Configurators, Languages, and Graphics

Edgar H. Bristol
Control Concepts Originated Fellow Emeritus
28 Union St. The Foxboro Co.
Foxboro, MA 02035
USA
Phone: (508) 543-8829
ebristol@mediaone.net

KEYWORDS

Control, Languages, Integration, Intent

ABSTRACT

Industry thinking for computer control standards and documentation emphasizes concepts and graphics derived from an earlier era when they were needed to trace circuit implementation A digital world can rely on the computer to compile and error check. Instead its documentation should emphasize the application intent and readability, permitting a much higher level of understanding and checking. For many years, the author has been describing a general control application language, based, to the extent possible, on such an Intent representation. The discussion is based on the language Idiom concept and traditional continuous control practice as a best available example of a practice supporting a recognizable Intent based representation. The paper illustrates the Idiom concept with a web site available continuous control documentation tool recently put together to develop wide community exposure. Beyond documentation, the larger language allows automatic integration the controls with other operational dimensions. Most obviously these include the automatic structuring of the operator interface. But the paper also extends this to show how the application information embedded in the control design can be used to support automatic failure accommodation and maintenance support.

INTRODUCTION

Industry thinking on control application standards and documentation emphasizes simple graphics and tabular data. These represent weak partial solutions, because the underlying needs are misunderstood. Each of these elements is intended to simplify control application and make the result more easily understood and utilized, its intent more apparent. The problem is that both the resulting graphics and standards emphasize control implementation rather than intent, and that implementation is based on pre-digital technical restrictions.

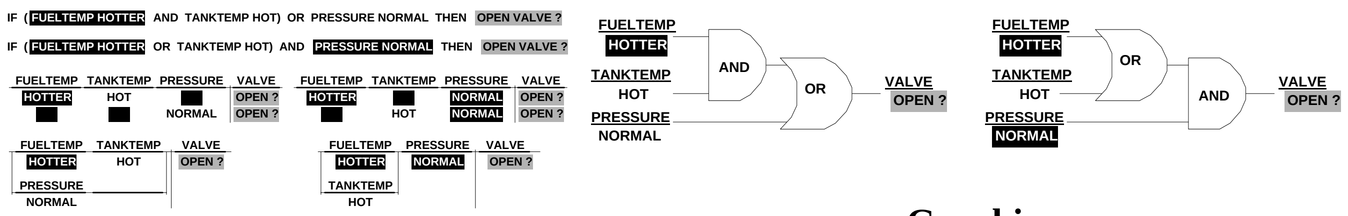
For example a complete process control instrumentation graphic is most appropriate for leading an engineer through an analog wiring of a control design. But engineers no longer need to trace wires in the control application, and implementing a control database about this model is antediluvian. The goal of the standards is to minimize the effort in training user personal and make the fundamental operation of the controls vendor independent. But a standard, based on analog control strategy is equally implementation oriented, and thus dated, focused on the wrong problem, and intent obscure.

Modern digital technology is ideally suited to automatically compiling a lower level implementation based on a higher level representation of intent, for example a machine language implementation from a FORTRAN, VB, or C specification. When the practice supports a complete, natural, well understood relation between standard application goals and their natural implementation; automatic compilation is the desirable result. Traditional continuous process control has represented such a practice, simple enough so that engineers can learn it with minimum specialized academic training. Unfortunately, the field has always dealt with the design intent in an informal manner, never developing or accepting the necessary intent formalization to make the Intent compilation possibility obvious.[1]

For many years, the author has been describing a general[2] control application language family, based, where possible, on such an Intent concept, and expressed in text and graphics.[1-7] The most obvious benefit of such a language is in clarifying user designs. Truly readable documentation would make process control application substantially less expensive and more universally understood. But when documentation and standards are based on control implementation, not only is the control design obscure, but its relations to other aspects of the application are also obscure. This loses for the industry one of the key benefits of standardization based digital technology: automatically supported integration. As a result, each separate dimension of the application effort must be separately designed and implemented. And to the extent that their intended functions are related, the design teams must puzzle out these relations in an integration effort.

