OpenUsability

Heuristic evaluation of amaroK

Preliminary study to learn about possible improvements in the user interface

Author Chi Shang Cheng

Note: the name “AmaroK” is used instead of “amaroK” for grammatical correctness.

Abstract

After a short heuristic evaluation of Amarok, a total of 41 usability issues were found. Only 8 issues of low severity were found, 11 were marked medium, and the remaining 22 issues considered highly severe.
Most of the issues were related to design flaws. The application was not tested for task-oriented usability. Even though the application functioned properly from a technical perspective, attention should be given to a more aesthetic user interface design in the future.

1. Introduction

Digital music is very important for many computer users. The KDE desktop has two major applications to fulfill the needs of the digital music enthusiast: JuK[1] and AmaroK[2]. This short usability report reviews the second most popular music player application for Linux, which is AmaroK[3].
The purpose of this study was to find possible improvements in the graphical user interface of AmaroK. A specific usability inspection method was chosen for this study: the heuristic evaluation, which will be discussed in detail later on.
The application has only been roughly reviewed. Most parts only glanced, other parts weren’t examined at all, such as the icons. This leaves material to be examined in the future.
Although a development version was used to conduct the evaluation, there Amarok Wiki and Bugzilla were visited in order to ensure no duplicate work would be carried out.
Please note that during the evaluation no users were involved. Some issues are merely assumptions, but do have some ground based on the heuristics. Some issues could be tested for correctness by conducting a user-based usability test with the corresponding use cases.

2. Methods

Heuristics, also called guidelines, are general principles or rules of thumb that can guide design decisions (Nielsen, 1990). Heuristic evaluation allows you to catch problems that task-oriented methods, e.g. cognitive walkthrough, would miss. The procedure is based on the observation that no single evaluator will find every problem with an interface, and different evaluators will often find different problems.
Not all problems will be found with this method. It is possible to detect all major problems within an interface that are "heuristically identifiable" with 3 to 5 usability experts, but they can catch 75 percent of the total heuristically identifiable problems. That is, problems with the interface that actually violate one of the ten heuristics (Nielsen, 1994).

2.1 Heuristics

Here are the ten heuristics Nielsen (Nielsen, 1994) comprised from 249 usability problems:

Visibility of system status

[1] http://developer.kde.org/~wheeler/juk.html
[2] http://amarok.kde.org/
[3] http://amarok.kde.org/content/view/46/66/

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Match between system and the real world
The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

User control and freedom
Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Consistency and standards
Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention
Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Recognition rather than recall
Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use
Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Aesthetic and minimalist design
Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Help users recognize, diagnose, and recover from errors
Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Help and documentation
Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

2.2 Test setting

The heuristic evaluation was conducted on a PC running Fedora Core 5. I have installed KDE 3.5 with AmaroK 1.4-beta3. I have used the Plastik theme and the Crystal icon theme.

2.3 Procedure

As already mentioned, this usability test doesn’t involve users. The user interface will be examined by looking for compliancy with the heuristics described above. In addition, I will be using other (scientific) literature where needed.
Every design issue will be formatted in the style of a KDE usability report (Ellsworth, 2003), to enhance readability. This format will also make reproducing and correcting the problem easier.

3. Results

3.1 Technical term used in heading of the first step of the First-Run Wizard dialog

Severity: medium
Description: After the welcome screen in the First-Run Wizard dialog, the user is presented with the first step of the wizard where the user has to choose between two window layouts (fig. 1). However, this step is titled “Interface (1 of 3)”. The word “interface” isn’t common for all users.
Solution: Change the word “interface” into something like “appearance”, or rather describe instead of only a noun: “Choose an appearance”.
Rationale: Speak the user’s language.

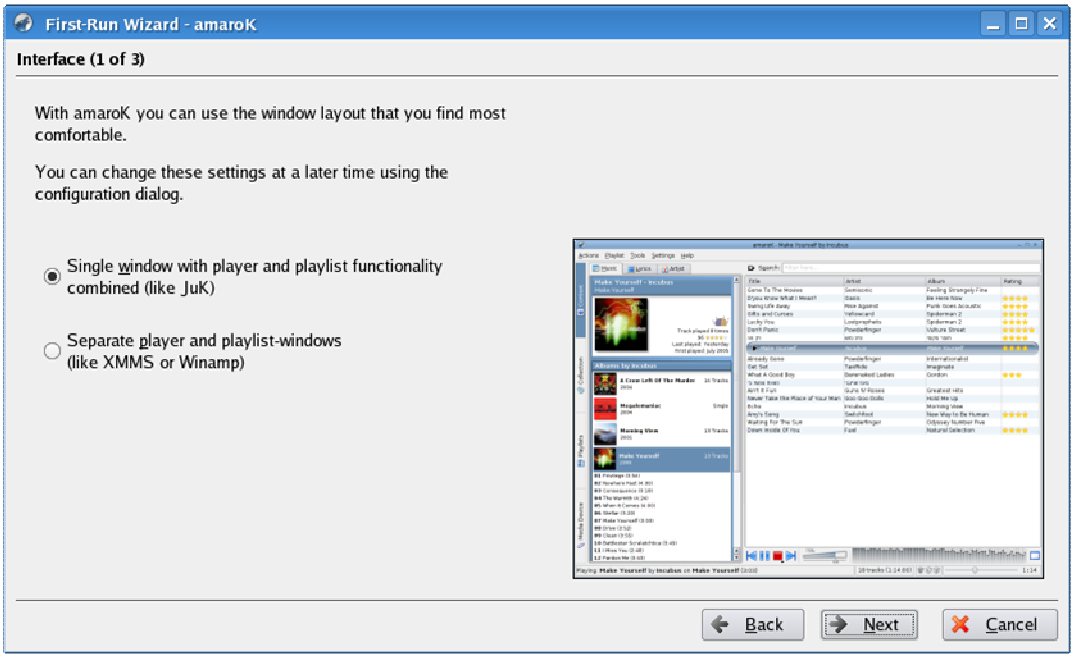


Figure 1: Luckily a picture can say more than thousand words; what are JuK and XMMS?

3.2 Reduce ambiguity in heading of First-Run Wizard dialog

Severity: medium
Description: After the welcome screen, the user has to complete three steps in order to finish the First-Run Wizard and start using AmaroK (fig. 1,3,4). These steps are titled in the top of the window. However, behind the first word, there is an indication of something, but it’s not clearly referring to what it exactly represents. For instance, the first step where the user has to choose between two window layouts. It’s titled “Interface (1 of 3)”, where the user doesn’t exactly know to what “3” is referring to.
Solution: Change the heading by adding the word step and also change the order words like this: “Step 1 of 3: Choose appearance”.
Rationale: Reducing lingual ambiguity will result in an interface that’s easier to understand.

