Getting more out of Koffice2

By Dennis Pennekamp

1 Synopsis

Making Koffice more efficient, starts with increasing the productivity of the text editing proces. I feel it has come to a standstill since the eighties because every office suite uses or mimics the same api. It is time to improve on the essentials and this is possible during this early stage of KDE 4 and Koffice 4 development.

2 Text editing engine

Text selection
Since the early days of the Macintosh when dragging selected text around, spaces would automatically adjust to their expected position. This was necessary as double-clicking a word would helpfully select the word (mostly space enclosed) before triple clicking would select the paragraph (hard return enclosed text). Easily selecting and dragging words would hardly be a convenience if you had to clean up spaces afterwards. I think we can do better than this.

Double-click plus clicks
Double clicking initiates a round trip cycle which starts with selecting a single character and per click moves toward selecting the entire text and back. Time between clicks may be endless until interrupted (by initiating a drag or moving the mouse from its position (away from the character).

Implementation
As soon as the mouse stops moving above a text an engine would start analysing the text for common structure elements. Recognising individual words, sentences and paragraphs. The engine would be available system wide, modular and relatively easy to expand. Meaning that later on recognition of selecting compound words, URLs, compound sentences and the compounds of URLs can be programmed and easily selected. Modules can be deactivated if they are unrelated to a particular language, keeping the engine as fast as possible.

Peer selection
When selecting a sentence or paragraph a darker selection color can signify all the words within. This is extremely helpful when rearranging the words to change the tone or logical structure of the text.

Example
I select the word 'have' either by dragging or by double-clicking:

I'm thirsty. Do you want to [have] an ice cream or a softdrink?

I could drag 'have' near 'you' but I would have to do it precise as there is no auto-formatting by default. If I keep the mouse pointer at the same location after double-clicking I can click again to select the entire sentence:

I'm thirsty.< [Do] [you] [want] [to] [have] [an] [ice cream] [or] [a] [softdrink]?>

Now I can drag the words within the sentence and spaces would automatically be corrected. Alternatively I can click again to select the entire line, the entire paragraph, and again for the entire document. If I would click yet again the maxed out selection would reduce its scope down to paragraph, sentence, word, character, nothing and then continually loop up and down this cycle. Just like you can loop boldness on, off and on again with [Control]+[B]. It's consistent.

A related key which should also cycle would be [Select All] from the [Edit] menu. Let's keep it's usage simple, two options plus a 'leave it as it was'.

Select All
Select Nothing
Select As It Was

Yes, an undo build right into the cycle just in case you change your mind as you see the result. Every cycle should have a build-in undo, even [Control]+[B] to allow you to think "What is better, making everything selected bold, removing bold or the way it was, let's see all three option come by again".

So I dragged, clicked or used [Control]+[A] to select everything:

{<[I]['m] [thirsty].> <[Do] [you] [want] [to] [have] [an] [ice cream] [or] [a] [softdrink]?>}

Notice that for English the engine would need to check a list of words which have remained disconnected but are considered to be a single compound, like 'ice cream'.

Some more grammar cases
Dragging 'ice cream' onto 'softdrink' would swap the two words. If the engine would be given some rules of basic English it would outdo the ancient space fixing trick by checking for a leading article when swapping nouns and also swap 'as' and 'a' at the same time. The text in the status-bar or tooltip would actually tell me it wil do so:

Release to swap 'an ice cream' with 'a softdrink'

As 'I'm' can only be a contraction of 'I am', you can drag them off of each other. With 'he's' this would be more difficult and would perhaps require a trip into the contextual menu to pick the wanted word. In reverse you could drag '[he] [has]' together as '[he]['s] by dragging it close to end of '[he]'. Depending on the advancement of the engine I could even drag '['m]' from [I]['m] not only would it convert to [am] while in transit, when I drop it behind '[you]' it would suggest to append it as '[you] [are]'. If I would be unhappy with this default behavior I could call the context menu upon '[are]' and it would have a submenu named [Paste suggestions] listing:

'm
am
are
m

No, don't add less likely yet still logical suggestions, you don't want to give the user information overload. For more see contextual menu.

