KOffice Functionality Mockup

by Aivaras Kirejevas

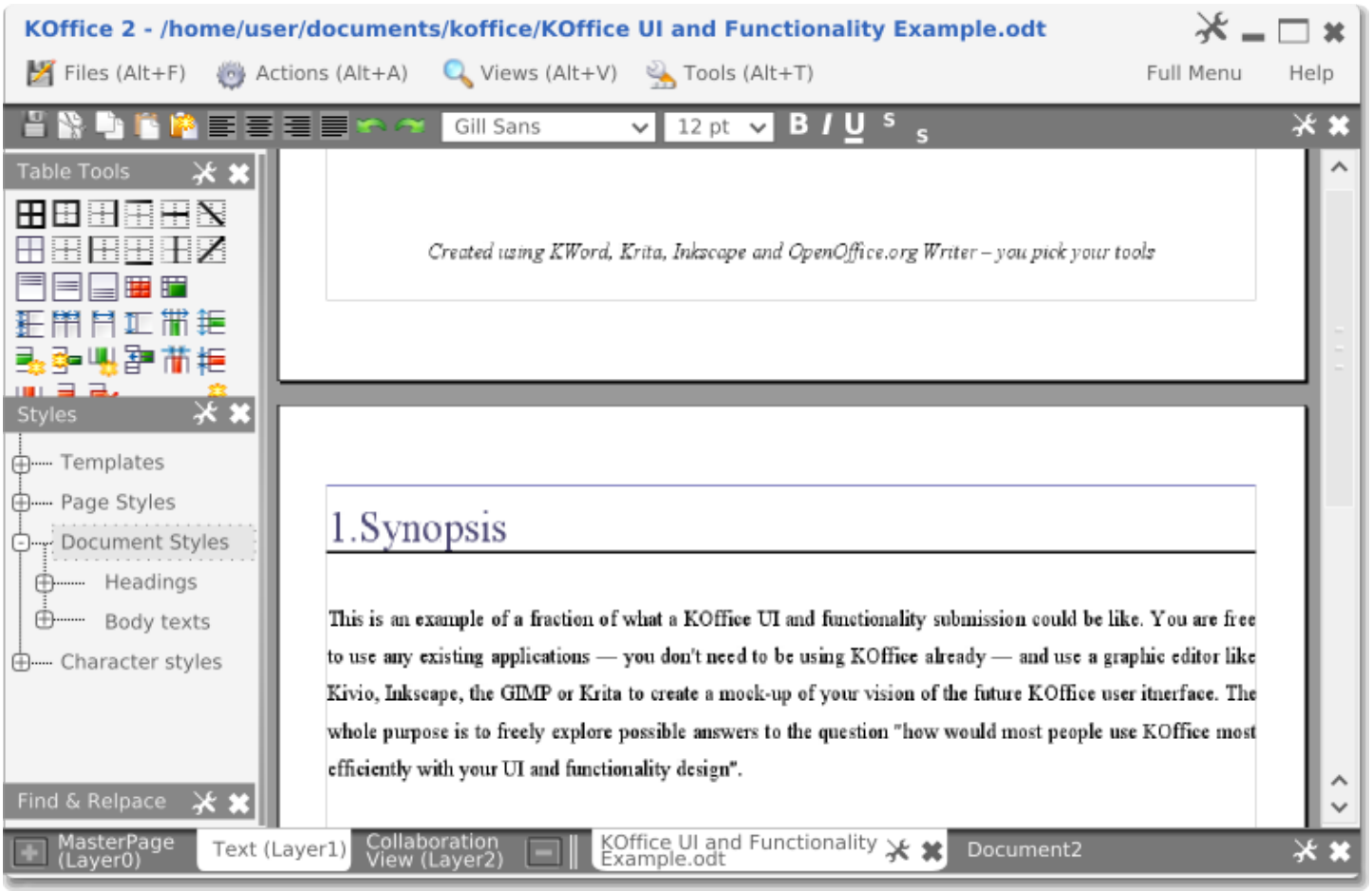
aivaras.kirejevas@gmail.com

2006, Kaunas, Lithuania

1.Synopsis

This mockup show more efficient and simplest KOffice UI. I hope my submission simplicity code writing for developers and allow advanced users easy extend and configure KOffice for his own task.