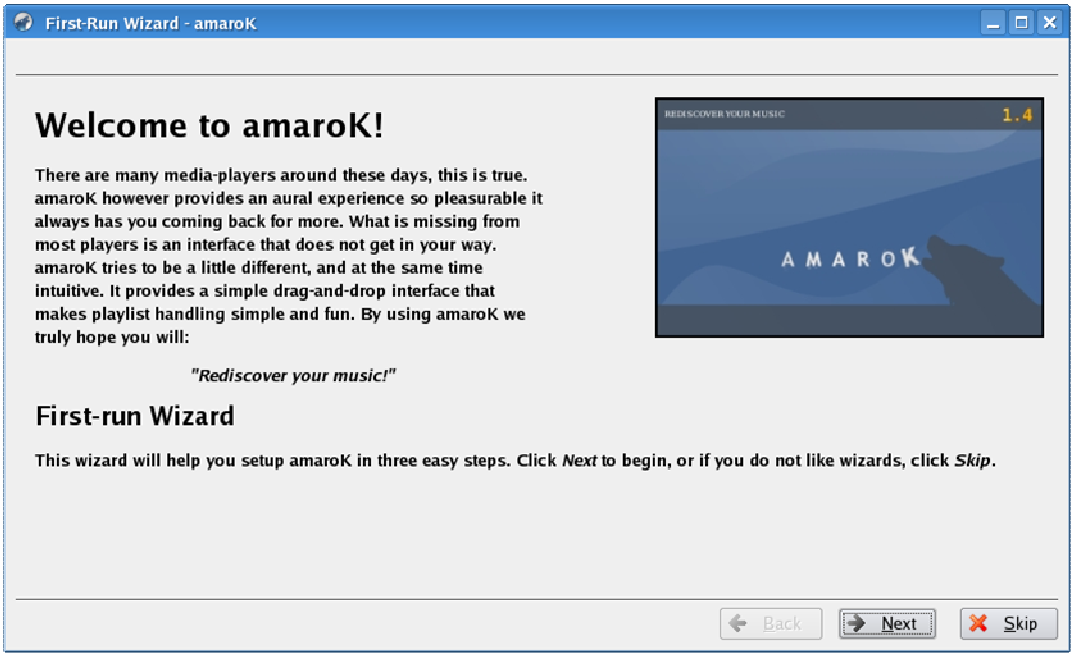
3.3 Use a more common word for the Collection concept.

Severity: low

Description: AmaroK has the ability to collect all your digital music. All the music that is collected by AmaroK is referred to as the “Collection”. However, the major music players on both Windows and Linux (iTunes[4], Winamp[5], Banshee[6] etc.) use the word “Library”.
Solution: Conform to the word that is most commonly used, in this case “Library”.
Rationale: Actually, in the real word, one would call all the music he possesses, his music collection, rather than music library. However, the library metaphor is linked to concepts such as searching, browsing, collecting, sorting and so on. Besides this metaphorical advantage, research has shown that when 80% or more of big sites do things in a single way, then this is the de-facto standard and you have to comply. Only deviate from a design standard if your alternative design has at least 100% higher measured usability (Nielsen, 1999).

3.4 Unnecessary use of bold text in First-Run Wizard

Severity: low
Description: In the first screen of the First-Run Wizard, the user is presented with some welcoming text (fig. 2). However, there is bold text in the body type. What’s even more worse, is that the remaining screens don’t use bold text at all.
Solution: Avoid bold text in body type.
Rationale: Extended bodies of text in italic are quite difficult to read, and bold type competes with headings for attention. Use italic and bold only for emphasis, and with small amounts of text (McCracken, 2004).



[4] http://www.apple.com/itunes/overview/
[5] http://www.winamp.com/player/walkthrough.php
[6] http://www.banshee-project.org/Guide/Listen

3.5 Technical term used in first step First-Run Wizard

Severity: low
Description: During the first step of the First-Run Wizard, the user has to choose which window layout he prefers (fig. 1). The word “window layout” may not be common for all users.
Solution: Use a word that is more commonly used, and within the vocabulary of the user, such as “appearance”.
Rationale: Speak the user’s language.

3.6 Unable to change window layout in the configuration dialog

Severity: high
Description: In the first step of the First-Run Wizard, the user can choose between two window layouts (fig. 1). The wizard also suggests that you can change the window layouts later in the Configuration dialog. This is not true, once a specific layout has been chosen, it will stick to this layout, unless you re-run the First-Run Wizard. This has implications for the main window, because it will have a different set of buttons beneath the playlist for each window layout.
Solution: Provide a configurable option for this feature in the Configuration dialog.
Rationale: This is obviously a software bug.

Figure 2: A bold welcome

3.7 Filesystem is shown in the second step of the First-Run Wizard

Severity: high
Description: KDE hides the filesystem structure by using more meaningful shortcuts, such as “Home”. Users might not be aware how the filesystem is structured (fig. 3).
Solution: Show only the home folder and its contents. Apply progressive disclosure (Apple, 2006) and include an advanced importing dialog, which is actually the dialog that is currently used.
Rationale: Keep it simple and try to avoid as much technical information as possible.

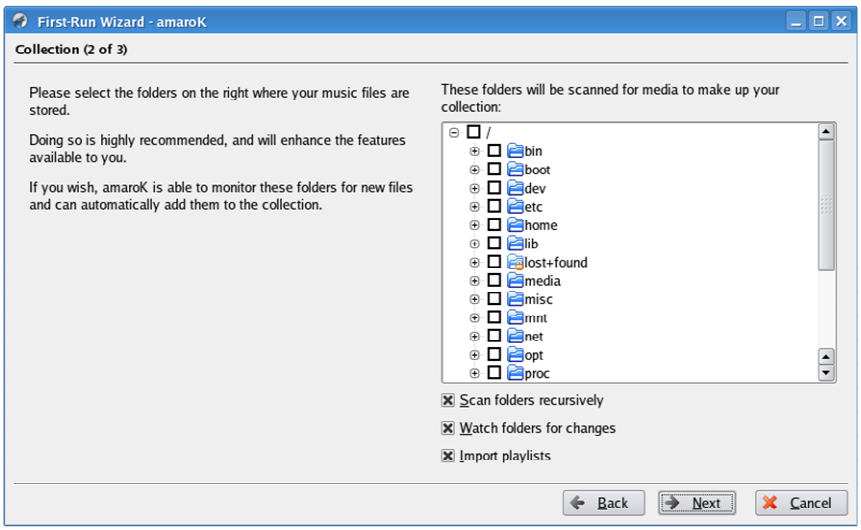


Figure 3: Second step of the wizard

3.8 Technical information in third step of the First-Run Wizard

Severity: high

Description: The third step of the First-Run Wizard requires the user to choose a database type (fig. 4). This information is highly technical and should not be shown to the user.
Solution: Remove this step from the wizard.
Rationale: Wizards are used to help users who are not experienced doing specific tasks. Choosing between different database systems is obviously specific for experienced users. Don't confront novice users with this. Hide technical information from the user.

