Dialog usability

Dialog boxes simply put are a way of communicating between the system and the end user, i.e. to establish a dialog between the two parties as it were. Dialog boxes could be many but primarily the intent of the dialog boxes are two, notifying an end result of any given task or to accept input before performing a task from the user.

At a macro level dialogs are modal (which block interaction on the UI) or modeless (which do not block interaction and generally goes off quickly enough). More details can be had from this Wikipedia link

From a usability point of view, there are several things that need to be taken into account while using dialogs.

Be consistent

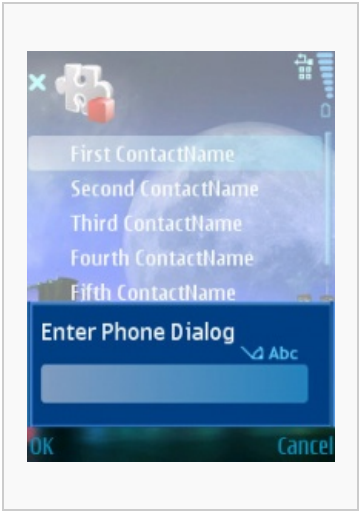
It is imperative to be consistent with the usage of dialogs. One should decide when and where to use a dialog, as the dialogs would generally pop up or open up a new form/window to the user, this could annoy the user if not used properly. Use the dialogs only when it is absolutely essential to, because it generally breaks the usual flow of events or the user experience. Some instances where using a dialog makes sense are, receiving inputs/settings from the user which are not in line with what is already being displayed on the view, showing a notification of an activity where simply switching views might confuse the user.

Also once you decide on using a dialog for a particular case make sure you use one in all other logical/similar cases, for instance input choices etc. Don't show a dialog in one case of a connection error while in another case simply change some icons or add content to the log file.

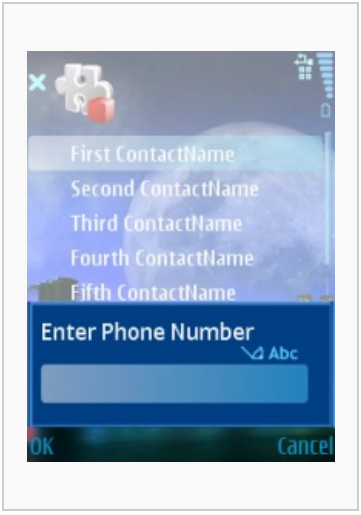
Keep your terminology the same throughout the application

Use the same terminology for the use of the dialogs throughout the application i.e. don't use different names i.e. if you decide to name phone entry dialog heading as "Enter Phone Dialog" don't name another dialog where you are entering server address as "Server Address Input". This sort of nomenclature could end up confusing the end user.

It is generally a good idea to never call a dialog by its name, i.e. in the first instance you should just call it "Enter Phone Number" instead of "Enter Phone Dialog"



Example of incorrect dialog heading



Example of correct dialog heading

Remember feedback

It's important to tell the user when something is happening.

In cases where the user has requested some functionality which could take time, for instance downloading a media content or uploading something or sending SMS etc, it is always nice to keep the user informed of the state of the request. If it is possible a timer implementation in %, time remaining etc is the best but where it's not possible at least a progress note/bar/time glass should be displayed so that the user waits for a response. Generally if the UI doesn't inform the end user, the latter tends to believe that the application has stopped responding.

Make informative error messages

Informative error messages let the user know what happened and what needs to be done to recover (see Error message).

It is always imperative to treat the user with respect and help him/her out of a situation that could arise due to any errors happening because of something they have attempted to do. Do not simply show a generic dialog like “Error” without giving out the complete details of the error, and the wordings of the dialog should be from a lay man’s point of view, avoid using technical lingo/jargon which they may not understand.

Also if possible provide them easy to follow help steps/to dos to come out of that error situation. Another important thing to make sure is not to pop up error dialogs all over the place, make sure you fine tune your logic to minimize the use of such dialogs, as the more number of such error pop ups more the chances of the end user thinking they have purchased a buggy/not fully working product.

Example of wrong error dialog

Example of right error dialog

Make sure the user has the opportunity to reverse from any situation

Depending on the UI this might be a back button, cancel button, or simply a back arrow.

This is especially important in the case of long time taking tasks like network intensive activities, downloading/uploading content etc, as the chances are that sometimes due to the underlying network limitations the time taken becomes almost unbearable for the user and they would like to come out of that seemingly un-ending loop. Another thing to keep in mind is that when the user does cancel a task, the underlying activity is stopped/cancelled as well to ensure that in cases of airtime/GPRS usage the user doesn’t end up being charged even after they have cancelled the task.

Example of correct progress dialog

Example of incorrect progress dialog

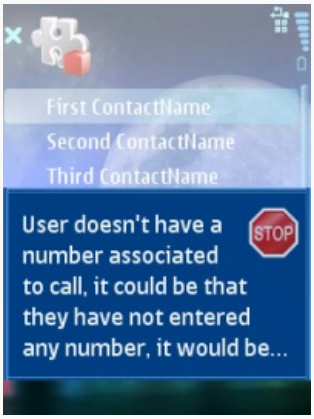
Follow the style guides for the usage of the positive and negative action option placements on the softkey. The S60 style guide can be downloaded from

S60 Style Guide

Keep the dialogs simple

It is important to no overload the user with too much information.

Information overload is never a good idea especially on the display area constrained mobile devices. You possibly don't want the user to scroll up/down endlessly in trying to make sense of the long/boring information/error/content being displayed. The choice of words and content is very important as more often then not the dialogs would be altering the usual flow of events from an end user perspective.



Example of incorrect dialog content usage

Example of right dialog content usage

Use the right dialogs at the right place

Always ensure that you are using the right dialog at the right place, for instance if you are reporting an error to the user, use the error note with exclamation marks or meaningful images, in the case of an information use the dialog with a positive meaning image like a tick mark etc. When you are giving the user a choice between a Yes/No ensure you use the binary dialog which has the correct options Yes/No, while when the intent is to let the user decide between 2 things/choices, it is always nice to put those words on the soft keys, for instance if you give the user a choice between Conference or Transfer, use them instead of just a plain Yes/No.

Do not use a modal dialog where a modeless dialog could do the job rather nicely, as the former blocks the UI/flow until the user has taken action on them. Some instances of this could be, a task completed and you want to notify the user but don't really expect them to say Ok/or take any action. But in case you are attempting something which could alter the normal flow then you should most likely ask the user, for instance deleting a file, using a different access point etc.

The simple ideas being you don't want to surprise/shock the user by displaying something on the dialog while the actual result/output could end up being totally different from what the user had expected.

Look and feel goes a great way in ensuring nicer usability

Use the right layout and fonts for displaying the content in the dialog. If you are displaying lot of content in the dialog ensure you use line breaks/paragraphs instead of just making it a continuous plain text outlay. The user doesn't mind reading through the dialog as long as it is presented in a nice manner. Ensure that you use the right images where possible as the adage goes "a picture can substitute for a thousand words" if used in the proper place.

In case of about dialogs it is always nice to put the company logo in the body/header of the dialog so that the user knows the provider. Providing contact details/website in the about dialog is also a good idea.

Related Links

- Dialogs
- Simple About dialog