The paper will develop automatic Intent based integration in terms of the most complete Intent based aspect of the above languages: the use of Idiom Control Loops to implement continuous controls. The author has put together a web site and a freely usable documentation package allowing anyone to try out the Idiom notation for control documentation.[8] The paper will briefly introduce this tool. But the paper will discuss how the explicit formalization of Intent, as part of a control language, allows other related system dimensions to be automatically integrated, or even automatically implemented. The paper will elaborate earlier discussions of automatic creation of the human interface, and further show how other, less obviously control related, application functions can be integrated. The paper will briefly discuss the potential use of these ideas in handling integration of control and maintenance.

In each case, the system dimension will have a subset of functions that are meaningfully derived from the control Intent specification, involved with any integration, and another subset of functions which can be handled locally in traditional tuning tables. The goal is a plug and play style of integration, but at a more advanced level: not between printers defined in fixed attributes and hidden drivers, but in application elements defined by user written application programs.


IF (FUELTEMP HOTTER AND TANKTEMP HOT) OR PRESSURE NORMAL THEN OPEN VALVE ?
IF (FUELTEMP HOTTER OR TANKTEMP HOT) AND PRESSURE NORMAL THEN OPEN VALVE ?
FUELTEMP TANKTEMP PRESSURE VALVE
HOTTER HOT OPEN ? NORMAL OPEN ?
FUELTEMP TANKTEMP PRESSURE VALVE
HOTTER NORMAL OPEN ? HOT NORMAL OPEN ?
FUELTEMP TANKTEMP VALVE
HOTTER HOT OPEN ?
PRESSURE NORMAL
FUELTEMP PRESSURE VALVE
HOTTER NORMAL OPEN ?
TANKTEMP HOT
FUELTEMP HOTTER
TANKTEMP HOT
PRESSURE NORMAL
AND
OR
VALVE OPEN ?
FUELTEMP HOTTER
TANKTEMP HOT
PRESSURE NORMAL
OR
AND
VALVE OPEN ?

Graphics

Graphics are often argued as the basis of clear engineering documentation. Even in traditional terms there is literature arguing that the usual belief in graphics is overstated.[9] As an example of graphics proposed to represent logical calculations[3], the figure above shows four ways of representing the same logical calculation: the IF statement, the Logic Diagram, the Truth Table and a variant form of Ladder

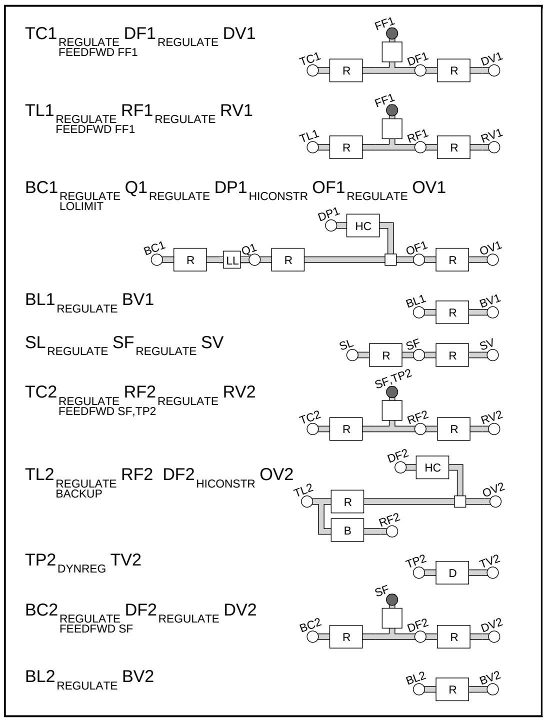
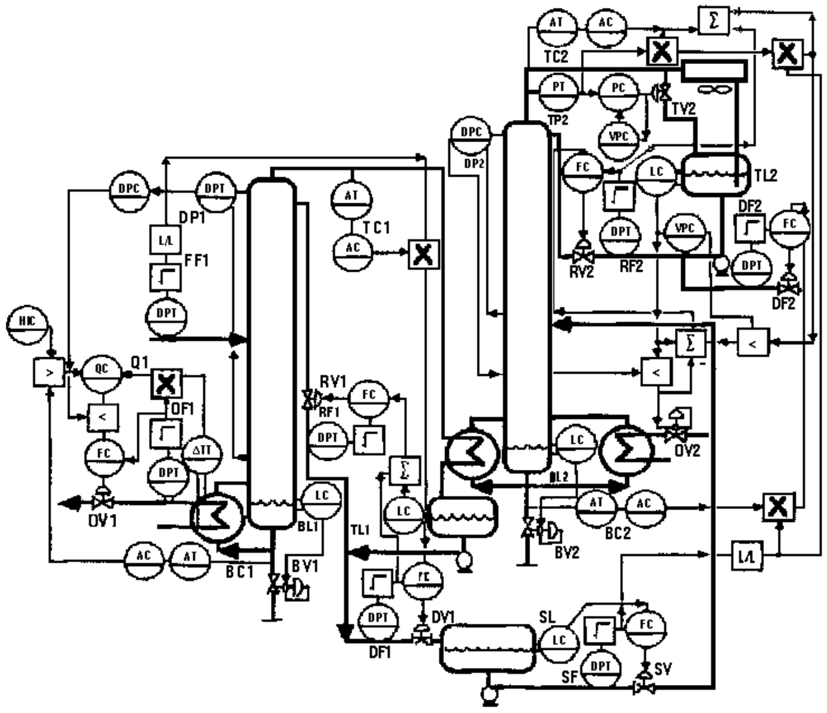
[1] Equally unfortunately, this lack has encouraged the academic community to base control teaching on a different, unnecessarily abstract and complex mathematical formalism.