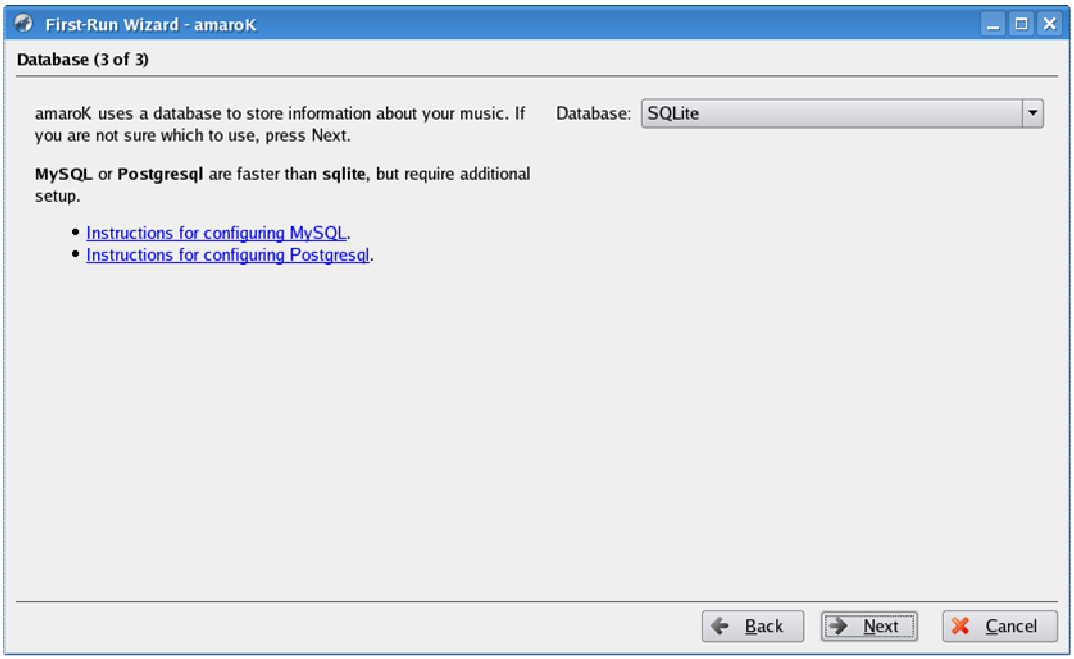


Figure 4: The database wars

3.9 Inconsistent placement of headings in First-Run Wizard

Severity: low
Description: During the First-Run Wizard, the heading is changing places. In the welcome screen, the heading is placed underneath the horizontal separator, whereas during the setup steps, the heading is placed above the separator.
Solution: Place all headings above the horizontal separator.
Rationale: When a consistent layout is maintained, the user can expect where are placed.

3.10 Low contrast of headings in First-Run Wizard

Severity: low
Description: There is not much contrast between the headings and body text/content, except for the welcome screen.
Solution: Use a white background for the heading with an increased line-height. This can be seen in a lot of wizards.
Rationale: Use visual cues to increase or decrease visibility of interface elements.

3.11 Unnecessary use of horizontal separators in the First-Run Wizard

Severity: low
Description: In the First-Run Wizard, the heading, body content and buttons are separated by two horizontal separators (fig. 1-4). These separators are not necessary, because other visual cues that are visually less interfering can be used.
Solution: Use the Gestalt principle 'proximity' to separate the elements or use separators with a lighter color.

Rationale: Keep a clean and minimal design.

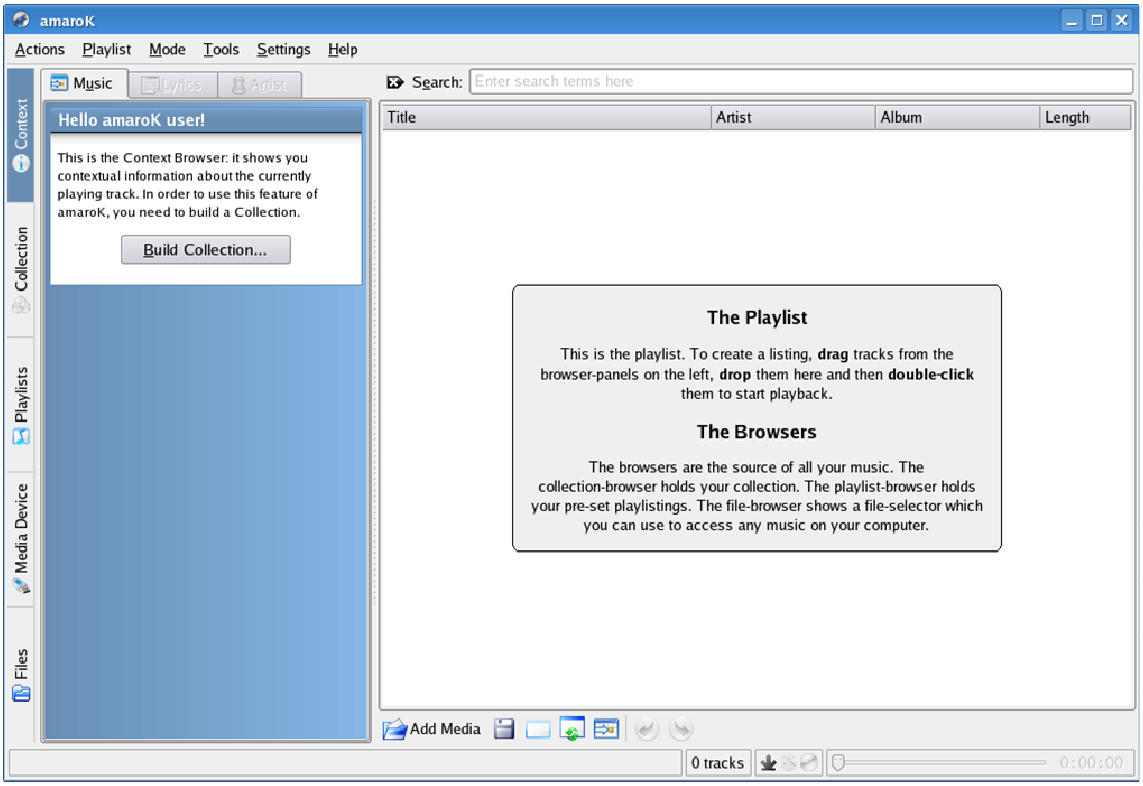


Figure 5: The main window

3.12 Vertical tabs

Severity: high
Description: A vertical tab bar is in place in the left of the window (fig. 5). The tabs on this tab bar are labeled with vertical text. This decreased readability. Another problem is it relatively small width, which makes it harder to click (Murata, 1999).
Solution: A clear-cut solution has not been found yet. There are also more problems with the browsers, which will be addressed later.
Rationale: Horizontal text is faster for horizontally arranged text than for vertically arranged text by 24% (Seo & Lee, 2002). Seo did not mention the implications for the human memory when vertically arranged text where to be memorized. I haven't found a report that already researched this. A much earlier experiment by Wanner (1968), suggested that people usually remember just its meaning and not its exact wording. Mandler and Ritchey (1977) found out that when people see a picture, they tend to remember a meaningful interpretation. Thus, it would seem that memory for both horizontally and vertically arranged text are the same. However, recognition memory should be better for horizontally arranged text under the same exposure time.

