Information Models for a Software Future We Never Know

E. H. Bristol
The Foxboro Co.
Dept. 0318, Bldg. C41–1H
Foxboro MA, 02035
Tel. (508) 549-2019
Fax: (508) 549-4379
email: ebristol@foxboro.com

KEYWORDS

Future Technology, Information Models, Idioms, Display Technology, Nonlinearity

ABSTRACT

Despite normal thinking to the contrary, technical change almost always proceeds slowly, appearing rapid only when a wider world becomes aware of the possibilities. The management of future process control technology depends on recognizing the underlying constants:

- Fundamental directions and rate of change of computer technology,
- Long term developing Control and Computational ideas,
- The underlying goals and intents of process control and their interrelationships,
- The time limits on human ability to make real sense of change.

This paper addresses the need to convert these constants into application Information Models, based on deep underlying intent and meaning (semantics) which can guide our designs, standards, and more short term efforts. Depending on computational expertise foreign to most process control practitioners, these models are nonetheless required for progress, and available from no outside source. A first example, models formalized control intent to bridge configured controls to automatically generated, operator interfaces. Other Information Models address other application opportunities, some new and obvious, like Virtual Reality and Large Screen Displays, some old and deep, like General Nonlinear Multivariable Control and "How to make sense out of alarms".

INTRODUCTION

Current controls thinking is naturally overwhelmed by the developments in new computer technology; it may be somewhat forgiven if it sometimes loses its way to all the noise. Among the developers of new computing directions, those addressing multimedia PC innovations have certainly been farsighted in identifying the necessary infra-technology and support standards. If process control is to be effective in profiting from these advances, it needs its own better vision of where the world is going. It must understand that, contrary to the usual view, the world moves slowly, and that the apparent speed of progress is an accident of the interaction of simple silicon advances with more conventional technologies that have been developing for a long time. After all, Moore's law is very simple. If you know that computing power doubles every year or two, what do you do? You plan on it; You assume it! As an example of the older issues now being faced, Virtual Reality is the projection of Reality. If you want to understand it you have to understand Reality itself, a difficult thing but hardly new.

So our problem falls back on understanding process control the way we should have all along. Our field's response to new technologies has been to look at a few obvious first applications and then standardize on the characteristics that happened to show up along the way. The result takes our first unimaginative attempts to embrace the new technology and freezes any real innovation. This approach will be inadequate to

track the kind of technology that we face. In the past I have compared this to standardizing on the Latin alphabet and then expecting the problem of language translation to go away. We need to begin to develop Information Models of the process control activity that transcend the short term technology solutions and thus better lend themselves to direct re-mapping onto any new technical opportunity.

Until now, this has been too foreign for our average process control engineer. My “recent” (for the past 15-30 years) work has concentrated on defining a general process computer control language. One of its more interesting developments is an expression of regulatory control more aimed at expressing control intent. Finding well documented application examples (which explained their control intent) for this development has been frustrating. You would ask the application designer what he was trying to do with his control design and he would say: “Well, we are taking a PID and a Selector...”; Grrr! Only if our standards and models of Process Control address the underlying application and commercial goals of our applications and the reasons for them, independent of implementation, will we be able to create architectures, standards, and designs which are “unchanged by change”. Then each new technology can be adjusted quickly to match this well established intent. In fact, if the programming has been executed in terms of intent, its “re-compilation” to each new technology automatically tracks changes in implementation.

Among other things, a good control language, reflecting the purpose of the control design, ought to generate the operator interface as an automatic consequence of the control intent and configuration. And it can, without the miracle of control engineers becoming operator sensitive. After all, when a single loop controller is sold, the faceplate comes for free. In computer terms, this becomes possible if the basic control configurations are semantically complete, defining the purpose as well as the implementation. The industry needs to understand the bridge between the semantics (the intent and meaning of the application) and the narrow, current implementation possibilities.

There are a number of long term problems and opportunities facing us. Some are obvious: like Virtual Reality and Large Screens. Some are deep: like General Nonlinear Control, or “How to make sense out of alarms”. The paper will take these problems and discuss their solution in a future world, and what it would take to solve them: in terms of software tools and standards. Hardware is not the fundamental issue. Ninety percent of today's problems, properly understood, could have been solved with the last decades computers.