[2] General purpose, allowing <u>both</u> simplicity and complete flexibility, as the user can always program any missing complexities for himself.

Diagram. In tests for recognition of the Intent of the design the IF statement took three times the time of the Truth Table and Ladder Diagram to understand, whereas the Logic Diagram took one and a half times as long. Historically Europeans have preferred the Logic Diagram. The relation is as follows: The Logic Diagram most clearly expresses the propagation of logic signals, an implemention issue; the Truth Table and Ladder Diagram both express cause and effect of application events and responses.

Thus they better express the Intent of the statement. Typical engineering diagrams are designed about the earlier implementation issues. But a control system represented in a computer should represent the application Intent not its implementation. In this day and age the computer can compile the implementation. Most conventional arguments for traditional graphic control diagrams ignore this. They rely on the traditional implementation diagrams and do not attempt the needed kind of Intent representation.

The remaining discussion will use the Idiom Loop Statement as a best example of Intent Based Modeling. To illustrate the comparison with the traditional diagram consider the two tower debutanizer


TC1 REGULATE DF1 REGULATE DV1
FEEDFWD FF1
TL1 REGULATE RF1 REGULATE RV1
FEEDFWD FF1
BC1 REGULATE Q1 REGULATE DP1 HICONSTR OF1 REGULATE OV1
LOLIMIT
BL1 REGULATE BV1
SL REGULATE SF REGULATE SV
TC2 REGULATE RF2 REGULATE RV2
FEEDFWD SF,TP2
TL2 REGULATE RF2 DF2 HICONSTR OV2
BACKUP
TP2 DYNREG TV2
BC2 REGULATE DF2 REGULATE DV2
FEEDFWD SF
BL2 REGULATE BV2


application figures below:

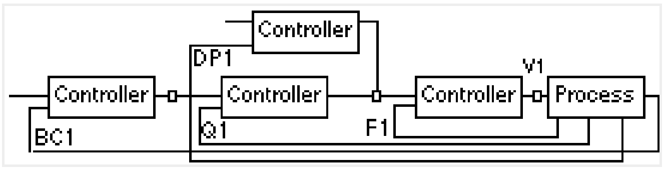
Most of us familiar with the kind of complex block diagram shown on the right can imagine spending an afternoon trying to understand it. In fact it includes only ten control loops. However, they are too intertwingled to be easily sorted out. Loop diagrams can be used to simplify such a design. The Idiom Loop Statement is a formalized loop diagram; the lefthand listing, describing the control system, summarizes the loops in a dual text/graphic form automatically related. It can be explained in five minutes in a way that can be taken away by the listener.