Basic Views



My vision are not revolution, only evolution. I want to using sidebars instead dialogs, add tab bar for documents and editing modes (this feature for advanced users only), using back side of windows for settings, options and properties and add extra flexibility to KOffice tools sets.

2. Flexible User Interface

Issue:

Users have not identical skills, knowledge and the needs. But application interface are one for all users. Novice and advanced user see a lot of menu items and various buttons. Developers balanced between complexity and functionality, between novice and advanced users.

Also, most users usually need simple tasks - read, print or little edit the documents and unusually create or format complex document. But we have one interface for all tasks.

One interface for different users.

One interface for different tasks.

Not so good.

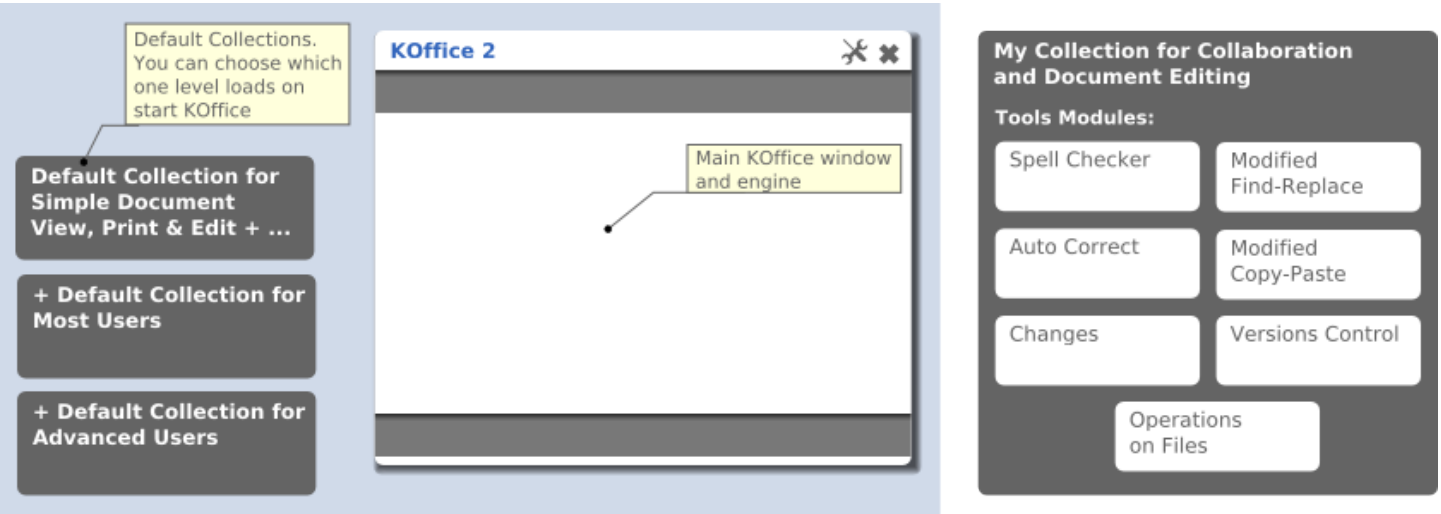
Solution:

My solution is three *levels* of flexible interface:

1. Simple interface for fast open, read, print and edit of document; minimum functionality;
2. Nice interface with common functionality and 'eye-candy' for every users;
3. Interface for advanced users and complex tasks.

How you can fit to this without a lot of programing?

- Using modular principle, PlugIns.
- Gather modules in collections for various needs.
- Collections of modules or separate modules place on the Web (good example - PlugIns Web page for Firefox).



