IS COMPREHENSIVE AND INTUITIVELY USABLE COMMISSIONING SOFTWARE FEASIBLE?

R. Bakker, T. Birke, R. Müller[†], BESSY, Berlin, Germany

Abstract

The commissioning period is challenging both for provider and user of accelerator operation software. With increasing knowledge of the specific accelerator new functionalities and methods become important. The variety of implemented features grows until certain standard procedures are settled and the programs can be simplified again for the all day tasks. It is not too difficult to provide the requested functionalities and certain handles for a graphical usage for expert users. For the general member of the commissioning crew, however, the GUI may consist of huge windows, context dependent presentations, hierarchically structured menu trees or compact screens dominated by (modifier) key and mouse click navigation. Understanding and usability of the programs tends to decrease with the increase of available features. By means of examples like the orbit display and control program the problem is outlined and some guidelines leading out of the dilemma are given.

1 GENERAL ORIENTATION

The fulfillment of the apparent requirements for a graphical user interface (GUI) on a windowing system is the easy part of the design for an accelerator man-machine interface. Windows should not be too large. This would often hide other relevant information. They should not be too small to be readable. Behaviour and appearance should be uniform and consistent. Conflicts, inconsistant or harmful usage has to be prevented by protection mechanisms within the code. Configuration of action elements (e.g. menues) should be determined by the actual program context.

In addition control software should obey some general rules that help to shuffle around the piles of windows. E.g. standardized head lines with a color coded title help to identify the scope of the application like accelerator part or device class involved. An 'About' button popping up author, version number and program status information helps to find the competent expert to get pointed support.

2 INFORMATION FINDING

The first obvious approach to give access to program functionalities would be an open arrangement of controllers on a flat screen. As the windows get overloaded or too big grouping of entries according to common area of functionality becomes necessary. This may be done by hierarchies of sub-windows or menue trees. Often the unifying root entry is no more unique and meaningful. A not very precise search is required to find the desired function. One solution is to offer multiple ways to accept user entries simultaneously: menue, buttons on subwindows and keyboard shortcuts. The other way is to aim at very compact screens built from a few well designed and possibly multifunctional elements. A well designed example is the CERN wheel switch[1] (see Fig. 1).

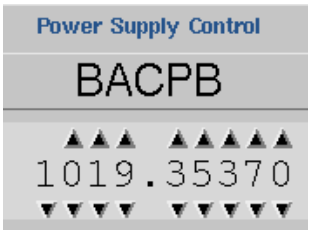

Figure 1: Wheel switch – example of a well designed widget

It can be operated in a secure way with mouse clicks, keyboard arrow keys and by entering the numeric value. The input field accepts only numerical digits. The accessible MIN/MAX values are apparent. Since it scales nicely it fits into most panel layouts.

The commercial XRT/graph[2] widget is another example. It supplies various convenient zooming, printing and data presentation options. The programming interface allows to install dedicated zooming grids, to attach pop-up labels to each data point, to (de)select points and to drag values.

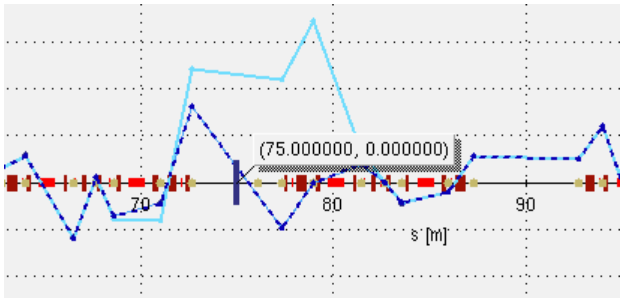

Figure 2: Clipping of a display showing the actually measured orbit data and the predicted effect of a 4-bump with angle and amplitude specified at 75.0 [m]

Fig. 2 gives an impression how it can be used to display the measured orbit and the predicted effect of a closed bump. The target position of the bump (long vertical bar) can be freely placed with arbitrary precision by zooming into the lattice and selecting the location with a mouse click. Experts like it, regular crew members are reluctant

* Funded by the Bundesministerium für Bildung, Wissenschaft Forschung und Technologie and by the Land Berlin

† Email: mueller@bii.bessy.de

to memorize the corresponding mouse button bindings.

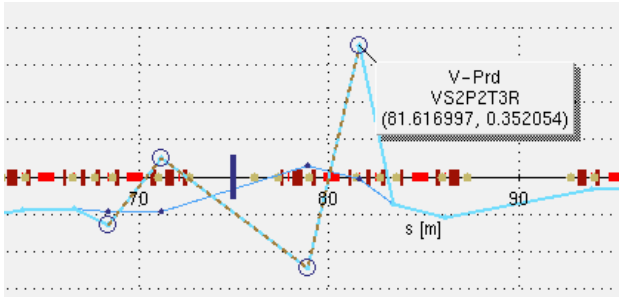

Figure 3: Clipping of a display showing the actual corrector set points and the new values the program would send to 4 power supplies to modify the orbit at 75.0 [m] as predicted in Fig. 2. The new set point of vertical corrector VS2P2T3R at 81.62 [m] will be 0.352054 [A].