In each above figure, traditional graphic diagrams are compared with alternative text or graphic representations. It should be clear that graphics are not the issue. The question is what should the document convey, and how best to do it. Traditional graphics represent traditional engineering implementations no longer relevant to a digital implementation which can be compiled from any desired form. The documentation should instead represent the closest to end user Intent for which automatic compiled implementation is feasible. And it should do so in terms of whatever automatic layout and display forms make the result clearest to the widest reader community possible, **once that community has been properly trained to the need and use of such clear documentation**.

The rest of the paper will build on the Idiom Loop Statement as an example of the benefits achievable from the Intent Based Modeling. It will show how a Intent language forms, generalized beyond just the needs of the control expert, can support understanding of a wider human audience and automatically integrate related application system dimensions. The examples used are human interfacing and maintenance.

IDIOM NOTATION

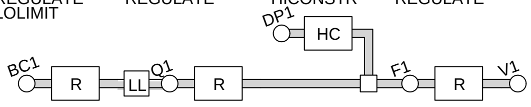
The figure shows a triple stage cascade taken from the earlier debutanizer example. The diagram corresponds to a single Process Control control loop.



[Calling this a single loop confuses people because the figure shows three nested control loops (in addition to the constraint loop). But it is justified because the basic goal of the control system is to control the Composition (BC1) by manipulating the Valve (V1). The other loops are subordinate parts of the main loop acting to improve the valve manipulation.]

Each Idiom Loop Statement in the original figure corresponds to one primary controlled variable and one final manipulated variable. The secondary cascaded controlled variables in the statement (to the right) represent successive variables along a Degree of Freedom Path through which the primary composition variable is controlled ultimately leading to the valve. As each measurement is manipulated (through its setpoint) to control the earlier stage, it is itself controlled by being subject to regulation and other limitations (e.g. Hi or Lo Limits, Hi or Lo Constraints) and refinements (e.g. Feedforwards). In this way, each measurement in the Degree of Freedom Path is associated with a Phrase, repeating a basic language statement regulatory pattern.

$BC1_{\substack{\text{REGULATE}\\ \text{LOLIMIT}}}\ Q1_{\text{REGULATE}}\ DP1_{\text{HICONSTR}}\ F1_{\text{REGULATE}}\ V1$

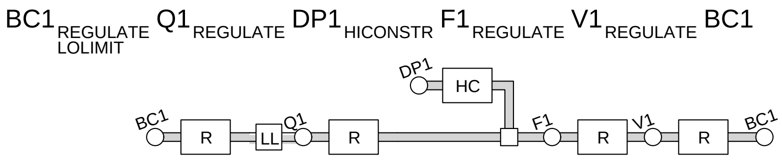


In the illustrated statement shows the simple LoLimit Idiom and the more complex HiConstraint Idiom as they are applied in the main Degree of Freedom Path to override control when necessary. The HICONSTR Idiom operator, in a sequence of REGULATE Idiom operators, is analogous to a multiplication operator in a sequence of addition operators in an algebraic statement. It has higher "precedence" and defines the overriding structure further illustrated by the graphic counterpart.

The details of the Idiom Loop Statement are developed in more detail in the references.[8] The web site provides a documentation tool with user's guide, which permits the development of listings like those shown here.[3]The reader can get a broader introduction to the notation in that guide.

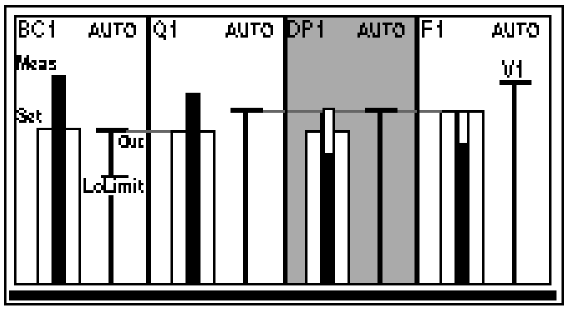
Apart from its lesser clarity the traditional block diagram (or any Object based counterpart) implies only local limits to the connections between blocks. The tool illustrates the far greater power provided by the Idiom Loop Degree of Freedom Intent model. Such a model expresses global constraints, not merely local constraints. For example, the block diagram would permit the connection of the V1 back to the setpoint of the BC1 in a Degree of Freedom loop. Nothing about the individual rules of controller connections prevents that, even though it is an absurdity from a control point of view.