3.13 Large width of search bar (playlist)

Severity: medium
Description: Above the playlist in the main window, there's a search bar available which allows the user to search through the playlist (fig. 5). However, this search bar has almost the same width as the playlist, which is a bit excessive, since most search terms will never span the entire width.
Solution: Shrink the search bar to fit the length of the average search term.
Rationale: Not every pixel on the screen has to be occupied. Use white space, i.e. not used space, to create a clean and peaceful environment for the user to interact with.

3.14 Unusual menu organization

Severity: high
Description: The menu bar is not well organized (fig. 5). For instance, the Actions menu is highly ambiguous, since all menu items imply a certain action. Due to the lack of guidelines for this, the menu organization is inconsistent throughout KDE. Therefore, I will not go into further detail, since there are no references I can use within KDE.
Solution: A new set of interface guidelines will be created for KDE 4. Apply these guidelines when available.
Rationale: not applicable

3.15 Unusual placement of control buttons

Severity: high
Description: The player control buttons (next track, play/pause, stop, next track) are placed below the playlist in the bottom-right corner (fig. 5), which makes it harder to find and look less important.
Solution: Place important elements in the top-left corner of the window. This guarantees maximum exposure, since the eye tends to naturally moves toward it.
Rationale: When considering the window as a flat, fixed-size canvas, halving it both horizontally and vertically creates four quadrants. The sentences of Western languages run for left to right across a page, and top to bottom down the page. The average viewer has learned to unconsciously regard each quadrant with a different importance because of this (Shea et al., 2005).

3.16 Inconsistent font size in context browser

Severity: high
Description: The font size in the context browser isn't the same as the rest of the interface. It disobeys the system font size and thus, inconsistent.
Solution: Maintain the same font-size throughout the interface. Use at least a 10-point type.
Rationale: The font size in the context browser is smaller than the text in the rest of the interface. Font sizes smaller than 10 points will slow a person's reading speed (McCracken, 2004).

3.17 Unnecessary usage of clear button of the search bar

Severity: medium
Description: Next to each search bar, there's a clear button on the left. The use cases for this clear action are pretty weak, as argued by Yanis Kekatos and Aaron Seigo on the KDE-usability mailing list[7].
Solution: Remove this button.
Rationale: Keep the user interface as clean and minimal as possible.

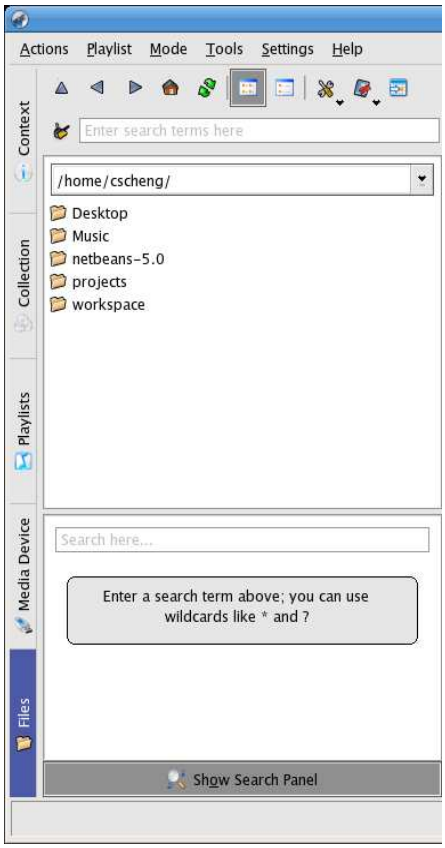


Figure 6: One search to rule them all

[7] http://lists.kde.org/?l=kde-usability&m=114532406509037&w=2

3.18 Unclear extra search functionality in button in Files browser

Severity: high
Description: In the top of the Files browser, there's a search facility to search in the folder that the user is currently browsing. In the bottom, a button can be found and when clicked, the user is shown a different search panel which can be used to search (fig. 6). At this point, three search bars are visible in the main window, which is too much of a good thing. Besides, the extra search panel has a better search functionality than the upper one, so why keep both?
Solution: Remove the upper redundant search bar.
Rationale: Keep the interface as clean and minimal as possible. Avoid redundant information on the screen.

3.19 Unnecessary use of Files browser

Severity: high
Description: AmaroK is designed around the "Collection". Thus, the preferred way is to add music from a certain source to the AmaroK music collection and then select and play the music from that collection. However, AmaroK is of course also able to play music that is not in the "Collection". Besides using Konqueror (or any other file browser) and locating the music file desired to play, AmaroK also offers a built-in file browser to facilitate this action. The use of this built-in file browser is questionable, since the user can already use facilities of the desktop.
Solution: Remove this built-in browser.
Rationale: It doesn't make sense to provide functionalities that are already available in the desktop. Keep the interface as clean and minimal as possible.

3.20 Awkward shortcut for "Go to current track" in Playlist menu

Severity: medium
Description: In the Playlist menu, there's an item called "Go to current track". This action also has a keyboard shortcut, that can be activated by pressing "Ctrl+KP_Enter" (fig. 7). However, where's the "KP_Enter" key?
Solution: Change the name or keyboard shortcut.
Rationale: Use mnemonics where possible for keyboard shortcuts.

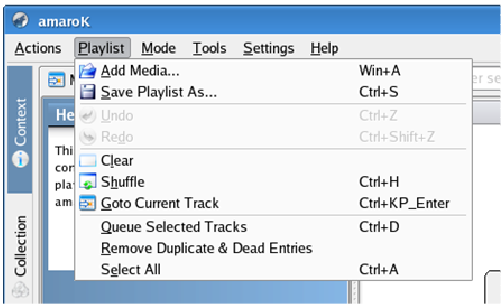


Figure 7: What's KP_Enter?

3.21 Default window size of Play Media dialog

Severity: high
Description: Via the Actions menu, the user can access the Play Media dialog to select and play a certain music file (fig. 8). The dialog window is very small, and the items in the list view are placed very close to each other.