Fig. 3 shows the corresponding corrector set point screen. The power supplies forming the closed 4-bump are selected by clicking into close vicinity of the desired element. As in the orbit screen device names, location and precise set points are accessible through pop-up labels.

During commissioning it frequently happens that newly released functionalities are hooked into the existing application tree as it is found to be adequate. After a time of experience or by adding other new features a better, cleaner and extendible structure becomes visible and the tree is adapted. Some users quickly accomodate to the new differing behaviour, others tend to feel lost and call for support.

3 BASIC PROGRAM UNDERSTANDING

Complexity and severity of actions caused by user request differ drastically. A mouse click may e.g. shuffle down a window and some pixels apart it may shut down a 1000 [A] power supply loosing valuable beam time. Conventional means to emphasize these differences are color coded 'dangerous' buttons, pop up confirmation windows, or blocked actions in inappropriate contexts.

However, there is a wide grey area where it seems to be impossible to give additional guidance. E.g. most users know that programs continue to run even if moved to another workspace or iconified. The majority would be annoyed if one would disallow iconification of a running correction algorithm to prevent it getting out of sight.

Where the perception of a certain action disagrees with reality the unrecognized misusage causes misleading problem descriptions and is a constant source of errors. An example is a user tuning the machine and saving several 'good' set points to the dedicated persistent memory fields[3] to be able to return to these values. For comparison with a reference he eventually reloads a snapshot file. As a specified behaviour all saved values are overwritten by values from the file. If the user is not aware of this sideeffect he notices an unexplainable modification of the data set he just worked on and reports an unreliable behaviour of the control system.

There is no self-explaining way of mapping commissioning progress into the system. E.g. during beam threading times the fluorescent screens are frequently needed and have to be freely and conveniently movable. As soon as beam based procedures prevail it is favourable to freeze the top level fluorescent screen drive buttons when reinjection is not possible. This prevents accidential beam loss. Of course on a lower layer the operator can force the screen to drive in. Without explicit explanation the user would notice a change and assume a malfunction. From his previous experience he would generally not be able to classify the new behaviour as a feature.

In another class of inappropriate understanding simple elementary assumptions about basic knowledge disagree. A programmer supplying a filter for the selection of device subsets takes the syntax of a regular expression for granted - which is unknown to a user not familiar with the underlying tools. The filter is then error-prone and of very limited use.

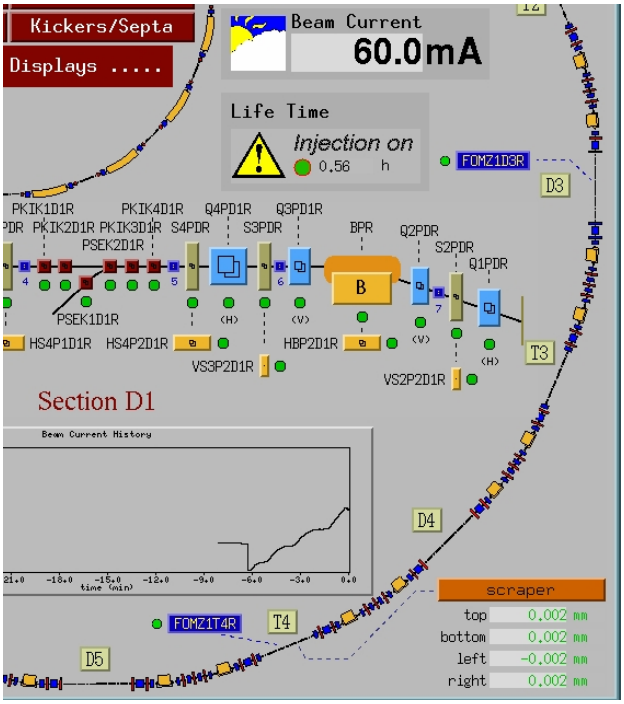

Figure 4: A complex synoptic view: action buttons, status information, related displays, performance summaries and active per-sector arrangements of the installed devices

For a power supply it is not acceptable if a new setpoint would require adjustment of internal regulator parameters until the required output current is delivered with the specified stability. This has to be already adjusted in the laboratory. Comparable tuning of the new accelerator requires beam under various conditions. Relevant parameters have to be identified and determined during commissioning. The meaning of necessary parameters require three levels of understanding. (1) basic elements: a user pushing a 'correct' button in an orbit control program has reasons to expect an action that results in an improved orbit. (2) program

intrinsics: information content of difference orbits comparing 'actual' and 'old' data is ambigous if the knowledge is missing that 'old' denotes the orbit measured when a prediction/correction has been made or a 'Set Ref.' button has been pressed and 'actual' means the constantly updated one. (3) applied methods: to be able to set the cut off factor for the eigenvectors of a SVD correction procedure to a reasonable value one has to know in principle how the SVD algorithm works. During commissioning the number of adjustable parameters grows in level (2) and especially in (3). But as soon as the correction procedures are settled level (2) and (3) will collapse to a few action buttons and the rest disappears for the regular user.