[3] All of the example Idioms except the Backup Idiom in the debutanizer are supported.


BC1 REGULATE LOLIMIT Q1 REGULATE DP1 HICONSTR F1 REGULATE V1 REGULATE BC1
DP1
HC
BC1
R
LL
Q1
R
F1
R
V1
R
BC1
0: Error 20: Looped (Final Variable = Initial Variable)

The documentation tool has built in rules, which recognize such basic errors. One may see this as kind of silly. But it as essential to such a tool as a spelling checker is to a word processor. The value of the spelling checker goes far beyond checking spelling errors per se; it recognizes misstypings. For many of us the word processor became a viable substitute for the typist only once it provided this bit of automated “proof reading”. The deeper the basic Intent “understanding” built into the language system, the greater the clarity, ease of use, and integration provided. In addition to the looping error, the tool checks for any other Degree of Freedom violations, involving any statement or collections of statements. Translated into control language design these kinds of error checking are essential and normal.

IDIOM OPERATOR DISPLAY



Even more fundamental to real value from a digital application in the modern world, and more missing in the current proposed standards and systems, is the integration possible across different operational tools related to the same application. Several papers[5] have shown how different operational displays could be automatically generated from the same Idiom Loop Statement, as shown above. Integrated multi-controller displays have been an important missed opportunity for many years; this technology makes them automatic.

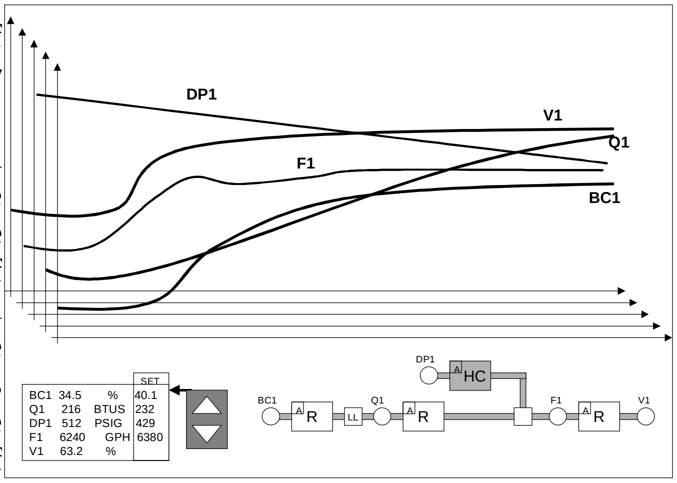
This integration recognizes that control and interfacing are different functions with different specification requirements. A PID acting as a Regulate Idiom (to regulate[4] a measurement to a setpoint) involves a setpoint, measurement, output, output feedback (with other output feedback quantities), and A/M or related operational state controls. In addition to their role in integrating related Idioms these are each required for integration with respect to operator interfacing. On the other hand, the bulk of the parameterization (e.g. tuning parameters) can be ignored for external connection and handled by local tables (separately accessible to operators in simple form, for tuning or maintenance function).

The integration related parameters, coupled with the Intent structure available from the Idiom Loop notation, are easily handled in a general manner to automatically structure human interface graphics in a way that integrates related Regulate and Constraint Idioms. The resulting display can show the operator not only the controller Setpoint/Measurement values, but also the affect of operator actions, and of related controls and constraint overrides on each other. In any such coupling of functions, each function will have the external Intent model, which can be standard and still so elegantly simple that it is useful. It will also have a more complex set of internal implementation parameters, which need not generally participate in the integrated display.

The figure shows a separate faceplate for each controller. But it also shows the activity of those controllers, shading the constraint controller when it is inactive [or shading the main controller when the

[4] The PID implements a dozen or more other usages, including the HiConstraint above, part of the reason that traditional control diagrams are difficult to understand!

constraint becomes active (take over)]. It shows how the controller outputs are passed to the setpoints of neighboring controllers also taking into account any overrides.



			SET
BC1	34.5	%	40.1
Q1	216	BTUS	232
DP1	512	PSIG	429
F1	6240	GPH	6380
V1	63.2	%	

The same operational information can be provided under a different display policy as illustrated by the trend record based display shown adjacently. In the example, the trend record displays the history of each of the variables; a separate text joystick area allows the operational modification of those variables or their setpoints. The Idiom diagram is used to show the regulation/constraint override activity and allow access to the A/M state of each of the controllers.