Sentences
You can swap sentences by dragging a sentence onto the word in another selected sentence. By dragging a sentence onto the word of an unselected sentence you can add the sentence being dragged into the precise location just like you can today.

By dragging a sentence onto a selected space within or in front of another sentence you can insert the sentence and perhaps the engine will help with correcting capitals and punctuation in obvious and expected ways. Perhaps there is no obvious way and you can pick options from the contextual menu after pasting:

Sentence >
    Simple
    First Compound
    Middle Compound
    Last Compound

For example, two simple sentences:

{<[I]['m] [thirsty].> <[Do] [you] [want] [to] [have] [an] [ice cream] [or] [a] [softdrink]?>}

We could call up the context menu in the first sentence and choose to quickly turn it into a First Compound:

<[I]['m] [thirsty], [do] [you] [want] [to] [have] [an] [ice cream] [or] [a] [softdrink]?>

Multiple selections
With an easy way to deactivate all selections in a document (Select All, Select Nothing) and after experiencing incapsulated selections the step to understanding and using multiple selections is quite small. This is where paste-board manager can step in to help organize copying, reordering and pasting multiple selections.

Keep the cursor in sight
The most important thing when editing is having overview of the text and where you were working on it. So let's use the power of KDE split screen to keep the cursor in sight while scrolling the document to look up stuff.

Scroll up from the middle of a text and just as the 7 lines containing the cursor are about to disappear at the bottom those 7 lines remain visible in a split window view box, while the rest of the page scrolls on.

Scroll down from the middle of a text and just as the 7 lines containing the cursor are about to disappear at the top those 7 lines remain visible in a split window view box, while the rest of the page scrolls on.

Click selections to copy and paste
Now that the cursor remains we can go out and collect text in different parts of the document for pasting at the cursor. We can select a paragraph by dragging or double-clicking and then click words and sentences which will automatically be pasted at the cursor. We can quickly rewrite a text without writing it again or destroying the original paragraph because of moving selections. We simply copy and paste a new paragraph out of existing text merely by point and click.

You can also make multiple selections with the arrow keys simply hold [Shift] while navigating the cursor over characters you want to select or deselect with the arrow keys. When using the arrow keys to move the cursor to the text you want to select you can use Home and End to quickly navigate to where you were last typing or pasting. See Home and End keys in next chapter.

Enhance selection instead of replace
When a partial sentence is selected pressing the quote key, or any similar enclosure symbol, would not replace the selection with this symbol but enhance it by automatically adding both the open and closure characters at the ends of the selection. Perhaps even respect some grammar rules.

Similar to changing style of the entire selected text by pressing [Control]+[B], pressing [Tab] on a selection would inline the entire text to the Tabstop and when pressed again would move entire selection to the next Tabstop.

Reverse
Combination with some modifier key (like [Alt] or [Control]) would reverse the enhancement: removing enclosures, moving Tabs to the left and removing Tabs.

Take selection as a suggestion
When the user tries to Save As the current text selection is used as suggestion for the name.

3 Execute on release

Ancient ways

To mimic a physical typewriter a key executes directly at being pressed down. Except for modifier keys like [Shift] and [Control] which gives them the advantage of being able to tell you what they'll do and show you what options there are. It would open up a new world of possibilities if all keys would execute upon release. For example holding down [.] and pressing [Tab] could fill dots to the next tabstop location, or typing [*][1][2] could place exactly 12 dots. But I guess that would require some getting used to and lead to some unnecessary resistance to adoption of Koffice and KDE.

Special keys tell you what they'll do

However I see no problem doing this to special keys meaning all keys that don't spit out a character. Holding down [Tab] and pressing [.] could also mean fill it up with dots until the next tabstop, and with another tap at the [.] key the next and so on, well you can see what you get. Holding down [Tab] and then holding down [Delete] could make the status bar say 'Delete this tabstop from this line'. Letting go of [Delete] would do so and adding [Delete] again could make the status bar say 'Delete this tabstop from this paragraph', and the next from this document, letting go then would really remove the Tabstop which any good framework would then allow you to undo, perhaps even while you are still holding down [Tab].