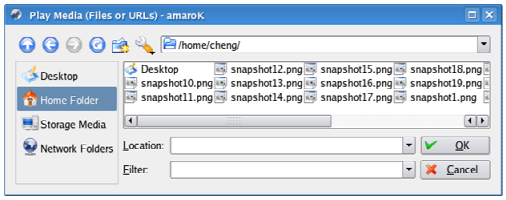


Figure 8: Notice the absence of white space

Solution: Set the default window size a bit larger. Also use the icon view instead of the default list view, to enhance contrast between the visual elements (icon+label).
Rationale: Research has shown that the preferred spacing between icons is ½ icon width and the perceptual span is 5 x 5 icons (Lindberg et al., 2003). Utilize this knowledge to enhance visual search. Also, the user should be able to execute a certain action, in this case

opening a media file, with minimum effort. The default settings should be desirable for most users.

3.22 Incorrect window title of Play Media dialog

Severity: low
Description: The window title of the Play Media dialog (fig. 8) is “Play Media (Files or URL)”. It implies that you can also open URLs via this dialog. Actually, another type of dialog is needed for opening URLs.
Solution: Keep only “Play Media”.
Rationale: Don’t misinform the user.

3.23 Extraneous information in Lyrics tab of Context browser

Severity: high
Description: When the user tries to view lyrics of a certain song, while there are no lyrics scripts running, the Lyrics tab of the Context browser gives the user some feedback. However, this feedback contains highly technical information and should not be shown to the user (fig. 9).
Solution: Either enable the lyrics scripts by default, or hide the Lyrics tab by default.
Rationale: It doesn’t make sense to show the Lyrics tab, when the lyrics script are not running and therefore impossible to show any lyric.

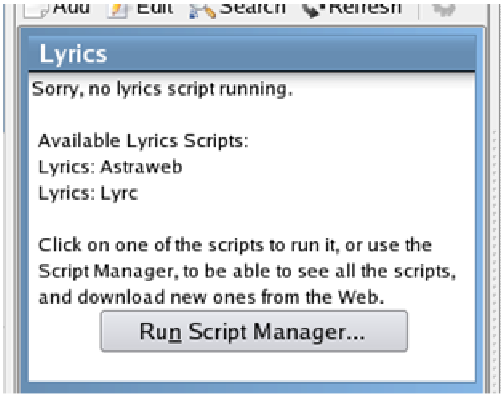


Figure 9: You’re sorry?

3.24 Unusual placement of seeking bar (playing progress)

Severity: high
Description: The seeking bar and time elapsed indicator are placed in the bottom-left corner. This corner is often used to display either redundant or unimportant information. Thus, important interface elements shouldn’t be placed here.
Solution: Place the seeking bar and the time elapsed indicator close to the current track information.
Rationale: The Gestalt principle “proximity” states that humans tend to perceive closely clustered objects as a group. Place related elements close to each other, to match the user’s expectations.

3.25 Unnecessary use of status bar in Cover Manager

Severity: medium
Description: The status bar in the Cover Manager dialog, which is also has a strange placement since it doesn’t span the entire window width, doesn’t fulfill a useful purpose. It only shows the name of the selected album, which is redundant because you can already see it when selected. Even if the selected track is not visible in the album view, for instance the user scrolled down, the selected album information has no purpose, since the user cannot execute any action with it.
Solution: Remove this status bar.
Rationale: Keep the interface as clean and minimal as possible.

3.26 Technical information shown in Cover Manager

Severity: high
Description: In the top of the Cover Manager dialog, above the album view, there's a dropdown-list button labeled "Amazon Locale" (fig. 10). This has no meaning for users that don't know or understand that AmaroK fetches album covers of Amazon.com.
Solution: Remove this button.
Rationale: Don't show technical information about the inner workings of the application to the user.

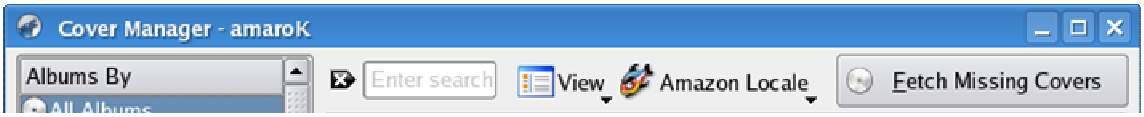


Figure 10: The Cover Manager

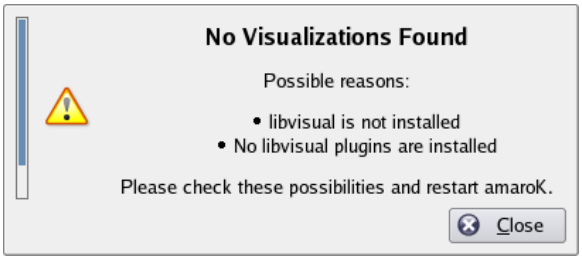


Figure 11: Is AmaroK too lazy to do that itself?

3.27 Visibility of error dialog unavailability visualizations

Severity: high
Description: When visualizations are unavailable, for any reason whatsoever, the user is presented with an error dialog (fig. 11). This error message is only visible for a few seconds, before it hides. This amount of time is not enough for both discovering the error dialog and reading the error message. The timer will stop if the user moves his cursor over the error dialog, but there are no visual cues to make the user aware of this.
Solution: Don't use a timer to automatically hide the error message.
Rationale: The system should keep the user informed about its status at all time, especially in case of an error. Provide understandable information that can help the user recover from this error.

3.28 Unusual text alignment of error dialog unavailability visualizations

Severity: low
Description: When the user tries to enable visualizations, while they are unavailable for any technical reason whatsoever, the user is presented with an error dialog (fig. 11). In this dialog, the text is center aligned, which results in increased virtual illegibility.
Solution: Apply left alignment on the text.
Rationale: Aligning items tells the viewer that items are related. This is an implication of the Gestalt principle "alignment". Centered alignment is the weakest type of alignment.

3.29 Vague error message when trying to enable unavailable visualizations

Severity: medium
Description: When the user tries to enable visualizations while they are unavailable, AmaroK responds with an error dialog (fig. 11). This dialog doesn't contain helpful information however.
Solution: Change the error message into something like: "There are not visualization plug-ins installed. Click here to get visualization plug-ins".

Rationale: Provide understandable information to help the user recover from errors.

3.30 Uncommon name of concept for Script Manager

Severity: medium
Description: The user can add extra functionality to AmaroK even after the application has been installed. These pieces of extra functionality are called “Scripts” in AmaroK. However, most major music software players uses a different word to refer to these extra functions.
Solution: Use the word “plug-in”, which is used by applications like Winamp[8], Windows Media Player[9] and Banshee[10] for instance.
Rationale: Make use of the experience of the user with similar applications where possible, to ease the learning process.

3.31 Search bar doesn’t work in Collection Statistics

Severity: high
Description: The search bar in the top of the Collection Statistics dialog doesn’t work. Since there’s no search button, and pressing “Enter” has no effect, the function of this search bar isn’t clear.
Solution: Either remove the search bar, or provide feedback by informing the user what is wrong. Or show the user with the number of search results. At least give some information to the user.
Rationale: Always keep users informed of what is going on.