Some characteristic of modules collections:

- User can start application in simple mode quickly without some modules for quick view and print document.
- User can switch to Basic mode (or start KOffice in Basic mode) from simple or advanced using mode. When switching, interface is changing by loading Basic Collection of toolbox modules.
- For basic users – basic set of modules.
- Every tools set can be loaded or unloaded separately from application. Depend on user wish (settings).
    - Also, same toolbox can be saved with different name and different settings ('All Table Tools' and 'Some Table Tools') in separated collections. It is useful for different tasks.

- Every module (tools set, collections) can be interchanged by user with another same module with different interface fashion and functionality deepness.
- Advanced users can collect his own individual modules collections and use it in advanced, basic or simple mode.

User can have tools sets as in real life: on your desk you have pencil, paper, rubber, glue; in kitchen room you have knife, glass, cups ... not everything in one place. As in real life, with flexible collectioning users can simple configure his own KBookMaker, KLetter, KPostCard, KWeb, KForm, KDiagram *etc.*, without programing and compiling KOffice, without a lot unused tools in UI.

Tools modules Collections must be main feature for KOffice *not optional*. With tools modules collections KOffice can ideal fits to everyone. Flexible KOffice saves functionality and looks more clear.

3. Styles

Big Styles list in OpenOffice.org is bad idea. For easy navigate this 'bad idea' OO.org using filter which not exactly fit to users. In OO.org user can't clear unused styles from default styles list, user need create additional styles.

Better is allow users manipulate on default, create or load his own & clear Styles list.

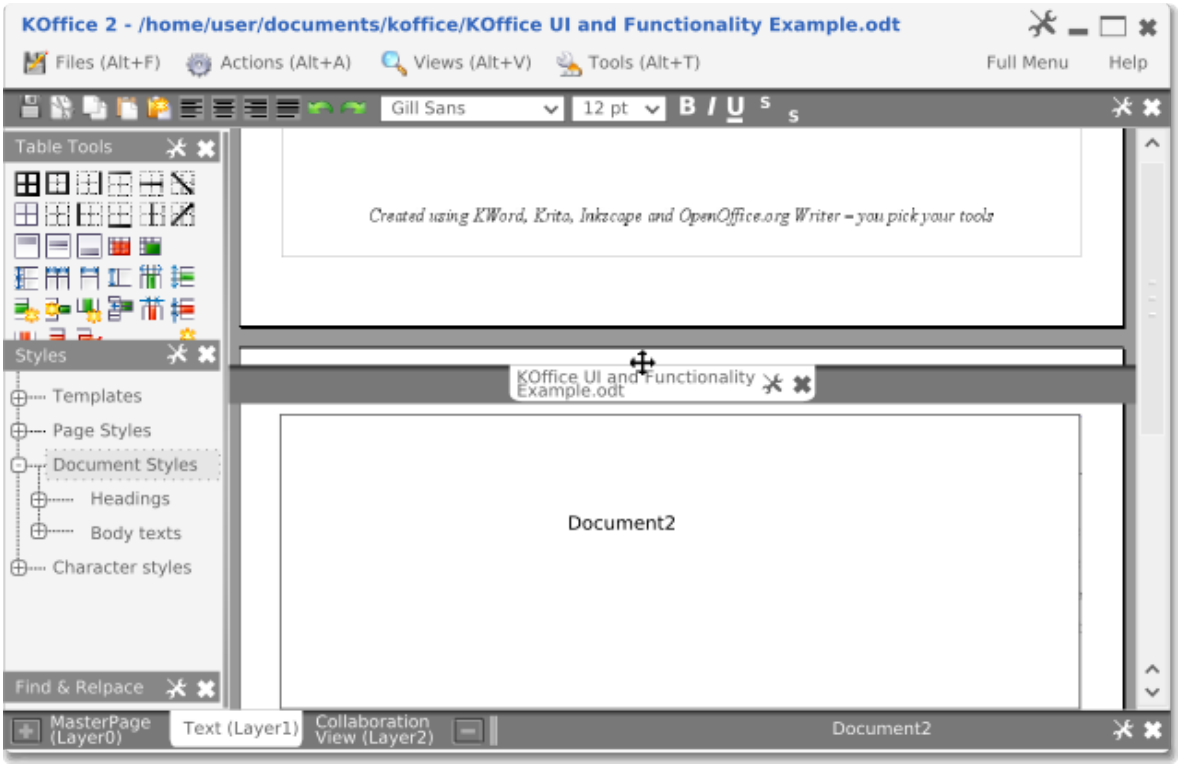
For beginners can be loaded default styles list.