Whoops I did not want to press this button

But execute at release also has the advantage of being able to cancel the action beforehand by hitting [Esc] before letting go of the keys. Imagine this when pressing [Control]+[Q] as a warning:

You have multiple tabs open and you are about to close them all. Our implementation off-line caching and history overview is lacking so you can't quickly find them again once you do, perhaps you first want to check them all with [Control]+[Tab]. Are you sure you want to quit and close them all by releasing all keys?

[Quit] [Cancel]

being rendered before release. You are given the option to quickly check them by letting go of [Q] and tapping the [Tab] key, and if you then release [Control] you will continue using the application right there. Alternatively you can also tap [Esc] to cancel quitting the application. Or you can click the wanted option with the mouse-pointer. Or continue the initiated action simply by releasing both [Control] and [Q].

As the Application has quit KDE may be smart enough to simply hide the application, allowing the user to realise its mistake and use Undo to cancel quitting an application before having moved on to something else.

A future advancement could be having touch sensitive keys on the keyboard to help you out as the statusbar could say what can happen when you make contact with a key, even before you press it down. But let's first continue with Koffice 1.5 goals by themselves.

The [Home] and [End] keys can also be looped functions

[Home] could make the cursor jump to the beginning of the current word, the current sentence, the current line, the current paragraph, the current page, and finally the top of the document. As [End] is there to reverse the steps pressing [Home] at the top of the document would land you straight to where the cursor was before you started clicking [Home]. Blindly hitting [Home] gets your cursor at the top of the document, you don't have to count how many times you hit, so to get back at the starting position requires a small usability pause which would be unnoticable when you are actually scanning the top of the document and are therefor what slower.

Home again
So as you hold down [Home] for the second time the cursor is about to go to the beginning of the sentence and the statusbar says:

'Cursor to beginning of sentence. Combine with [Page-up] for others.'

So you keep [Home] pressed down and hit [Page-up] and the cursor goes to the beginning of the sentence and you hit [Page-up] again and go to the beginning of the sentence in front of that one, and so on. Hitting page down would reverse until you're at the beginning of the original sentence, bring you back to the starting position, then it would move you beyond, to the beginning of the next sentence and then the next and the next.

The advantage of it all
You don't need to learn lots of modification key combination (it's true, I never learned vi or emacs :-), you can expand on the usage of a special key and the statusbar will help you explore options. You keep [End] held down and statusbar says:

'Cursor goes near the end of this sentence. Combine with [Page-up] for others.'

If you let go the cursor goes to in front of the punctuation dot and you can expand the current sentence, you can hold down [End] again to make the statusbar say:

'Cursor goes beyond the end of this sentence. Combine with [Page-up] for others.'

Like [Home] it then goes to the end of the line, end of the paragraph, end of the page, end of the document. And at anytime you can add [Page-up] or [Page-down] to move the cursor to across any of the current peers.

These functions of [Home] and [End] can also be used be used during selection. Even when you are dragging from the middle of a word back toward the beginning of a sentence by holding down the [Left mouse button] you can hit [Home] to help the selection to move along toward the beginning of the word, the sentence, the paragraph etc. Consistency means that once you have learned a trick you can apply it at different occasions.

Scroll-up
Now that [Home] and [End] move the cursor around in the text in all kinds of expendable ways, the [Page-up] and [Page-down] keys can focus on scrolling the page without ever moving the cursor, which is extra nice as scrolling automatically splits the window. It would be even better if you also know where you where when you where reading and needed to scroll. In KDE 4 this can be easy: as you press down the [Page-down] key an alfa blended overlay appears and as you release to scroll-down the alfa blend scrolls along and afterwards you can clearly see where in the text the window border used to be, and before the blend has faded away you have found the sentence you were reading. People will say it is mere eye candy but the average reading speed will go up.

