Integration of Controlled Industrial Systems with Intent Based Information Models and Language

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

Abstract

Industrial controls relate to many other dimensions of the programmed automation: documentation, human interfacing, alarms, scheduling, maintenance, and management access. Unfortunately the design (and now standardization) of each of these elements proceeds independently, integrated only in the eyes of implementing engineers, if then. Such a lack of integration loses the special benefits of digital control. This paper builds on general purpose Intent-Based language work of the author.

The work has been aimed at ease of use, clear control documentation, and integration of the different kinds of control programming. But the paper will show how the Intent-Based design extends across these other automation dimensions. The key Idiom concept, as an example of formal programmed Intent in the language, facilitates the automatic generation of associated functions in these other plant automation dimensions as well, for example in the human interfacing and maintenance.

The Intent-Based concept could be applied to many aspects of programming, particularly when restricted to the programming of a well defined practice. The paper briefly suggests where the necessary solutions can be found. It argues that the dreams of Artificial Intelligence, realistically based on practical engineering rather than magic, lie here.

Keywords: Process Control, Control Idioms, Intent-Based Modeling, Automation.

Introduction

Classically an industrial control system includes many programmed elements: controls, documentation, human interfacing, alarms, scheduling, maintenance, and higher level management controls. In practice, these elements are separately programmed by hand even under the increasingly broad and draconian international standards being developed. The problem is that these independent programs have no integration except in the eyes of the implementing engineers. When these engineers are not part of an integrated team or do not understand each other, the results of their efforts are unrelated.

Thus, so much of the intended benefits of computer based automation are lost to human based implementation. In the end the computers can only be helpful to the extent that we are able to provide computers with some representation of the control intent. If this is done, the other above aspects of the implemented computer system can either be automatically generated or correlated.

For a number of years, the author has been working on a proposed Intent-Based control language[1-10], which integrated previously disjoint aspects of process control: continuous control, logic, and sequencing. Of these elements, the classic continuous feedback control has a particular advantage in the thorough development of practices that related widely varying human intents with standardized implementation.

These practices relate the many different forms of regulatory and constraint control to the necessary degrees of freedom paths involved in their integrated implementation. This makes them an especially good example of how the proposed Intent notation can automatically be compiled into the associated control structure and validated.

More recent discussion has used the same notation to suggest how human interfaces could also be generated[8]. The present paper will develop these ideas in a further maintenance example. The same notation can be combined with current predictive maintenance thinking to generate appropriate dynamic maintenance schedule displays based on predicted failure conditions and the different control flows necessary to support the economic and safety objectives of the control design.

On a higher plane, different aspects of the operation of an industrial plant involve quite different levels of Intent, whose relation is not understood, by the different levels of plant engineering and management. Yet the relation exists because, at base, the plant is subject to a single set of physical and economic laws. Starting with the above example, the paper will discuss how quite different kinds of Intent can be sep-

arately defined in a way that allows their collective integration.

Intent-Based Modeling

As defined here Intent-Based Information Modeling[10] must be based on some aspect of system design work supporting a recognized[1], realistic, complete engineering practice.[2] The Intent-Based Modeling must then be structured in a formal graphic or algebraic notation, where named Intents are represented and implemented, within that standard practice, in a systematic way. Ideally, the notation will permit a simple visual and conceptual pattern, fundamentally related to the practice.[3]

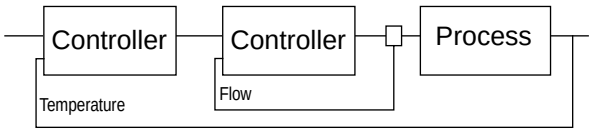
In a language this pattern becomes the basis for automatic error checking and proof-reading of the application program. But within the goals of this paper, the underlying application structure allows other aspects of the application, outside the language implemented system, to be coordinated automatically, not only on a function by function basis,[4] but globally, as multiple functional elements have collective effects.