Styles list can be shared separately from KOffice as all another tools. Users can set individual styles names. When user use in document styles list from another document, for easy apply user need something like *Styles Substitution Table* like as Font Substitution Table.

4. Document TabBar

In my vision, ToolBar and SideBar are same as in most applications. Additionally I'm add TabBar.

'Window' menu item are rudimentary thing. Tabs for switching between documents are better. Off course, KDE have taskbar. But on taskbar can be placed a lot of applications and document buttons frequently 'disappear'. In Tabs we can directly change file name or can Drag it for split KOffice window:



5. Layers TabBar

Layers can be useful only for advanced users. In every layer user can setup which tools activate, which hide, change layout view, etc. User can setup layer, for example, for view Forms Editing tools and when switching to this Layer, in SideBar and ToolBar opens FormTools and hides another unused tools.

In short, layers in my vision means tools and settings *collections switcher*.

User can save his own created layer with some tools as Tools Collection and share it to community or using for another works. Layers can hide many of tools (which user not usually need) in KOffice window and can contains individual settings for various user tasks.

6. Flip Window for Edit Properties

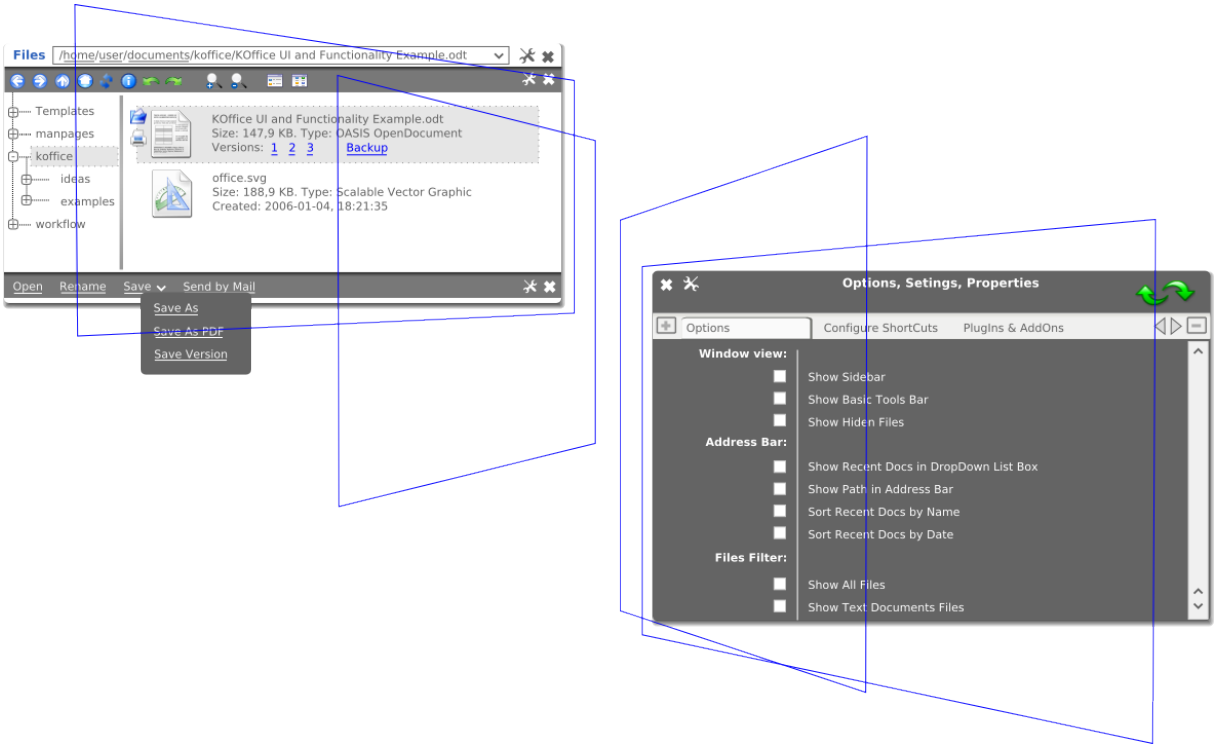
Issues:

