My control system wish list

Digital control systems have undergone quite an evolution since their first appearance in the second half of the seventies: The hardware has become much more powerful and also the software has grown in features and performance. Yet, being active in controller troubleshooting, development of advanced controls and performance analysis on many different systems I find it always surprising that many of the features and functions needed to work effectively and effectively are still not available in most of them. Therefore I want to point here at the most needed ones with the hope to create stimulus for more user input into the vendors and eventually to see them being generally available.

Significant progress has been made - but more is needed

No doubt, DCS systems and control computers have made strong progress over time: First off, the processors have become much more powerful. The first control computer I ever used was a Honeywell 4500 / PMX system which had just 64 k memory (!). Yet, due to the clever, efficient design of the operating system (Microsoft do you hear me?) this did not prevent us from running dozens of advanced control applications plus several LP- (Linear Programming) modules in real time.

Also, operating systems and application software have evolved significantly. But a closer look shows that the main developments were made in the operator interface and in supporting functionality such as alarming, reporting and the like. The 'true' process control functionality, however, has hardly seen any significant evolution at all. Also, concerning auxiliary functions for the building and maintaining of controllers and control schemes, progress has been rather modest in most cases.

Certainly, in some systems the standard PID controller was expanded in functionality and in some cases truly enriched (- unfortunately in others rather diffused with a whole flood of options and parameters that are hardly understood by the users, let alone used). But there is more to process control then just the PID controller. In order to be able to handle all the control tasks at hand we need a whole range of different control technologies, a full 'Technology Set' - the arsenal of process control. And, as said above, we need also more functions to build and maintain more powerful but also often more complex control schemes.

Of the two, the control Technology Set is the more important and therefore should be discussed separately. Let us here concentrate on some key functions that would make our job easier and faster.

Facilities for better operation and controller performance monitoring, analysis and troubleshooting.

First of all, the information on the process needs to be further improved. Many systems still lack adequate (the emphasis is on '***adequate***') display and esp. storing mechanisms for our most important information, the operating data. I find it truly absurd that users are forced to buy and use external real time data base systems. They also have their own specific user interfaces

which violates a fundamental demand, namely, that all important functions should be handled in a fully integrated and coherent way.

The trouble with the operating data is often augmented by shortcomings in their presentation. Admitted, we see today higher screen resolution, faster updates, more parameters that can be updated on one particular display, etc. But as far as one of the most important needs of both operators and control engineers alike is concerned, namely, truly information-rich dynamic representation of the data over time, in other words the plotting and trending of the process variables, there is still a lot to be desired.

In some cases only a few variables can be trended on one screen which does not allow to see **all** key influencing variables **together** with the controller's PV, setpoint and output - which is so important for troubleshooting and tuning. In a few cases it is even very difficult to have the output trended. In some systems setting up a new trend display is quite cumbersome. In others, trends of the PV, SP and OP are shown on the controller display – really nice, but the plotting scale is always equal to the measurement range and cannot be adjusted at all. The result: Just a bunch of straight lines without any information content whatsoever. Sometimes, when we look at a trend display but need to go to another display just for a moment, that trend image is lost and the trending just starts again at the current time point - a real pain during testing or troubleshooting.

What we need are easy to configure and use functions that allow at least 6, better 9 or 12 variables to be shown, where we can easily change the update intervals and ranges which we can associate with a certain controller or display so that they automatically are available when that controller or display is called and that allow us to follow the variables without any data loss when we need to switch over to another displays for a moment.

Furthermore, we need better facilities to see (and store) all the different pieces of process information, different data types, together: For the analysis of problems and upsets it is important to follow the development of continuous process variables like temperatures and pressures **together** with events like alarms, status changes of switches, controller modes etc.

Finding the needed display is another difficulty in many cases: Some systems show huge lists of cryptic display names and only allow accessing them by their exact name. More elaborated linking mechanisms between displays (not just toggling between the current and the previous display) like an Associated Display feature (best in three or more levels) would help to speed up things considerably Also, linking display names directly and automatically at build time to the unit ID would be another helpful feature as well as a fuzzy search on display (and also tag or block) names.

Improved controller and control scheme creation and maintenance

Some systems are based on individual building blocks: Every major function like an analog signal input or output or a ratio control function or the PID is represented as one single entity. This approach gives of course very high flexibility.

Other system use a predefined structure for their controllers, so-called ‘tags’ or ‘points’ where input, control and output processing are already combined and the user only needs to specify the needed function in every block. In this case vital information from one part of the structure to another is already automatically passed on. A typical example is windup protection: When the controller output has reached e.g. the high end of its range then there is no point in allowing setpoint changes that would try to push it further up. In block-based system it is typically left up to the user to take care of this – which takes extra time and effort. This information exchange is not only needed within one single controller but of course also between all the controllers in a cascade.

Another example is the automatic stopping of the control and output processing upon detection of an invalid (“bad”) measurement and the automatic re-initiaization when the measurement becomes “good” again. Of course, also this functionality can be done by the user in block based systems as well – but again, at the expense of extra effort.