Idiomatic Control

Traditional Fluid Process Control focused on continuous feedback control of large processes. Because the controls were implemented by connecting different standardized control functions, often designed as a free service by the vendor, a systematic practice evolved where most any desired control objective could be achieved directly, quickly, and cheaply.

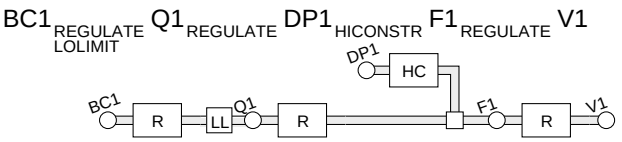
The practice was expressed not in terms of Intents, but in terms of its implementation, as expressed in the traditional analog Block Diagram. What is rarely openly appreciated is that such diagrams can become undesirably complex, like any similar (TV or Computer) circuit diagram. The Idiom[5] concept[2-10] was coined to clarify the situation; and a corresponding notation and graphic diagram[2] developed to clarify complete designs.

The key concept in continuous process control is the Degree of Freedom: the notion that every controlled measurement requires a corresponding actuator (usually a valve) to control it. In process control the concept is often extended in control cascades where, for example, a measured temperature is controlled by manipulating a flow. The flow is conceptually an actuator, but actually another measurement, which must then in turn be controlled by an actual valve/actuator.



There can be more than two Controllers in cascade, making up what will be called a Degree of Freedom Path. In Process Control the Controllers are usually P, PI, or PID Controllers. The Path starts with the initial controlled (Primary) variable (the Temperature above) and going on to the Secondary variable (the Flow) and the Tertiary variable, etc.

Notationally, the control structure is called an Idiom Loop Statement, and developed in both text and (automatically generated) graphic form:



Textually, variables making up the Path are listed in order in the Statement (**BC1** (Bottoms Composition), **Q1** (Heat), **F1** (Flow), **V1**, in this case making up a three stage cascade with the final valve **V1**).[6]

The subscripted, tabularly-listed words following the variables (or the box symbols following the labeled nodes) designate the Idioms. The Regulate Idiom is the Basic Control Idiom corresponding to the Controller in the Block Diagram. It indicates the regulation of the measured variable to a specified target or Set point, which is an implicit Attribute of the variable, as developed in the notation and language.

In the initial case the Regulate Idiom is supported by a LoLimit Idiom which limits the output action of the controller on the low side. Support Idioms can implement a number of functions like Feed Forward which improve or qualify the control without altering the Idiom's basic function and Degree of Freedom allocation.

The illustrated Idiom Loop Statement also includes a **DP1$_{\text{HICONSTR}}$** Phrase. The **DP1** (Differential Pressure) variable is not in the main Degree of Freedom Path; the HiConstraint Idiom is not a Basic Idiom but an Override (or Constraint) Idiom. This is typically implemented with a PI controller and what is called a Selector. It function is to impose a High Constraint on the critical **DP1** variable, overriding or switching the main Degree of Freedom control path if necessary to achieve the goal. As with Set points the associated Hi Limit is an implicit Attribute of the **DP1** variable.

---

[1] So that a community of users will be able to support the associated language tools.

[2] Academic design engineering optimization typically attempts such a practice but generally leaves out too much of the real practice.

[3] Like the alternation of operand and operator in the normal algebraic computer language assignment statement.

[4] As is possible with Objects or any other subroutine-like elements with lasting database representations.

[5] The reason for the term Control Idiom is that the different Intents in a control design are implemented by sets of implementing functions whose collective behavior would often not be understood by a user unless he were widely familiar with the practice, in the same sense that a language idiom is not necessarily clear unless the listener is familiar with the language practice.

[6] The **DP1** variable is explained shortly.

More sophisticated Idioms[2, 13] define various kinds of coordination and operational or safety constraints. Idioms can manage multivariable Degree of Freedom Paths. These considerations are beyond this article but can be seen in the Web Site discussion of tools demonstrating the use of the Statement in control documentation, as referenced[2].