Options and Properties of applications, windows, documents and tools usually are hidden deep in menu tree. For configure something user have long search where is little CheckBox which user need.

Solution:

Good example (in OpenOffice.org) are 'Visible Buttons' feature (little arrow on the ToolBars end) for adding tools into ToolBar. Hidden tools are very close to place where it need.

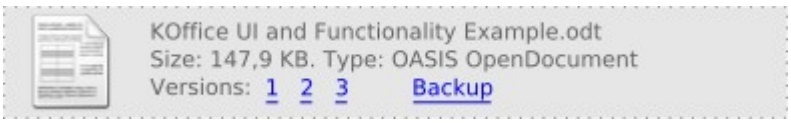
For another boxes and windows my solution is 'Flip for Edit'.



Flip to window backside (with or without animation) and edit tools options or document properties, configure what you want as you want. This thing can be used for everything: KOffice window, ToolBars, ToolBox in SideBar, TabBar, documents etc. This idea are easy understand for all and are very user friendly. This could also reduce the menu clutter of KOffice significantly.

7. Document Versions

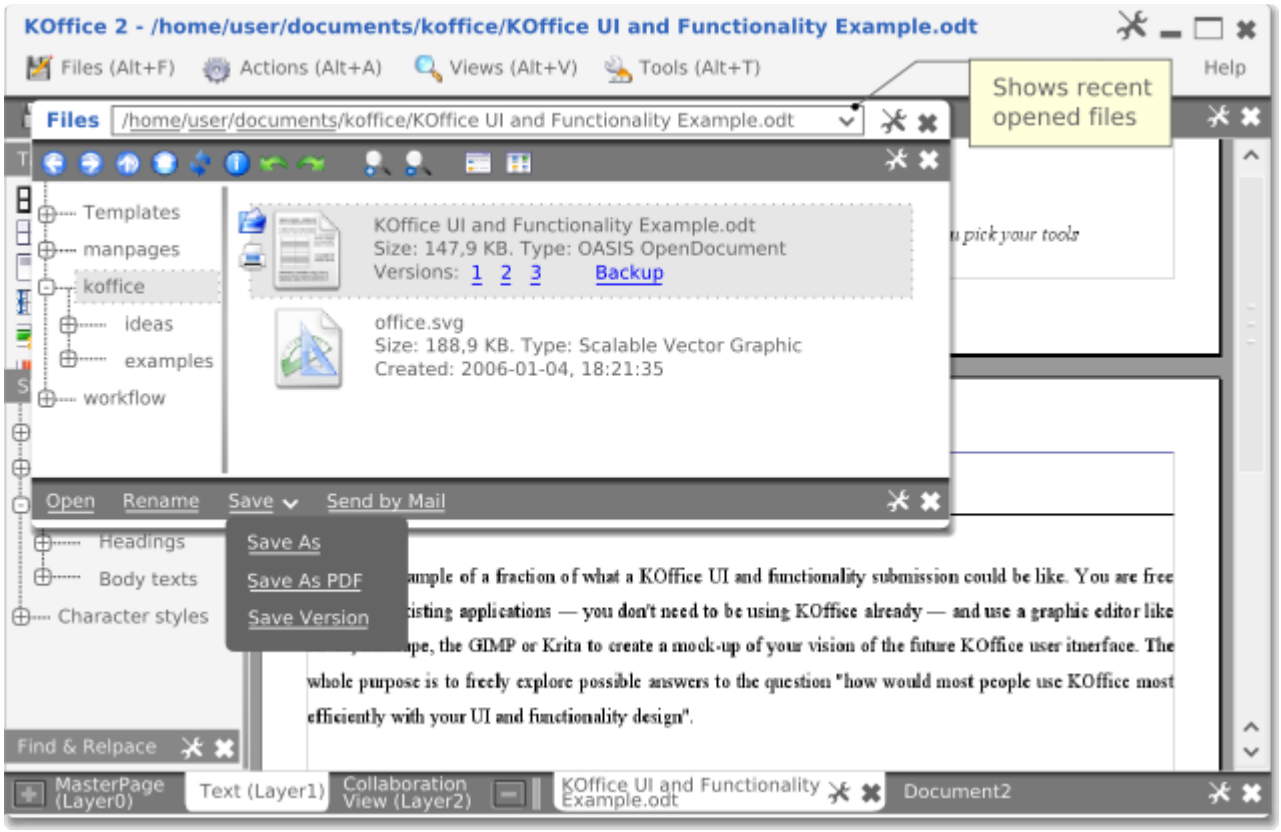
At this time when user need save or open versions, he need remember about this functionality. With backup copies are worse, user must know where backup copy of lost document are placed and how to open it. I think, document versions must be easy view in Files operation window and backup version must be easy accessible.