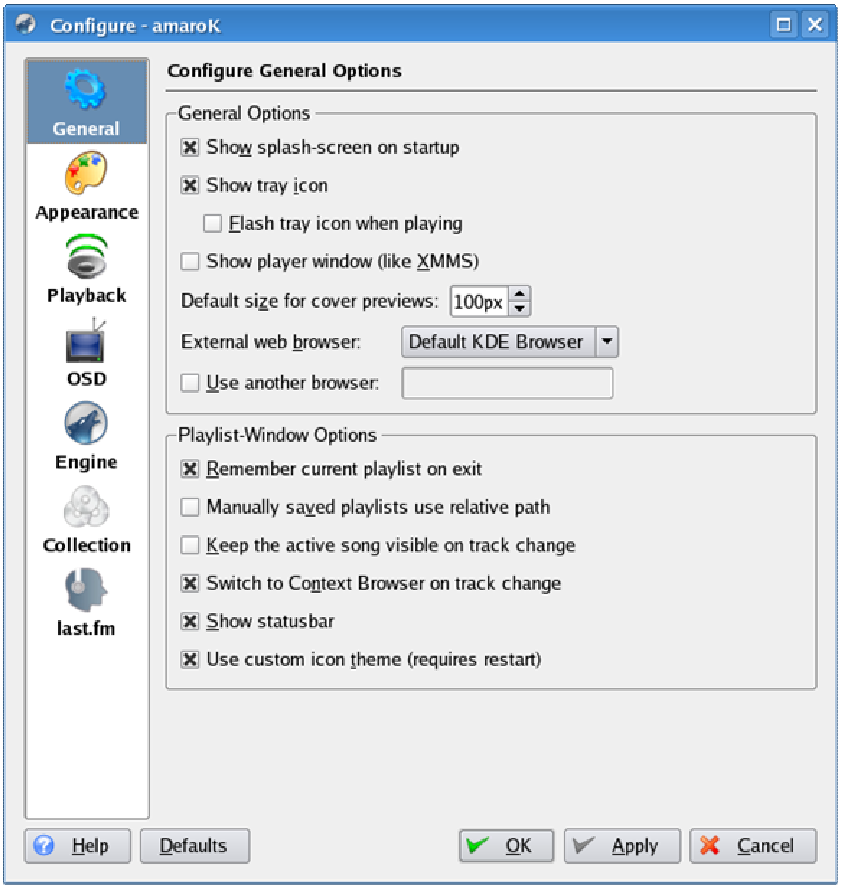


Figure 12: The Configuration dialog

[8] http://www.winamp.com/player/walkthrough.php
[9] http://www.microsoft.com/windows/windowsmedia/mp10/getmore/plugins.aspx

3.32 Unnecessary bold text in list boxes

Severity: medium
Description: In many dialogs, such as the Configuration dialog (fig. 12), there's a list box with icons on the left. This list box serves the purpose of a tab bar (also known as tab box[11]), even though it doesn't look like an array of tab elements. This is a different problem, which will be addressed later. The point at issue here, is that the icons in these list boxes are labeled with bold text.
Solution: Don't make the text bold.
Rationale: Use bold only for emphasis. When bold text is too often used, it will lose its emphasis.

3.33 Last.fm icon in Configuration dialog doesn't match icons/graphics Last.fm

Severity: medium
Description: The Last.fm icon in the Configuration dialog (fig. 12) doesn't match the Last.fm logo or graphics derived from it[12].
Solution: Create an icon based on Last.fm logo/graphics.
Rationale: Rely on the user's ability to recognize, not to recall.

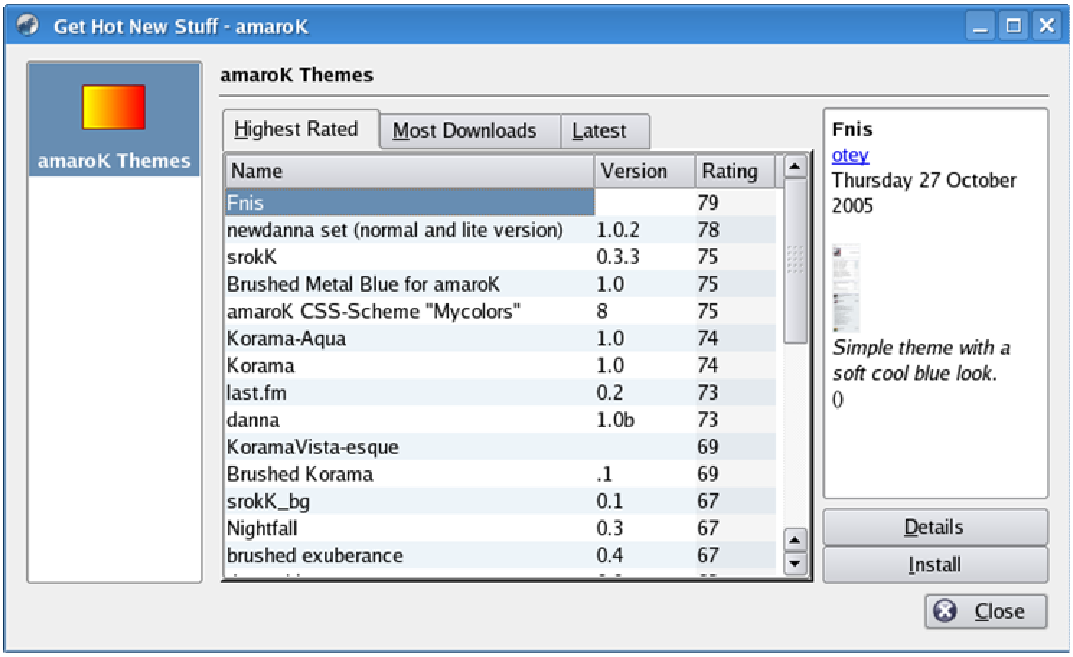


Figure 13: You need a magnifying glass to see the hot new stuff

[10] http://www.banshee-project.org/PluginRepository
[11] http://www.xulplanet.com/tutorials/xultu/tabpanel.html
[12] http://www.last.fm/onyoursite/banners.php

3.34 Unnecessary use of list box in Download Styles dialog

Severity: high
Description: In the Download Styles dialog (fig. 13), there's a list box with icons on the left. However, there's only one item in this list box, and therefore pretty useless.
Solution: Remove this list box, since it doesn't have a function.
Rationale: Keep the interface as clean as possible.

