-efficient number crunchers and slide doctors. In some cases (such as professional slide designers and table formatters), their very jobs are at risk as users discover the features they've been using for years. The new UI puts these skilled users on a more equal footing with the normal end-users we're trying to empower. Where will we stand with power users?

Professional formatters will actually benefit from the new UI. Because they already use a substantial subset of product functionality, the improvements in organization and efficiency will help them get their work done faster. Instead of fighting the UI or needing to manually customize it, their basic work will be streamlined. Great results will be possible with only a few clicks because of galleries and the new formatting UI; as a result, day-to-day projects will be completed quickly. Professional slide designers today are by-and-large paid to do things that aren't possible to do in PowerPoint—as the state-of-the-art improves, these designers will also improve their state-of-the-art. The new UI will allow them to spend more time creating stunning visuals (something they're better at than basic end-users) and less time fighting with the program itself.

So-called "power end-users" will like a number of things about the new UI. For one, we know that these users are very sensitive to screen real-estate issues and might not react positively to having the ribbon open on the screen all the time. Because of this, we have designed the ribbon to collapse to a single-line (about the height of the menu bar today). From there users can use the new keyboard accelerator system, floaties, the enriched context menus, and the Quick Access Toolbar to provide all common functionality in a smaller UI footprint than today. When necessary, the user can expand the ribbon temporarily to use an infrequently-used command and collapse it again with a minimum of effort. Because we've consolidated all of the command UI in the product into a single place, these power users are in total control of their screen space. They can collapse the ribbon and that's all the UI that will be taken up without the user's consent: no unwanted floating toolbars, palettes, or task panes popping up.

Additionally, there are efficiency gains compared to today's UI in each part of the system: the new keyboard access system, floaties, the new context menus, galleries, contextual tabs, the QAT, etc. Power end-users who want to discover the most proficient way of getting things done will have no shortage of hyper-efficient ways of using the product to achieve expert status.

Power users are already knowledge leaders; they know how to use more of the functionality in Office than anyone outside of Microsoft. As end-users finally begin to discover the rich feature set power users have known about for years, there will once again be an opportunity for these power users to exert their ego—to get behind the new release of Office. Because their existing knowledge becomes more valuable, power users will be in great demand as tutors, teachers, and helpers in corporate and social situations. There's a lot to appeal to power users in the new Office user experience.

Supporting Office Developers

Developers are an important part of the Office ecosystem. They write the add-ins and macros that delight end-users and provide functionality not found in the core applications. They also play an important part to our continued success in the corporate market; many large businesses rely on the extensible nature of Office to run their business. Although in many cases, these customers only need to add a few commands to Office, without this capability, a new Office isn't a viable option.

Most of the ability for developers to surface their functionality in Office is through the CommandBars object model, which provides access to the menus and toolbars. Add-ins today which use the existing OM will have their functionality surfaced in an "Add-ins" tab in the ribbon. The "Add-ins" tab intelligently maps requests to add menu and toolbar items into a compatible type of UI in the ribbon. Any requests by developers to remove UI will simply be ignored. In this way, add-ins targeted at previous version of Office will run without modification.

Developers will be encouraged to update their add-ins to take advantage of the new object model, which provides rich access to many parts of the new user experience—especially the ribbon. The new object model allows designers to add chunks of functionality to tabs on the ribbon to seamlessly integrate with core Office functionality. Or, a developer can add their own complete tabs if they have a scenario that warrants adding a lot of commands to the product.

As a nod to simplicity, the hard-core developer tools must be enabled in Options. This keeps them out of the way of the end-user, but allows us to build a feature-rich ribbon user experience for those developers who want it. Instead of trying to “hide” the developer UI amongst the core application functionality, we can embrace it as a complete task of its own and provide it with a very powerful experience of its own.

Users of the programmable Task Pane introduced in Office 2003 will continue to be able to use this space to build custom solutions. Some improvements are planned to this space, such as allowing UI to exist at the application level as well as at the document level.

In Closing

The parts of the Office 12 User Interface System work together to create a powerful user experience for the customer. Users will begin to discover and harness the power of Office and use it to create great documents, presentations, and spreadsheets. They will discover the system-level Office functionality that allows them to share, protect, and manage their documents. People will stop cynically commenting “nothing in Office has changed since Office 95.”

We can add features until we’re blue in the face, but the last ten years have shown us that we will not get credit for our work until we empower the user to take advantage of them. It’s time for Office to redefine state-of-the-art interaction design, to take a fresh look at our products and build the user experience that will serve as the runway for the next decade of innovation.

By empowering and exciting end-users, we will make Office 12 the first must-have piece of software of the decade. The limits of our continued success in productivity software are constrained only by the power of our imaginations and our passion to delight our customers.