All these features should be provided in every system, regardless of the basic design philosophy. And it is truly not too difficult to design a DCS in such a way that macro-structures consisting of blocks that naturally belong together like input–control-output processing are automatically detected and all the necessary information interchange is automatically ensured by the system.

Furthermore, connections between the tags or blocks respectively are often very difficult to trace. A sound and smart “cross reference“ utility that lists and presents these connections in a clear, informative way and protects from unintended deletion of referenced tags or blocks or unintended de-activation of these connections would be very helpful as a ***standard*** feature.

Productivity and performance tools

The key motivation for buying and using an expensive DCS system is (or shall we say: should be) to deliver better control performance. Yet no system comes with a really sound, dynamic Performance Monitoring System built in that allows to check if this objective has been met or not. Besides, such a system would tell us which controls do not need any attention and where we should focus our maintenance work.

In many cases we need quantitative knowledge about the process behavior: For example, the Controllability Ratio, the ratio between the process deadtime and the dominant time constant, allows us a quick check if the PID is suited for the given process. Secondly, many methods exist that allow us to calculate sound PID tuning on the basis of the process parameters. And thirdly, for certain control techniques like feedforwards and model based control we cannot work without having quantitative information, e.g. the process parameters, at all.

Consequently, we need also tools for estimating the process parameters from the test results or sound operating data. Of course, there are tools around like GLIDE, MIDSA, TOPAS etc., but again, with respect to inter-connectivity and user interface issues it would be preferential to have this functionality integrated in the system. And, of course, tools for tuning the most widely used controller, the PID, should be a standard feature of any control system.

Finally, to develop functionalities that cannot be done with the standard building blocks and functions, a neatly integrated programming language should be available in any system. The key demand here is to make the communication with the DCS database as easy and as safe as possible and to allow every tag or block to execute such programs as specified by the user.

Conclusion

Despite the advances made there is still a lot to be desired and we just have highlighted here some key areas. Surprisingly enough, most of the lacking functionality has to do with the 'core' task – with the fundamental information and control actions. Besides, improvements in the data handling and representation we ultimately need a true engineering toolbox. Strangely enough, the 'old' Honeywell 4500 / PMX system mentioned in the beginning had already more of that above wish-list realized than many of the latest DCS systems on the market. A couple dozen of these systems are still in use and many of the users have been and are still reluctant to change because of that. In any case, with this article I hope to trigger some stimulus for more user input into the vendors which would eventually give us the features needed to better cope with the ever growing demand for better control in shorter time.

Hans H. Eder

ACT

hans.eder@act-control.com

www.act-control.com

Addendum: Some more problems, more motivation for betterment.

The situation: In most cases I get called when the customer has a problem. Typically the problem has existed since long time (in one case 10 years) yet changes in operations/safety regulations have escalated problem. Own staff has tried many things, yet without success. They need a solution – fast. This is the job and these solutions are always developed in close cooperation with the folks responsible for the system (brings up another issue in the modern world: Outsourcing of such vital functions with all the negative consequences).

Lesson 1 to be learned: Never wait so long before calling for outside help. In the meantime quite a bit of money is lost.
Lesson 2: Develop and keep DCS experience in-house to have really "full control" over your system.

I have given many training courses for several vendors, typically at their training centres. Therefore I know their system trainers personally. Also many vendors offer engineering services and many engineers have taken my training courses.

Therefore when I ran into questions or problems with the system must first phone call goes to the trainers or the engineers. Yet in almost all cases the answer is "I don't know how to do this", "I don't know why it's

not working", "never thought about this / never used this feature", "although I cannot figure out from the documentation how this is supposed to work".

The consequence: You have to figure out yourself by trial and error – at the expense of the customer, at the expense of your sleep at night etc. When trying out something and only that feature or function block has to be compiled this costs you a few minutes. However if for every little change the whole compound or library has to be recompiled this costs you hours.

Some examples:

PID controller. The question: Does it work internally normalised or in engineering units – important to know for the tuning. Documentation doesn't say, trainers don't know. Back to trial and error.

Delay/deadtime table: We estimated a deadtime of 15 seconds and parameterised the table accordingly. First test showed that this was underestimated, so we increased to 17. Result: A big bump in the control scheme. Reason (found after detective work): Upon start-up the table is initialised only up to the user specified number – not in full. The rest contains just garbage. Reducing the number of delays is fine, increasing it means disaster.

Also, why on earth are these tables in some systems normalized, in one case 0-100, in another case 0-1? The functionality needed is: Read a certain value in, hold it for the specified time and than pass it on – as is. Nothing else, as simple as that.

Lead/lag initialisation. Why on earth is the result of the very first execution a 0 and not the input value? Result: Bump (again). Thanks god, I had insisted on setting temporary clamps on the valve.

PID options: In one case there are more than 70 parameters/options (!) – heritage of a predecessor. Many are more than poorly documented. The process: A batch reactor, exothermic, danger of temperature runaway. One day, during a test, a sudden jump in the setpoint. We did not do it, console log proved that the operators did not do it either. After long search: In his desperate search for betterment and battle with all the mysterious parameters, the previous control engineer had set an option that set the SP equal to the PV every time the OP hits an OP clamp. Who needs that feature? I really would like to know!