3.35 Unnecessary use of column "Version" in list view of Download Styles dialog

Severity: medium
Description: In the Download Styles dialog (fig. 13), the user is presented with a list of downloadable styles. In this list, there are several items visible for each style, namely the name of the style, the version number and the popularity rating. Placing the version number of a style in a separate column is useless, since sorting on version number is of no meaning, in contrast with sorting the name or popularity rating.
Solution: Remove the version number column. If information about the version of a style should still be available to the user, apply progressive disclosure by hiding version information in the Details dialog.
Rationale: Keep the user interface as clean as possible. When there are a lot of interface elements on the screen, there is little free space to play around with for creating clarity and contrast.

3.36 Too small preview of style in Download Styles dialog

Severity: high
Description: In the Download Styles dialog (fig. 13), the user can see a preview of the selected style. However, the preview is very small and therefore useless, since the user can't really see what the style looks like.
Solution: There are several solutions. The first one is to create a larger thumbnail of the entire image. The user would have to scroll to see the whole thumbnail, but it would be a better solution than the current one. Another alternative is to create a thumbnail of the upper portion of the image, so the interface wouldn't suggest the user to scroll (because of the scrollbars), since there isn't more information to display. A third possibility would be to create a preview with the exact size in different window or in a larger panel.
Rationale: If the user can't see the preview, he cannot decide whether or not to download it.

3.37 Awkward dialog when opening AmaroK handbook

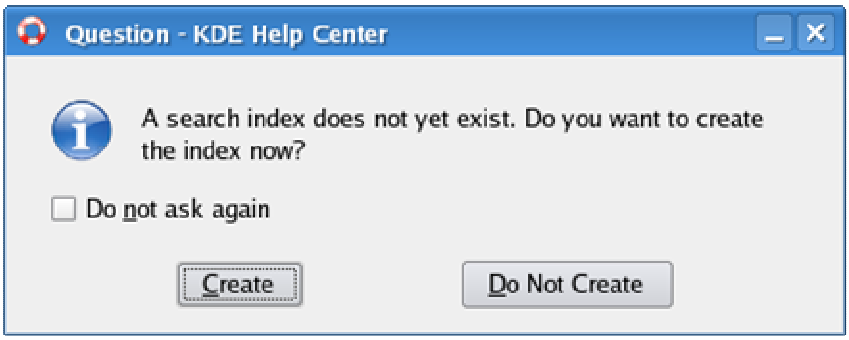


Figure 14: What happened to the 'Ok' and 'Cancel' buttons?

Severity: high
Description: The first time the user starts the AmaroK handbook, the user is asked to create a search index (fig. 14). The question is technical oriented, and does not have any

meaning for the user. Above that, the dialog doesn't say what the consequence is of creating or not creating a search index.

Solution: If the user has to be empowered with the choice of creating/not creating a search index, or in other words have the ability to search in the manual, then this dialog will be necessary. Still, it has to be changed in order to make sense for the user. However, searching through the help should always be possible, and the user should be able to search by default.

Rationale: Speak the user's language.

3.38 Clumsy handling of removable media devices

Severity: high

Description: When a removable device, such as an mp3-player is plugged in, AmaroK reacts by showing a dialog. This dialog titled "Removable Medium Plugin Chooser" asks the user to choose what plug-in should be loaded in order to be able to transfer music from and to the device. However, this dialog contains highly technical information. Don't expect that the users knows what a "Generic VFAT Media Device" is (fig. 15).

Solution: The ideal solution would be automatic detection of the device that has been connected, and that the system responds appropriately. An elaborated study for a possible solution is necessary for a detailed solution.

Rationale: Speak the user's language.

3.39 Unclear status of connection with removable media device

Severity: high

Description: Once the user has chosen the right plug-in for connecting a removable media device, AmaroK doesn't react properly (fig. 16). As can be seen in the Removable Media browser, the Connect button is not activated and the list view does not show any file, indicating that there's no device connected. The user need to click manually on the connect button, before he can enjoy music file transfer.

Solution: Automatically connect the device once connected.

Rationale: When a user plugs in a removable device, it is common to think that this device is connected to the computer. The user may be surprised to see that the device hasn't been connected at all, even though the system detects it.

3.40 Can't transfer music using the menu

Severity: high

Description: To copy some music from the collection to an mp3 player, the user has to right-click the music file, and add it to the Transfer Queue. However, this action is only available from the context-menu.

Solution: Add it to the main menu.

Rationale: Actions that are in the context-menu, should always be accessible from the main menu.

3.41 Transfer Queue panel actions are context-menu only

Severity: high

Description: Once music files are added to the Transfer Queue, there's a new panel visible in the Removable Media Device. This Transfer Queue Panel contains a list of music files that are to be transferred to a removable device. However, when the user wishes execute an action with a certain music file (for instance remove from the queue), he can only do so from the context-menu.

Solution: Provide the context-menu actions also in the main menu.

Rationale: Actions that are in the context-menu, should always be accessible from the main menu.

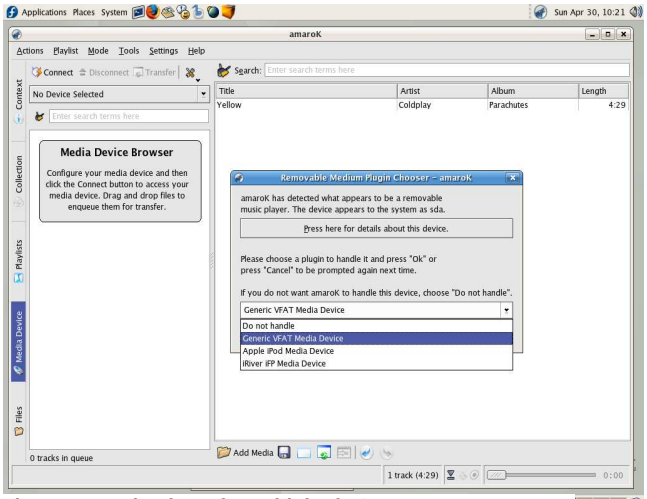


Figure 15: Is that low vfat or high vfat?