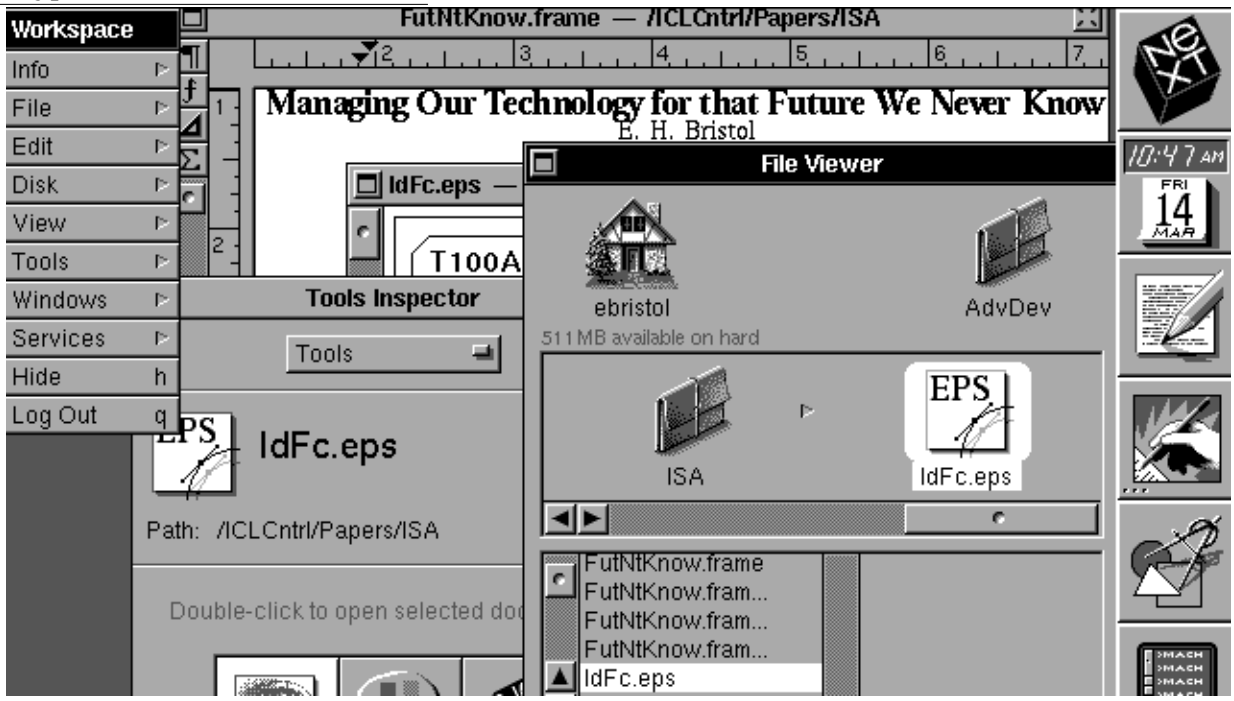
NOW IS THE PAST'S FUTURE

To illustrate how one can plan for a future based on this “slowly moving technology”, examine how a now common place technology, the function of a Macintosh®/Windows®, style GUI function might have been implemented as a pure Text User Interface (TUI) on text only screens. The exercise is not entirely empty: Pure keyboard computer interfacing is known to be faster (easier?) than the mixed mouse/keyboard action that we are used to; the contribution of normal mouse menu operation is the operating reminders that they provide.

As a reminder of the requirement, Figure 1 shows a set of application windows on a NeXT® (a “SuperMac”) computer screen (jammed into the 1/4 the space of the full screen). The figure shows the elements that would have to be matched within our TUI:

- The “Dock” on the right of the screen containing icons to call up specified applications.
- Main Windows for three open documents each supported by different open applications:
  1) The Main Workspace File Viewer, indicated as the current active application.
  2) The Document for this paper (“Managing Our Technology for that Future We Never Know”, file named FutNtKnow.frame) running under the FrameMaker® Application.
  3) A PostScript file (IdFc.eps) running under a Previewer application (and current file selection).
- The active application always shows a Working Menu in the upper left hand corner. This menu shows submenu selections (the triangles), and control key substitutions (e.g. ‘q’ for Log Out).

