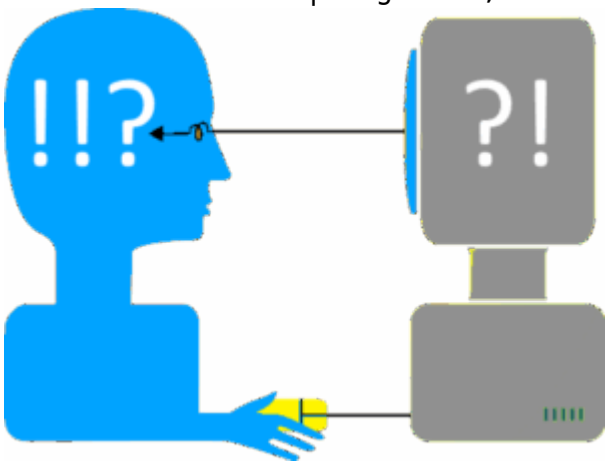
Software Usability

Category: LabVIEW Talk
scritto da admin | Luglio 4, 2018



As software developers we pay special attention to usability.

We chose an approach based on Nielsen's 10 heuristics usability of interfaces.

These derive from the application of factorial analysis techniques on 249 usability problems.

In this article we want to share our interpretation, with particular reference to our LabVIEW design environment.

1 – Visibility of system status

> The system must always keep the user informed of what he is doing, providing adequate feedback in a reasonable amount of time.

We have grouped the following categories of controls to help design sharing with customers.

System Log: System Log: A String Indicator should be useful to notify status of system also if system work properly. Here we can enter system responses to operator inputs (Aknowledge).

PopUP: to notify priority errors and request action from the user

Led Status: Notify constantly a state, for example connection of the system to the network.

Progress Bar: When there are many back ground operation, we choose to use a progress bar that gives visibility of how much time is to have the availability of the system.

Cursor : Hourglass to indicate the status of Busy.

DIsabling Controls: Avoid unwanted actions. (We use the disabled and grayed)

2 – Match between system and the real world

> The system must speak the user’s language, with words, phrases and concepts familiar to him.
>
> .

Buttons:

Boolean Text (what the action does)
Icon (only if the action is common and is familiar with everyday use, eg SAVE, DELETE etc. we use icons that are more visible and readable.We prefer Boolean Text when the action is less common (eg BUY ITEMS , START ACQUISITION, etc.).

Caption: The first thing, we replace all the labels with the captions, using them for example to give a descriptive name and for the translation (localization).

Tip: Used only for further details, it is displayed when the user is already on control. The user could use the control before reading the tip.

3 – control and freedom

> The user should have control of the information content and move freely between topics.

Use **TimeOut** or other **exit** strategies to not block processes.

Avoid long **wait** by replacing them with tickCount or elapsed time with the possibility of an exit user.

Avoid unwanted actions (Request confirmation, enter a code to enable a critical check etc.)

4 – Consistency and standards

> The user must expect that the system conventions are valid throughout the interface.

Use the **subPanels** whenever possible and the **splitbars** to maintain the main context.

Maintain a unique graphic style.

5 – Error Prevention

> Avoid placing the user in ambiguous situation, criticism, and that can lead to error.

Give the possibility to go back

Avoid ambiguous actions that may lead mistakes.

6 – Recognition rather than recall

> The instructions for use of the system must be clearly visible and easily retrievable.

Simple layout (max 3 columns) limit the density of controls and keep good spaces between one control and another.

A clear TEXT (you can not remember it easily but you recognize it better) can be better than an ICON (remember but not always recognize).

7 – Flexibility of Use

> Offer the user the possibility of differential use (depending on his experience) of the interface.

Use configuration wizard panels.

Provide basic functions (beginners) and advanced (expert)

8 – Design and minimalist aesthetic

> Give more importance to the content than aesthetics.

Avoid unnecessary images, use FLAT controls and in any case avoid accentuated DECORATIONS.

Use controls and indicators similar to the reality of use (eg if we are doing an interface for an instrument we try to imitate the tool as much as possible).

Too much information is confusing, using effects like blink or bright colors only for real attention situations (alarms etc.) otherwise they are just DISTRACTION.

9 – User help

> Help the user to recognize, diagnose and recover the error.

Sending errors in an Error Handler in which they are sorted, the action is decided on the exception (Ignore and clean, Notify only and clean, Ask for an Action etc.).

The error messages where possible are customized (on the already integrated errors we add only some information necessary for debugging or tracking, usually these errors should be eliminated during debugging).

The message describing the error must give information about the error and where it has occurred in a comprehensible way (eg ACQ.VI: The acquisition was interrupted as a result of exceeding the maximum allowed thresholds. acquisition has been interrupted, Go to the configuration menu and change the gain to 12).

Error Message: Where, What Happened, How to Resolve it (including way out).

10- Documentation

> Although the system should be usable without documentation it is preferable that it is available

The documentation is partly made together with one or more users to gain confidence and have feedback on immediate use.

It must be accessible directly from the software (Help button)

Focused on the user’s task

Structured in a set of understandable steps