TIEING THE MAINTENANCE SCHEDULE TO THE APPLICATION

The field has begun to consider the possibility of integrating maintenance information into the so-called "Smart Sensors". The possibility of integrating internal diagnostics into the sensor and actuator database to make the sensor data self-correcting has also been suggested. The recent interest in Predictive Maintenance represents another attempt at allowing maintenance data to be more live. But incorporating live maintenance data into the operations has the danger of further compounding the data overload of operating people.

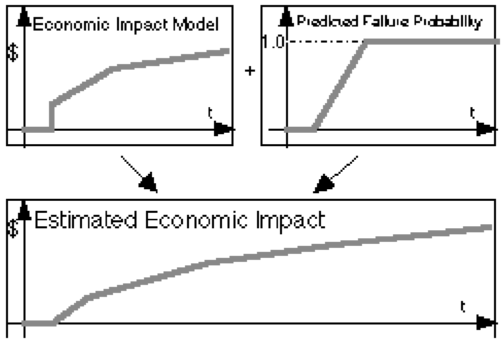
Ideally maintenance data will be used to infer actual and potential failures automatically. Predictions can be made based on specific measures of the sensor or actuator device health, or general measures of the behavior of the process (e.g. vibration). They can be made on the basis of measured health of supporting electronics such as the power supplies. There is a catch 22 here because there is little incentive to provide effective, standardized diagnostics if the application systems provide no guidance as to the effective use of them and little incentive to provide good application tools if there are no complete, standardized diagnostics to drive them.

But let us assume that sensors and actuators could be provided with meaningful self-diagnosis including predictive diagnosis. Effective use requires display aids to both the operators and the maintenance personnel. The operators need indications in their displays of the criticality of current and anticipated failures and their impact on the actual application. The maintenance people need guidance in scheduling their work orders to have the most timely benefit to the operations. Earlier papers discussed a three level smart sensor protocol.[6]

This protocol proposes three levels: raw Device Diagnostic Data, user defined Application Role codes, and generic Failure Status automatically translated from the first two levels. This generic status allows different devices, with different detailed failure behavior to be treated in terms of a more general standardized point of view. For example, a thermocouple may fail by open circuiting and a pressure tap can clog. But the application affect of these can each be translated to a single zeroed value, expressed as a single status indication. In each case the application effect is total loss of any sensor indication of process activity. Other generic status codes might indicate a less severe loss more suitable to a more constructive accommodation.

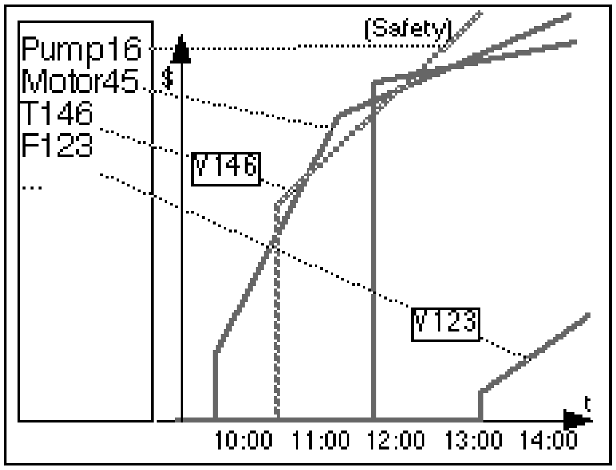
For example, a secondary control measurement, subject to a ten percent error offset can continue to function in its control role because the primary controller will be able to make up for the effects of the error on the primary variable. This kind of reasoning can be built into the controls and alarms, notifying the operator of crucial failure situations and accommodating appropriate failures, as long as there are suitably standardized generic failure codes and application roles. In the three level protocol, the application role is manually configured into the system. But the Idiom notation allows that application data to be generated automatically from the Loop Statements.