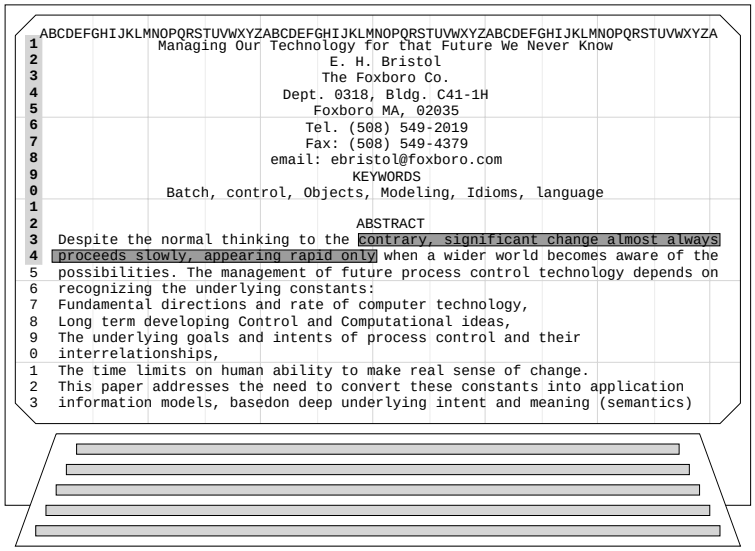
Fig. 1. Typical GUI Contents



- The active application may have additional supporting inspector windows, like the Tools Inspector shown, which allows the selection of software Tools to be applied to the current file selection (IdFc.eps).
- Selection of any open document or application, of any menu or inspector item, and of any editable text string.

Figure 2 shows the Word Processor application as active application in our TUI. As an alternative to text

Fig. 2. Text GUI with Word Processor Application Active



selection by mouse, each row and column is marked by the alphabetically ordered column marks at the top of the page, and the numbers at the left of the page. A grid allows the eye to locate the grid positions within the screen. The rows are indicated as 1 – 9, 10 – 19, 20 – 23; the columns by A – Z, AA – AZ, BA – BZ, CA. Thus the shaded text, “contrary, significant change almost always proceeds slowly, appearing rapid only”, is selected as: ***S*** 3L – 4M (See below.), relying on the eye to be faster than mouse selection.

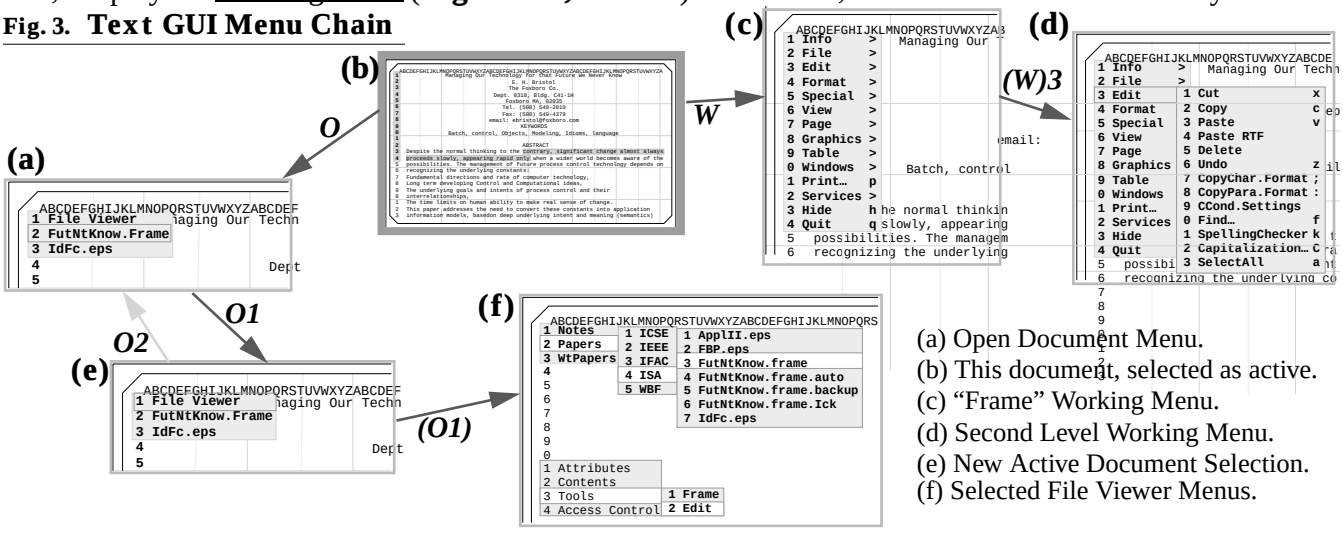
To provide a counterpart of the different classes of menus, we need (a higher level menu or) a small num-