White out
A translucent black overlay may be the obvious choice for the visible part about to be moved, but consider a white overlay on the invisible part as it allows for a trick I call white out. When memory is too restricted for a large fully rendered document (or more) you can cache it as white out meaning that a translucent white alpha blend covers it thereby reducing the number of colors. It can still be recognised and even read by humans but costs little memory and can be efficiently compressed with PNG reducing the need for memory page out while remaining quickly scrollable. As we stop scrolling one of these too large documents which we have opened on a computer with little memory that visible part of the document can be rendered again and faded into visibility, making the user think its just added eye candy, while it is actually keeping the computer responsive under overload. This can also be used to keep tab switching on tabbed browser windows, multiple scrollable windows and multiple virtual desktops responsive on slow computers while the system scrambles the page outs needed to make the selected part interactive on computers too slow for using KDE 4 as exuberant as they learned on a better computer. And on better computers white out can be used to have a full sized

preview cached with little memory cost, for example to show the last state of a webpage as auto-completion helps you find the page you want to type.

Start scrolling in any direction
You can combine the held down [Page-Up] with an arrow key and release both of them at the same time to scroll in an alternative direction instead. Leaving one finger on [Page-up] and tapping the arrow keys is not as comfortable as tapping the [Home] or [End] keys to scroll to the top or end of the document, but you can transfer the function of a special key onto another key which does not modify that special key. Example:

While holding down [Page-up] we can hold down [0] and release [Page-up] to make [0] act as [Scroll-up] key, now the arrow keys are easily reachable and we can scroll the page in any direction. As we were holding down both [Page-up] and [0] the status-bar would have said:

Release both keys to stop. Release [Page-up] to transfer its function to [0]

Indeed, you can use an unassigned key to stop the function without any further action. This is a shortcut for use of the [Esc] key. After pressing down [Esc] the status-bar says:

Stopped

You can release the keys in any order you wish, simply pressing down [Esc] has stopped current events immediately, just like it always does.

Surround access 1
Assigned keys should be at predictable locations, meaning that most of the time the arrow keys can be used; to adjust direction or to navigate through a list of related alternative functions. But they may be too far away to reach comfortably. We learned how to transfer functionality of the special key but we can also do it the other way around.

The unassigned keys surrounding an execute-on-release key may mirror the arrow keys. See surround access 2 for more.

Character keys
So the user experienced execute-on-release, understands how it works and wants more of it. The user goes to the preferences and changes [Character keys] from [Repeat mode] to [Execute-on-release].

Characters always had alternatives which have mostly been difficult to access, requiring learning of ASCII codes, combination sequences or a full list to be displayed and searched through. Pressing a modifier key like [Shift] was simpler as the keyboard shows the possibilities but [Alt] (as on Mac OS) had to be learned. Execute-on-release still allows these learned approaches but adds the possibility to navigate through all available alternatives for each specified character, reducing eye scanning for wanted options.

Alternatives would be made available from a list which can be navigated by tapping the left and right arrow key while the character key is being pressed. Some alternative alternatives are are more logically ordered underneath up and down arrow keys like [Shift] or super and subscript positions and placing numbers in a complex formula.

Let's assume c and c with a cedile are the only two options for character c, the status-bar would help you navigate with the current choice in bold:

Release to type: **c** Alternatives: **c** ç (use arrow keys)

After tapping the right arrow key the hint at the end could change to tell you about the (Mac OS) shortcut you may wish to learn if you use this alternative very frequently:

Release to type: **ç** Alternatives: c **ç** ([alt]+[c])

Left and right arrow keys should let you walk entire list of options, tapping [Right-arrow] again while still holding down c doesn't bring you round to the beginning of the list but enters the list in [Shift] mode:

Release to type: **C** Alternatives: **C** Ç (use arrow keys)

For alphabetical characters a logical alternative path would be to tap [Up-arrow] for capitalized version of current alternative and tapping it again for superscript.

Another classical style of shortcuts is to add a visually similar looking character with a modifier key to combine the two into an alternative. For example [,] because a cedile is located at the bottom of the baseline or [5] because a cedile looks like a 5 hanging at the bottom of the character. Well all shortcuts are open for debate on wether or not they're an intuitive combination so it's nice that we can show the user the short cut without needing a lookup list.