8.Main Menu

This is not important changes. IMHO looks better and are more useful for novice users. In figure, Files menu item *directly* open window similar to Konqueror. User don't need open another window for operation on files. Because more functionality and properties can be placed in sidebar or window backside, menu can be short.

However, advanced users can install Tools Collection with old fashion menu or something else.



9. SQL for Text

Issue:

In current programs user can with macro's create his own tool or menu commands. It is good for advanced users for power up some tasks. But:

- not every advanced user (writers, managers ...) know programing languages very good;
- automatic 'RecordMacros' are not very useful;
- tools and menu sets are not fitted for community experiments on it without bug and wish reports:
  - user can't change basic tools or menu commands, user can't redesign tools for his own purpose-build;
  - developers can't create (have not a lot time) tools which exactly fits to everyone users, advanced or novice.

Solution:

I think, advanced user need some specific text modification language like as SQL. Relational databases would

be difficult to use without SQL. Other languages not allowing easy access and change data.

'*SQL for Text'* functional simplicity and simpler command set. In my opinion this are good solution. This language add more advantage for every users:

- like SQL for relational databases, *SQL for Text* more simplest comprehensible language for users, comparing with other script language. For example, changing style to 'Head1' from 'Default' for text which centered, in bold and shorter than 5 words:

  ```
  UPDATE document1 SET (paragraph.style.Default=Head1)
  WHERE paragraph.alignment=Centered
  AND paragraph.word.count<=5
  AND paragraph.word.count>=1;
  ```

  There no loops, no complex scripting. Try to write the same in other script language available in KOffice and compare;
- SQL have object orientations (`document1.paragraph.style.Default`) which needed for XML document structure.
- flexible operations on text, allow processing several documents in batch:

  ```
  SELECT Head1 AS toc1, Head2 AS toc2, Head3 AS toc3
  FROM '/home/doc1.odt', '/home/doc2.odt'
  ```
- simplest way for creating users own tools or recompose default tools. For example, recompose find tool: mark founded text in color instead 'find next'. Part of program code can be moved from KOffice engine into tools settings box, easy accessible for advanced users and no compilation required.
- For KOffice developers '*SQL for Text'* means less coding works - only '*SQL for Text'* interpreter, not text oriented tools.

Of course, user can have all this with other scripting language, but I think '*SQL for Text'* are more simplest and comprehensible for all users.