ber of dedicated Function Keys (small enough to be easily remembered or supported by special labeling of the conventional control or function keys):

O; Select a menu which displays the menu of Open Documents (**Figure 3a, e** below). This menu allows selection of a new Active Document.

W; Display the Working Menu (**Figure 3c, d** below) in full form, and allow an item selection by number.

Fig. 3. Text GUI Menu Chain



(a) Open Document Menu.
(b) This document, selected as active.
(c) "Frame" Working Menu.
(d) Second Level Working Menu.
(e) New Active Document Selection.
(f) Selected File Viewer Menus.

D; Display a Dock menu.

S; Select a file by name, as in ***S* FutNtKnow.frame**, or a text string, as above. (An insert position might be selected mnemonically as ***S*<3L** (select before 3L) or ***S*>4M** (select after 4M).) The use of this function requires an open text line to display the function text.

M; Prepare to Move the selected item (file or text string) to the next selected object or location (directory or insert position).

Typical users would not find the TUI anywhere as compelling as a GUI. Even so, the TUI completely duplicates the remaining capability of the true GUI, including its mnemonic capability to remind the user of the available selections. For (text) applications where the user must shift back and forth between keyboard and mouse, the TUI would be actually easier and faster to use. In fact, many pre-GUI interfaces approached this kind of menu use.

What they failed to do is define and support the universal platform design, illustrated above, which the Macintosh®, and NeXT® so successfully embodied (technically if not commercially)[1]. These newer systems allowed their applications to be developed into a common platform framework with a universal background support of housekeeping capabilities (cut/paste, file manipulations, screen manipulations, spelling checkers, dictionaries) in an ever more general and easy to use "Look and Feel". Had someone succeeded in making earlier operating systems provide a TUI, with a uniform set of menu and functional support capabilities, how much easier it would have been to switch later to a mouse based technology.

So the problem is the old fashioned Catch 22. How could one have got some Alternate World IBM and Microsoft to start with an operating system whose conceptual modal allowed for the later support (**not just interfacing; SUPPORT**) of a mouse and uniform housekeeping functions. It is not as if pointing devices

[1] These technical qualities are far more important than usually credited. When the NeXT platforms first came out, developers found that their application effort could be carried out in one tenth the effort of other platforms. This improvement in efficiency in a more competitive market could mean that applications were effectively developed by the much smaller groups, able to advance the technology much faster, with fewer false steps. The field of competitors could be much larger, with many more opportunities for differentiation. Small markets could become more viable and specialty markets (like ours) could be much more precisely served.

or their attractions were not familiar in the technology. At that time there would have been a legitimate argument over relative pointing device economics or the ultimate winner, but not the need or feasibility. In lieu of such a model, there was never a possibility of a smooth TUI to GUI transition. But with sufficient user impetus (such as could apply in an industry like ours)[2], these characteristics could have been developed in their proper economic order.