These classical shortcuts can also be accessed while holding down the original character. Adding in [alt] seems logical but you can as well add [,] or [5] to go straight to that character in the list. Not as impressive with this list of just 2 alternatives but you get the picture: Hold down the master character use the combination key you do remember and you jump to that position, forgot the combination then use the arrow keys instead.

When the status-bar is too small to fit the full instructional text it only displays the essentials in this case that would be the list of alternatives:

c **ç**

Shifting numbers
The number keys at the top row of the QWERTY keyboard are an exceptional case compared to the alphabetical characters since shifting them gives access to different characters, luckily we stated that left and right arrow keys give access to all alternatives so let's use key [5] to explain. Holding down [5] and tapping [Right-arrow] key gives access to its Shift alternative and the status-bar will say:

Release to type: **%** Alternatives: 5 **%** (use arrow keys)

Meaning that we can use up and down arrows to access super and sub script notation on '5' without going into shift mode, which is very logical (dare I say intuitive) for a number. Statusbar makes it seem logical at least:

Release to type: **5** Alternatives: **5** % (use arrow keys for super and subscript)

It also means that we can access percentage character when holding down the [5] key located on the number-keypad. Yes, I do love consistency :-)

Surround access 2
Key 5 on the number keypad is made for showcasing surround access; as you hold down [5] the keys surrounding [5] can be tapped to navigate the alternatives of 5 just like shown on the number-keypad itself. It's not hard to imagine them all moving to the right as you move to the right and start with holding down [6] to access the alternatives of 6. The key on the right of [6] works as the right arrow, the key one the left [5] as left arrow. Let's take a more complex location next.

Keys in a row
Let's look at key [5] from the top row of a qwerty keyboard. The key above is aligned meaning that as long as 5 is pressed [F4] can mirror as the [up-arrow]. The keys below are not aligned so to keep things simple we can use either [R] or [T] as down arrow key. The row of course is nicely aligned:

[1] [2] [3] [4] [5] [6] [7] [8] [9]

While pressing [5] nearby keys mirror the arrow keys. The key on it's left as left-arrow, the key on it's right as right-arrow.

[1] [2] [3] [left] [5] [right] [7] [8] [9]

But what if [5] was the [§] key which doesn't have any keys on its left side? Well let's put it next to the [right-arrow] key, and do the same on the other side:

[1] [2] [right] [left] [5] [right] [left] [8] [9]

But what if the keys above or below are not there to mimic the [up-arrow] key? Expand some more:

[down] [up] [right] [left] [5] [right] [left] [up] [down]

Notice that from now on you can comfortably type all you want with just one hand and little movement. Channel the status-bar output to a Braille reader and we've got some nice usability improvements.

Execute-on-release expanded
We can further develop the opportunities that execute-on-release gives us. For example when holding down [Control]+[B] to make text bold we could allow the mouse to click entities, like words, we want in bold to apply bold. No more precise selecting just pick want you want to change. Where will it end? I don't know. But I do believe that once you have become accustomed to Koffice 2 you will feel left out in the cold when sitting behind non KDE applications.

Overview of Execute on release
Character keys remain in repeat mode by default to ease acceptance.

Pressing a special key gives you a preview of what will happen upon release, likely using the status-bar.

You can add the [Esc] key to stop further action and safely release all keys being pressed.

You can add an unassigned key and release both keys at the same time to stop further action.

You can transfer a function to an unassigned key by adding it and then releasing only the original key.

The status-bar may also hint you with assigned keys which you can press for related alternative functions.

Alternative functions are executed upon being released, together or without the original key.

After an alternative function has been executed, releasing the original key:
may or may not execute its original function (programmer's preference);
may execute not its original function but an alternative function instead.

Unassigned keys surrounding the original key may function as arrow keys to easily access alternative functions.

The mouse may be used to apply the function of the special key to objects while the special key is being pressed.

An on-screen keyboard or keyboard with a screen can reveal all assigned keys.