4 USER DEPENDANT ABSTRACTION LEVEL

Synoptic views are a combination of navigation tool, documentation, aid to memory and fault detection facility (see Fig. 4). For the entry level user they are a kind of tutorial access to the control system. At a running facility synoptic views dissappear. They are replaced by more efficient tools. Example is an alarm handler, which is a much faster, complete and selective fault detection tool than the human scanning of synoptic screens (see Fig. 5). But it is abstract, corresponds to a programmers view and does not meet the expectation of a running-in crew. At the beginning of commissioning a balanced compromise between pictorial and abstract tools is required.

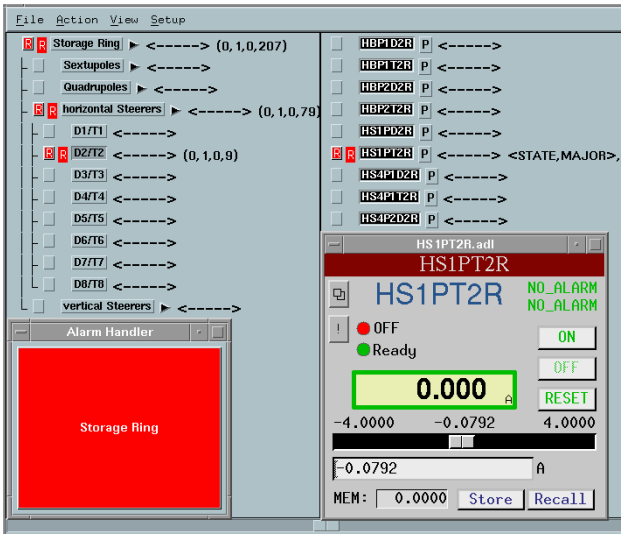

Figure 5: Alarm handler: the main button (lower left) starts blinking, clicking opens the tree leading to the faulty device(s), the 'P' button opens the device control panel suited to fix the problem.

An orbit measurement tool needs code for data handling, statistical evaluation, comprehensive display etc. A visual orbit correction tool needs most of this code too. In addition to the orbit data the corrector set points, conversion factors, drive limits etc. have to be administered. With respect to management of all data orbit modifications by closed bumps and orbit flattening procedures are closely related orbit control tasks. Furthermore the described functionalities are very similar for any accelerator, especially for the different accelerator sections transfer line, synchrotron and storage ring. With respect to code re-usage and simultaneous development it is an efficient approach for a programmer to implement all functionalities named above within one generic program. The GUIs for the different accelerator sections and the corresponding specific functionalities are the remaining individualized code fragments that have to be adapted.

Typically operators have a different view. They have to concentrate on one task: orbit measurement *or* orbit correction *or* bump modifications. Additional elements superfluous for this task are disturbing, distracting, making an efficient usage of the program more difficult. It would be a considerable improvement to provide distinct GUIs for the elementary tasks measurement, correction and bump. They should contain only elements that are relevant for the specific context. In addition each of these instances should have a user mode with all default values set and hidden and an expert variant with full control of all parameters. Preference has to be given to a separate instantiation over an additional expert screen. Experienced users find additional navigation effort caused by expert level windows annoying. With the GUI server used at BESSY[4] these modifications would not be very complicated or hard to maintain - it simply has not been thought of.

5 SUMMARY

Complex software tasks like providing accelerator commissioning programs require a certain amount of teaching how the software is meant to be used. Providing concise manuals is mostly not sufficient. One has to allow for a learning time until a precise use of the tools is possible. Functionalities of effectivity-enhanced expert modes are appreciated later. Due to shortage in man power and a transient nature commissioning software can not compete with the streamlined and consistent graphical user interfaces of commercial packages. Further more the constantly ongoing software development under time pressure is accompanied by newly introduced bugs that did not show up during tests. This adds complexity to an already difficult situation.

6 REFERENCES

[1] The wheel switch widget has been originally developed at CERN (Contact: *Franck.DiMaio@cern.ch*). Later other laboratories (ESRF, BESSY) contributed.

[2] XRT/graph is a trademark of KL Group Inc., Toronto, Ontario

[3] R. Bakker, T. Birke, B. Kuske, R. Lange, R. Müller, 'Experiences with Commissioning Software Tools at BESSY II Status', this conference (MOP31).

[4] T. Birke, R. Lange, R. Müller, Proceedings of the 1995 ICALEPCS, Chicago, 1995, p.648