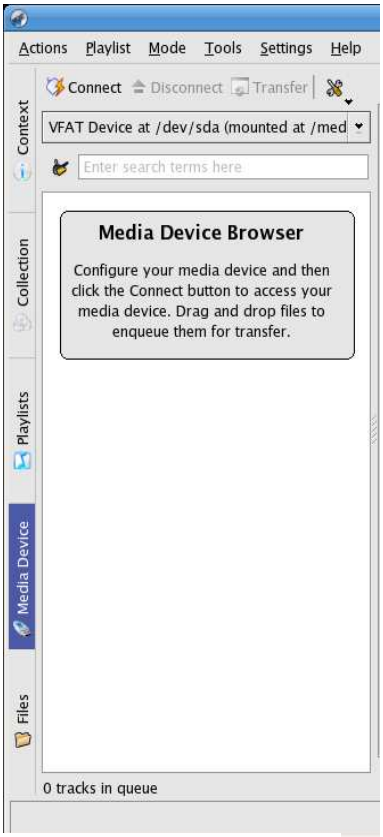
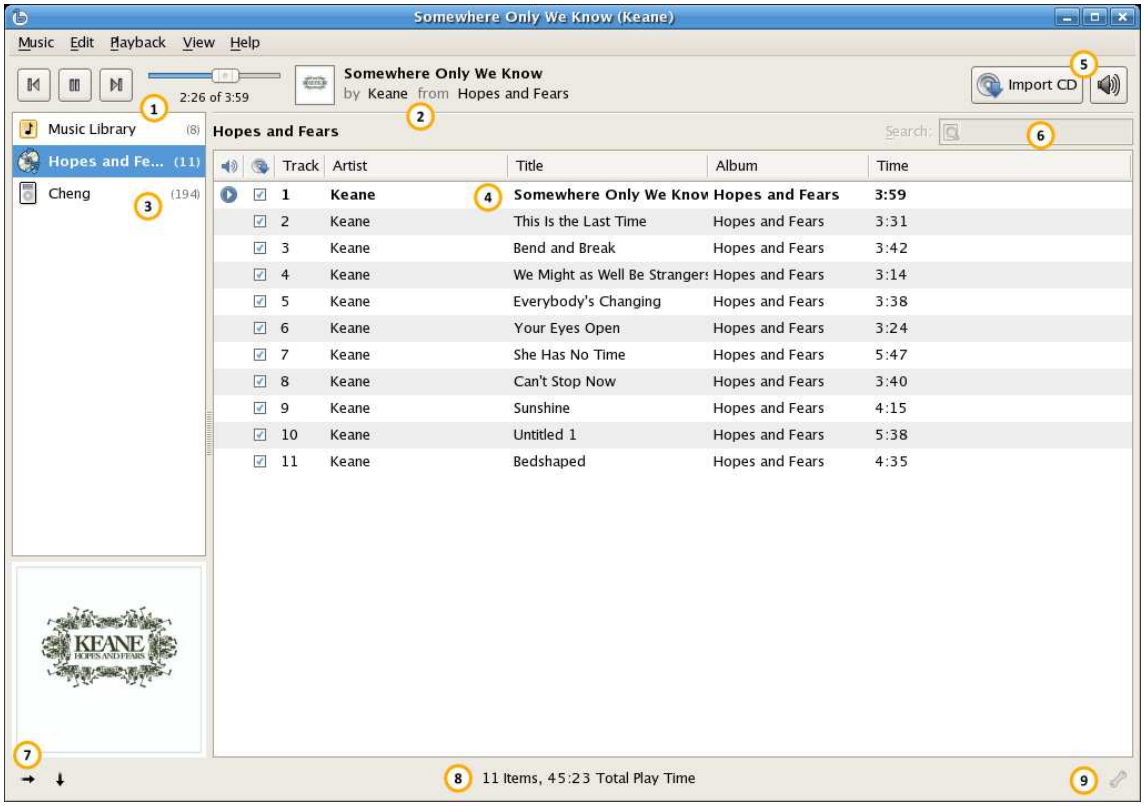


Figure 16: Notice the amount of info crammed in the drop-down listbox

4. Discussion

After a rough evaluation, a total of 41 issues were found. For each issue that has been found, its severity has been graded. Only 8 issues of low severity were found, 11 were marked medium, and the remaining 22 issues considered highly severe.
The heuristic evaluation consisted mostly of creating screenshots of every part of the interface. Afterwards, every screenshot was examined and every issue found was noted. The behavior and usage of the application was minimally tested. User testing is needed to get accurate and reliable results about task-oriented usability issues.
Only a very few issues were related to software bugs, i.e. most issues were related to wrong design decisions. The application also didn't crash during testing. The back-end seemed reliable and did its job. However, it wasn't the back-end that was subject to the test, it was foremost the front-end that suffered from a great amount of usability design issues.
The amount of highly severe issues that have been found after a quick look is quite worrisome. Are similar open-source applications suffering the same problem? Here's Banshee's UI design review in a nutshell:



1. Important elements are placed in the top-left corner for increased visibility.
2. Contrast is created by coloring the words "by" and "from" lighter. Thus, visually giving more attention to the other words.
3. Again, contrast created by differing font color. Notice the icons helping the user to identify the music source.
4. No animation is used to highlight the current track. Simple and effective methods suffice here.
5. Placed in a corner to contrast with the rest of the interface. Makes it easier for the eye to distinguish the shape. Visual elements placed according to their importance in this specific corner.
6. One search bar is sufficient.
7. Same as 5.
8. Lots of white space in the status bar to enhance clarity and contrast between visual elements.
9. Same as 5.

It is important to note that Banshee is developed in-house by Novell, and thus the developer of this application probably has a usability expert team at his disposal. Unfortunately, the AmaroK developers don't have this luxury and have to cope with this difficulty.

5. Conclusion

Based on the results of the heuristic evaluation, more attention should be given to the user interface of AmaroK. The application itself is fairly stable and has the basic features every music management software should need. Beyond those basic needs, AmaroK also provides more innovative features such as the context browser, which allows the user to see related information about the song he's currently listening. The developers should however prioritize their development efforts. AmaroK has the technical potential of being an outstanding open-source music management application, but it would be great to see this would also being expressed in its user interface.

References

Apple. (2006). *Apple Human Interface Guidelines*. Retrieved 1 May, 2006, from: http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/OSXHIGuidelines.pdf

Ellsworth, E. (2003). *A brief introduction to writing a usability report*. Retrieved 27 April, 2006, from: http://usability.kde.org/activity/usabilityreports/howto/

Lindberg, T., Näsänen, R. (2003). *The effect of icon spacing and size on the speed of icon processing in the human visual system*. Displays, issue 24.

Mandler, J.M., Ritchey, G.H. (1977). *Long-term memory for pictures*. Journal of Experimental Psychology: Human Learning and Memory, issue 3.

McCracken, D.D., Wolfe, R.J. (2004). *User-centered website development*. New Jersey: Pearson Prentice Hall.

Murata, A. (1999). Extending *effective target width in Fitts' law to a two-dimensional pointing task*. International Journal of Human-Computer Interaction, vol. 2 issue 11.

Nielsen, J. (1990). *Heuristic evaluation of user interfaces*. Proc. ACM CHI'90 Conf.

Nielsen, J. (1999). *When bad design elements become standard*. Retrieved 27 April, 2006, from: http://www.useit.com/alertbox/991114.html

Nielsen, J., Mack, R.L. (1994). *Usability inspection methods*. New York: John Wiley & Sons.

Seo, H., Lee, C. (2002). *Head-free reading of horizontally and vertically arranged texts*. Vision Research, issue 42.

Shea, D., Holzslag, M.E. (2005). *The Zen of CSS Design*. Berkely: New Riders.

Wanner, H.E. (1968). *On remembering, forgetting, and understanding sentences: A study of the deep structure hypothesis*. Unpublished doctoral dissertation, Harvard University.