4 User interface Layout

The user interface should imho be cleaner and faster. The cleaning requires further evaluation of default elements needed on screen, more of a groups process then an assignment like this report, I will suggest a default but it's just a starting point or target to keep in mind. For faster I have some innovations which will make KDE 4 and Koffice leap others in usability and productivity department.

The default interface should try to reduce itself to 3 diverse rows at the top of the window and 3 icon columns at the right side of the window with optional panels on the right side of the columns. The 3 row consist of:

Menu-bar
Icon-bar
Status-bar

Faster menubar access
Just like a click on the menubar was redesigned to become sticky in 1986, the menubar can keep up with modern times too and still remain compatible with what users have come to expect. As our screens grow access to the menubar becomes a longer haul, even with mouse acceleration.

So we introduce the traveling click. As we move towards the menubar we click while in motion. The pointer seems to become magnetic and shoots straight through to the first widget in its trajectory, likely an iconbar button, with a universally small travel distance (as in being independent of the size or length of the item) we make the mouse jump to the next widget: the wanted menubar item.

After clicking the wanted item, an empty 'visual divider' item or the [Escape] key the pointer actually travels back to the original position plus all additional movement, making it compatible with absolute input devices (tablets). This can actually be visualised by leaving the shadow of the pointer behind during a magnetic hopping session.

Faster submenu access
Menus have been a great way to organize, find and spaciously remember the whereabouts of general functions. But the actions of other office suite developers trying to distance themselves from the competition has people wondering if indeed menus have become overcrowded and too inaccesible. Why not first look at how we can improve menus.

Accessing a submenu requires following a narrow path along the name of the master item. In most menu implementations you can deviate a few pixels without missing the submenu, but you can't go straight to the submenu item you want as you would travel across the other items in the master menu.
Solution:
Clicking an item with a submenu makes the submenu slide underneath the mouse pointer, and the content of the submenu then scrolls to position the default item underneath the pointer.

Why that last bit?
Because most users have a scroll wheel now. For those that don't we still have boundary initiated scrolling. For the very fast the scrolling means they can click items as they scroll past. For the fast user it means they can interrupt the scrolling with a counter scroll, and travel very little to click the desired submenu item. For the slow it means they end up at a predictable location in the list and can start seeking from there.

The default item of a submenu could be a preset item, a popular item, or the currently selected item.

The outline of a submenu would not scroll, only its content. The list of items can be scrolled as a continuous loop, at least 3 empty items would mark the roll over at the end of the list.

To further speed up the decision making process we can bolden menu items the application predicts as the next logical step in a workflow or sequence. When their is no obvious workflow a developer can also turn this thought process around and keep items which are logically at the right location in a menu

but are unlikely to be commonly used from being displayed in bold. This results in menus which can guide you while you keep a complete overview of available functionality. At the same time menus continue to work as they used to.

For those who hate to click but still want to access the submenu in the same way there can be a simple mouse gesture: a u-turn toward the submenu drawn as horizontal dash forward and back, symbolizing you pulling the submenu towards you.

And I have thought of another method to predictably increase the usability and feedback in (sub)menus. Let's explain by example:

Three magic columns
In a submenu listing fonts the outline of the submenu is secretly divided into 3 mouse-over columns across all items. As we move along the font names we can get a preview balloon when we move the pointer from the left column over to the centre column, as we move into the third column the font preview shrinks abruptly then smoothly grows as we move further to the right edge of the submenu outline.

Meaning we can
1 - move across the list of fonts without tasking bandwidth,
2 - quickly see a (prerendered) preview of each font
3 - smoothly (on a fast computer) see a preview of a font at different sizes
Sure a fast computer can live reformat the entire document as we move across the list of fonts but this can do more and does not require users to upgrade hardware to get a fast responsive interface.

There will be other uses found for the mouse-over columns, the most obvious would be showing a short tool tip when the pointer is above the second column and a extensive explanation when above the third column. The numbering of the columns can depend on the space available on the screen: will the preview or explanatory window have enough space to float on the left or the right side of the menu. The float would be at a distance from the (sub)menu and fade when you move of the menu but if you move fast you can go to the float and actually click links (to manuals etc.) there.