Note that the control variables participating in the control Degree of Freedom Paths play a much more fundamental role in the system than variables supporting simple calculations (for example feed forward computed variables or measurement compensation variables). The incorporation of the implicit Set point is only one of the differences. In the cascaded control computation, differences between measurement and Set point support automatic recognition and accommodation of lost control under manual intervention, valve limiting, or constraint override.

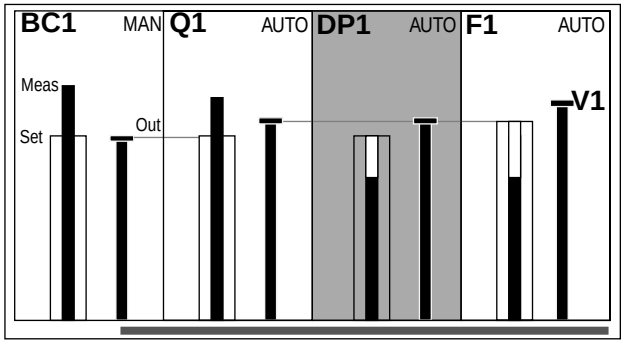
The Degree of Freedom and Variable-Subscripted Idiom structures provide the kind of pattern earlier called for in the discussion of Intent-Based Modeling. They contribute to the far greater readability of the Idiom Loop Statement, and simplify error checking in programmed control documents. All of this comes without loss of flexibility in the possible control designs, and serves to preserve the power of traditional control methods.

The referenced documentation tool[2] checks for improperly looped Degree of Freedom Paths or variables occurring more than once in Paths. Despite the lack of commercialization, people who have taken the time to use the Idiom tools find them far easier to use than anything else available. In this respect, the Intent-Based notation, with its pattern, accomplishes, in more successful form, the presumed benefits of graphic documentation[11].

Intent Integrated Human Display

Different users may expect different responsibilities out of their plant operators. In the Idiomatic operation this would be accounted for by applying different Display Policies[8]. For the immediate purposes, the discussion assumes operator discretion to operate at any level in the controller cascading, by putting appropriate controllers in Manual. With this latitude he is responsible for ensuring the appropriate operation of the control loops but may also be enforcing control objectives or constraints not covered by the automation.

The operating display (with supporting manual controls) below shows the associated Auto/Manual state of each controller. The horizontal dark gray bar at the bottom of the display shows the range of controllers, from the valve back, that are in control. The display further darkens the override controller faceplate when it is not overriding, darkening the main controller faceplates when they are overridden. Gray horizontal line connect controller outputs to the Set points of following controllers, passing through faceplates of inactive controllers. In this way the operator is assisted in monitoring the key collective control effects.

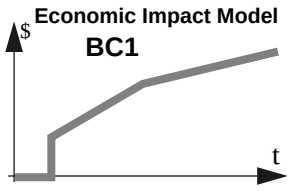


The traditional individual controller faceplates could automatically display the corresponding controller state. But it takes the integrated treatment of the Idiom Loop Statement, with all of its implied distinctions, to provide the automatic expression of the interacting behavior of connections and overrides.

Intent Integrated Maintenance

Normally Maintenance is driven by work orders from the operational staff as equipment fails or begins to show weakness. But recently efforts to use various sound spectra, vibration, and other prevailing sensed data to provide background checks of the equipment have coalesced into what is called Predictive Maintenance. This still has a very ad hoc character compared to the Idiom practice described earlier.