For maintenance scheduling guidance, failure data needs to be augmented by indications of its impact on the process. The Idiom Loop Statement indicates how failures will affect control but it does not include data on which process variables are most important. We can accommodate the last problem by including a (in this case graphic) representation of the cost of loss of control of each process measurement as a function of time after the immediate failure to control. The model includes a time from initial failure to first economic impact and then any further incremental continuing effect in two stages.



Ideally, predictive failure indications would take the form of a (here graphically represented) probability of failure shown over some interval of time, from the initial time in which the failure might occur to the final time when it would be expected to have occurred. The user configured cost model would be combined with the prediction probabilities to display the probabilistic cost figure.

The Idiom Loop Statement augments this data by the process relationship between the different actuators and sensors. Thus a failed sensor may include supporting backup strategies for inferring the missing process value and the associated accuracy. If the accuracy is adequate then the residual failure cost relates more to how long the data back up can be expected to continue. On the other hand, a failed valve or secondary measurement (unless supported by an alternative valve) will begin to affect some different more economically important measurement.



The Idiom Loop Statements allow the inferred connection between the failed cause variable and the economically affected measurement. An automatic system can combine the language and configuration data to generate a scheduling diagram showing the anticipated direct and indirect costs of all current and anticipated failures.

For both operations and scheduling, the goal is to allow general policies and general failure behavior specifications to be used automatically to translate the control Intent to the appropriate displays and functioning.

CONCLUSIONS

For many years older industry leaders, familiar with the design of analog products had difficulty relating to the modern applications and digital systems requirements. More recently we have made great progress in translating applications to the implemented digital systems. Now vendors need to codify our application understanding and practice and learn to translate it into an automatic compilation product technology. The pendulum has swung from an emphasis on specialized vendor expertise. Now emphasis is on raw applications. It needs to swing back to a quality vendor expertise, which better automates application Intent translation to the implied implementation.

The paper has reviewed earlier Intent based language work. It has introduced a documentation tool designed to further illustrate the power of the earlier work and make it accessible to users who might want to consider the desirability of improving their application documentation. It then extends the discussion to address other larger benefits of such systems, discussing previously introduced work on integration with the operator interface. Finally it shows how this higher level kind of control configuration can be used to integrate other aspects of the operation, going beyond the areas normally considered in the design of the control systems, in this case addressing maintenance.

As long as the industry continues to think of control configuration in terms of the implementation strategies which preceded the digital era, or indeed, in terms of implementation rather than the user's intent, we will fail to achieve the benefits of a computer age. In this we are not alone. The bulk of current industry use of computers is suspended in a limbo of implementing in a digital world the identical methods found appropriate to the old pre-digital world and unlikely, in the end, to maximize benefit from the new one. Of special irony is that the proposed Intent oriented approaches are much more in tune with traditional straightforward control practices than the practices that are often developed and taught academically in lieu of an adequate application formalism.

REFERENCES

The author papers below are available on the web site: http://homepage.mac.com/ebristol, along with related demos.

[1] E.H. Bristol, "A Language for Integrated Process Control Application", Retirement Symposium in Honor of Prof. Ted. Williams, Purdue University, West Lafayette, IN, Dec. 5 - 6, `94.

[2] E.H. Bristol, "Not a Batch Language; A Control Language", World Batch Forum, San Francisco, May `95; also ISA Transactions, Dec. `95.

[3] E.H. Bristol, "Redesigned State Logic for an Easier to Use Control Language", World Batch Forum, Toronto, May 13-15, `96.

[4] E.H. Bristol, "Information Models for a Software Future We Never Know", ISA97, Anaheim CA, Oct. 5-9, '97.

[5] E.H. Bristol, "Deriving the Human Interface from the Automatic Controls", Automatic Control Conference, Philadelphia, June 24-26, `98.

[6] E.H. Bristol, "Intent-Based Process Control Configuration Models", ISA99, Philadelphia, Oct '99.

[7] E.H. Bristol, "A Field Device Language in a Process Control Language Family", ISA2000, New Orleans, Aug. '00.

[8] E.H. Bristol, "Idiom Based Documentation of Continuous (Feedback) Process Control", distributed on the above web site, Most Recent Version: Apr. 18, 2001.

[9] M. Petre, "Why Looking Isn't Always Seeing: Readership Skills and Graphical Programming", Communications of the ACM, June 1995, pp. 33-44.