The icon-bar
Shortcut icons on the icon bar should be grouped per 3 items, making them easily transferable to the side-bar while the user continues to recognise them immediately just because of their grouping.

Transfer may be necessary as the window is too narrow to contain them all or a lengthy horizontal strip is required for a specific application. They transfer to prevent having to tack another row onto the window, reducing the vertical space for the document.

Horizontal
A1 A2 A3 B1 B2 B3

Vertical
A1 A2 A3
B1 B2 B3

The grouping of three can of course combine to a grouping of 6 related items. But we can also abuse the 6 into 3 pairs of shortcut icons, which gives us enough options to handle reality and still keep appearance elegant and quickly recognizable.

Horizontal
A1 A2 B1 B2 C1 C2

Vertical
A1 A2 C1
B1 B2 C2

Notice that it has the elegance of a Tetris game. C1 would preferably be an 'up' related function and C2 as 'down' (for example undo and redo).

Why is 3 the magic number?
Three columns would transfer just as easily from the sidebar to a cheap slim lcd remote control. Three columns elegantly house numbers, navigation buttons and mirrors the 3 rows. It is slim enough to be placed next to the panels such as needed for Krita without looking cluttered. Remember the kids office mock-up response not on usability (which would require reading the articles) but on looking clean and uncluttered. (See my blog). Now image Koffice itself being visually compatible with such a layout, making the transit for young users easier. Imagine having the icon-bar and sidebar filled with icons and switching to a larger screen, as you widen the Koffice window the groups of icons smoothly travel from the sidebar to the icon-bar until the sidebar is empty and gone. You can choose to optimize vertical or horizontal document space, without loosing the grouping of icons, without manually laying out the interface.

Status-bar
Earlier we have used the status-bar extensively to inform the user. The position underneath the icon-bar is optimal to further this task without needing tooltips. Example:

As we move the mouse over the status-bar current text reporting on progress of background proceses gets replaced by the short names of all the shortcut icons. As we mouse over a specific shortcut icon the text becomes bold and may expand to its full name. There may be two short help sentences in plain text. One explaining the function the other for hinting (for example hinting its keyboard shortcut). Two means that they can be placed on one side when the shortcut icon is at the border of the icon-bar or on both sides when the icon is in the middle.

Especially in simple applications this will look very clean and uncluttered.

Sidebar
Depending on how much pixels you want to move the short names may overlap the top of the icon row underneath the shortcut icons which you hover, or push down all rows underneath to make space. The two help sentences appear in the status-bar.

5 Text composer

When writing a document you may not know beforehand how big it will be. Will it have chapters or just bold headings? Do you want explanations to be footers or to be collected in a vocabulary, will it be small enough for a fold-out? Will referenced pictures stay close-by? Within the 2 visible pages of an opened book? How can you be sure? Will it require full proofreading with crosschecking? You may find you are wasting time continuously correcting the layout while you are writing. With text composer you simply write your text and mark it as different from the rest and similar to its peers. After having written the text you can assign the right labels to the structured text and test out different layouts. But while writing you simply pick color labels.

Text composer would be a new component within Koffice allowing users to color code structured text without knowing the element names beforehand and without using layout tools. Output could be converted into structured markup text like Latex, HTML etc. Implementation would use code from the multicolor selection engine.

Using a limited HSV color-picker all background colours remain in the pastel style of nonintrusive colors keeping text readable with high contrast. Each color used gets an entry in an overview panel, a bit like the layer panel for Krita.

Using indentation in the overview panel a color can contain the full color index again but now as part of the master color. This is how we mark relations between elements in the text, for example a footnote belonging to a word or sentence.

A color in the overview panel can be picked with a shortcut or applied to words or sentences lick a fillbucket. Precise implementation requires evaluation of best practises among current HTML and Programming editors.

When the text is done colors can be reassigned to layout elements (green colored text are chapters or headings) using a choice of layouts (chapters are numbered, chapter start on a new page, there is no index, chapters are also placed centered at the bottom of each page etc.)