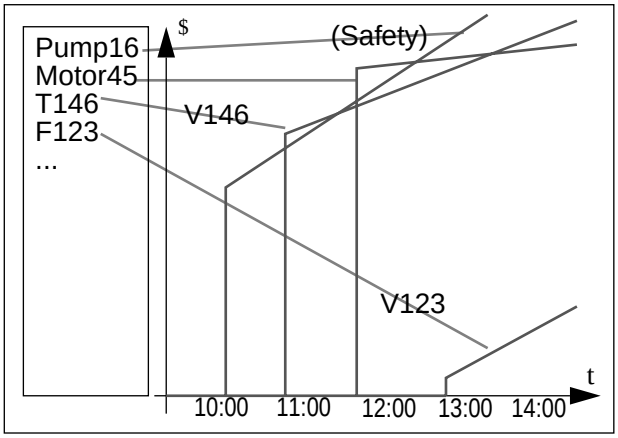
Ideally such a technology, supported by equipment vendors prediction data would allow the automatic generation of failure likelihood indications or statistics, usable to schedule maintenance efforts. Suppose that such results were generally available, as time predictions of failure or perhaps changes in the presumed Poisson exponents of expected failures. (Predictive Maintenance has yet to evolve such an standard practice.) But these would be failure indications of sensors or actuators, not necessarily of the actual economically important process variables.



However, the Idiomatic Loop Statements defining the control system would allow prediction of loss of control of important variables inferred from predictions of any associated sensor or actuator failure. The system maintenance system needs one further kind of information: a time projection of the economics of a failure once it has occurred, perhaps coded with indication of safety hazard or other basic effect classification.

The Prediction, causal (Degree of Freedom), and Economic (and safety) data can then be combined for

all of the devices in potential failure to give a maintenance projection diagram which gives operators and maintenance personnel an overview of the economic (and safety) impact of anticipated failures and allows them to dynamically adjust a schedule or otherwise react. In the figure below each translated economic projection is labeled both with the dominant affected process variable and the actuator variable (Pump/Motor/Valve) actually failing.

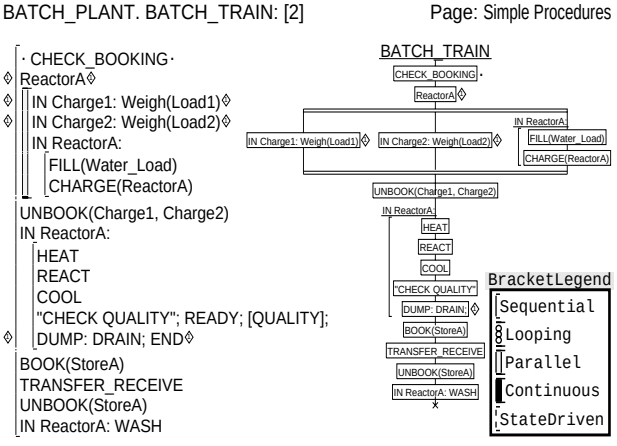


This kind of Integration requires a rigorous standardization of the internal representations of:

- Essential Structural Data.
- Essential Parameters (e.g. standard controller tuning parameters)
- Cleanly distinguished optional alternative structures, with their distinct essential parameterization.
- Vendor or Application Specific Parameters, for which the system must provide a general display capability without the deep participation in the Intent Based computation.

Higher Level Intent Integration

The full language includes Intent-related strategies for representing all of the computational functions necessary for implementing process control, for example Truth Tables based on naturally named States rather than Boolean logic for logical control, and a number of structures for representing different kinds of sequencing including the procedural sequencing structure illustrated in the figure below.



In this case the illustrating (Sequential Function) diagram is assumed to be generated automatically from the text equivalent, where the text also includes graphic brackets to clarify the different kinds of sequenced operation. None of these structures support the specialized patterning of the Idiom Loop Statement because the corresponding practice has not developed; in these cases a more general purpose (but less Intent specific) practice prevails.

The Idiom concepts are most similar to the Patterns of Object Oriented Programming. But with Idioms, the pattern supports open ended control designs with any number of control Phrases in each Statement and any number of possible interrelated Statements, whereas Object Patterns typically relate to a single larger abstract computer technology function such as stack operation

There is an opportunity to cast even general purpose computation in a more Intent-Based form; the Named State based Truth Tables would be applicable in such general usages. But the Intent-Based Integration of higher levels of industrial operation will have even greater benefits. Existing efforts to address these levels are extremely cumbersome and inflexible. Improvement requires a greater effort to codify a systematic practice, and identify usable patterns to clarify such a practice.

The Idiom Degree of Freedom pattern is based on a specialized version of a causality pattern. In looking for appropriate notations for higher level functions one needs to look for similar underlying technical constraints that suggest useful patterns for structuring and disciplining the result.

Examples of candidate Intent-Based Modeling constraint structures include:

Causality (as with the Idioms).

Closely related to causality is: Prerequisites, the notion that certain (generally sequenced) operations should always be preceded by specific set up operations, or really be associated with initially established process states. The German NAMUR industrial group has defined a “Grundoperation” concept that captures this strategy. I would like to see this term be more formally developed as an international word like “leitmotif”!

Sequencing, as with the Sequential Function Chart, and in simpler or other more specialized forms.

Inherent ratios of component elements like the stoichiometric ratios of chemical processing, or Bills of Materials in discrete products manufacturing.

Thus our notation design problem requires a matching of the application activity to be expressed, the critical issues making it up, and potential patterns which can structure the notation. Once chosen the Intent-Based Modeling elements and the pattern need to

be converted to a visual documenting form. The usual focus is on ease of programming. But more important is the ease of reading the resulting document. Appropriate programming tools can always be matched to the intended readable documenting from.

The traditional general purpose language design has always emphasized generalizing the computational mechanisms as much as possible. Because past failures of industrial computation have been associated with insufficient flexibility, the Intent-Based elements should be as flexible as possible. The existing block diagram practice has been worsened by the conflict between the desire to simplify usage by standardize blocks and the resulting inflexibility which drives the vendors toward more and more complex blocks to solve the range of actual application needs.

At the same time, distinct Intents should have distinct structures even when implemented identically. In the above Idioms the Regulate and HiConstraint were each implemented by variants of the PID controller; distinguished by the distinct Idiom names. The PID controller may have dozens of distinct Intent usages, calling for distinct Idioms and Idiom names!

An Intent-Based strategy is inherently hierarchical, with yesterday's Intent structures becoming tomorrow's implementation tools. Nevertheless, the concept of each level of Intent-Based Modeling having its own structure and pattern (in which each "Statement" is short and concise, and represented in a visually meaningful and complete way) can be applied at every level in the hierarchy.

Conclusions

Industrial computer control is stagnated in an explosion of disjoint tools and standardization efforts. As formed these expressions are incompatible with the kind of true integration needed to fully develop computer control. Existing computerization focuses on low level traditional implementation of controls or narrow (e.g. HTML/XML driven) commercial communication.

Continually higher level Intent-Based design can accomplish a much greater level of integration. It can achieve some of the original Artificial Intelligence dream, but in a way compatible with orderly engineering practice rather than with appeals to magic.

References

Most of the papers referenced below as well, as several relevant live demos, are available on the web site: http://homepage.mac.com/ebristol .

[1] E. H. Bristol, "SuperVariable Process Data Definition", ISA SP50.4 Working Paper, Oct. 24, '90

[2] E. H. Bristol, "Idiom Based Documentation of Continuous (Feedback) Process Control", Continually updated on the web site.

[3] E. H. Bristol, "Simple Single Loop Idiom and Simulation Demo", Continually updated on the web site.

[4] 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.

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

[6] E. H. Bristol, "Redesigned State Logic for an Easier to Use Control Language", World Batch Forum, Toronto, May 13-15, '96; also ISA Transactions, Dec. '96.

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

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

[9] E. H. Bristol, "A Field Device Language in a Process Control Language Family", ISA2000, New Orleans, Oct. 5-9, '00.

[10] E. H. Bristol, "Intent-Based Process Control Configur-ation Models", ISA99, Philadelphia, Oct. 5-9, '99.

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

[12] SP95, Enterprise/Control Integration Committee, Ongoing ISA Standards Committee, any of the Working Papers.

[13] F. G. Shinskey, Process Control Systems, McGraw Hill Book Co., 1988.