IDIOM DISPLAY EXAMPLE

Several of the references introduce Idioms to simplify control[1,2] and integrate the entire control environment[3-6]. Idioms illustrate the role of a special purpose Information Model in automating the connection between several distinct operational functions. An Idiom is a standardized pattern of control elements[3] which:

- Is based on a standard control function or problem, expressed as a broad process control intent; implementation is compiled from that intent,
- Computes the needed control action to be applied to its output Variable targets from the input and output Variable Attributes,
- Derives all target and state data uniformly from input and output Process Variables, automatically using (Setpoint and Measurement) Attributes appropriately without formally distinguishing their data connections,
- Is cognizant of its involvement in control degrees-of-freedom to automatically recognize loss of control over output Variables and restructure the control appropriately, and
- Is designed to apply to any set of Process Variables as a generalized control operator in Idiom Loop statements.

As an example, the Idiom Loop statement, **T100AV**$_{\textbf{REGULATE}}$ **P100**$_{\textbf{HICONSTR}}$ **F100**$_{\textbf{REGULATE}}$ **V100**, expresses the cascaded control of an averaged temperature variable, **T100AV**, following a degree-of-freedom path, through the Flow variable **F100**, to the valve **V100**. The **REGULATE** Idiom operator represents normal regulation (nominally with a PI or PID controller). The **HICONSTR** Idiom represents a high constraint override (nominally implemented with a PI controller and selector), applied to the pressure variable **P100**, which will override the control path to **F100** under any constraint violation. While the basic purpose of the Idiom and Idiom Loop statement is to provide a simpler (and more language integrated) control description, the higher level of operational intention modeled by it automates related support.

Conventionally, process control displays are based on a fixed layout of a fixed display area. Conventional text environments modify this restriction with word wraps. But the more general concept of pretty-printing is rarely used. Figure 4 is a demonstration of pretty-printing of listings, back compiled from the execution data base corresponding to a simpler Idiom statement (with tuning parameters), to a Ramp (Theme) statement (defining a progression of temperature rampings, illustrated further below), and to the language "SuperVariable" definitions. Note that the columns are arranged automatically to accommodate varying Attribute sets, and largest entries.

Figure 5 generalizes the pretty-printing to graphics objects showing two possible operator displays derived from different policies (the first based on faceplates, the second on trends) that might be automatically generated from the original Idiom statement. The displays of Figure 5 would be automatically generated to take account of the number of required faceplates, trends, and control function icons, and the available space. Each display allows for the full display of operating states as well as the display of the override/non-override states implicit in the constraint structure.

---

[2] **and user vision.** Here my dream becomes more tenuous and less convincing.

[3] Implemented in any technology convenient, not necessarily including the conventional blocks.

Fig. 4. Pretty-Printed Back Compiling

```
LOOPS:
 T101            F101            V101
     REGULATE[1]     REGULATE[2]

             STATES    PB        INT        DER
        [1]  AUTO,+  200.000  2.833 MIN  0.000 MIN
        [2]  AUTO,+   5.000   1.667 MIN  0.000 MIN

THEME_STATEMENTS:
  RAMP T101:          START: FOR VENTTIME; †[1]
                      HEAT:  TO COOKTEMP + BIAS IN HEATTIME;
                      COOK:  AT COOKTEMP FOR COOKTIME;
    RAMP T102.VALUE: COOL:  TO 0.000 *[2] IN COOLTIME

   †[1] T102.VALUE = COOKTEMP
   *[2] T101 = COOKTEMP * (T102 / COOKTEMP) ^ 2.000

 DEFINITIONS:
             NAME          NAME            IN    VALUE  MIN        MAX  UNITS  SET  HI  LO   DEV
             T101       TEMPERATURE_1     ECB1     _    0.000  250.000  DEGF    _   _   _   10.000
             T104       TEMPERATUREAV      _*1     _    0.000  250.000  DEGF    _   _   _   10.000
            T123456  TEMPERATURE_1223456  ECB23    _    0.000  250.000  DEGF    _   _   _   10.000

             NAME     NAME       IN   CONV  VALUE  MIN        MAX  UNITS  SET  HI  LO   DEV
             F101  OUTLETFLOW1  ECB2   1      _    0.000  250.000   GPM    _   _   _   10.000

             NAME  OUT   VALUE  MIN        MAX  UNITS  HI  LO
             V101  ECB3    _    0.000  100.000          _   _
```

Fig. 5. Idiom Generated Displays

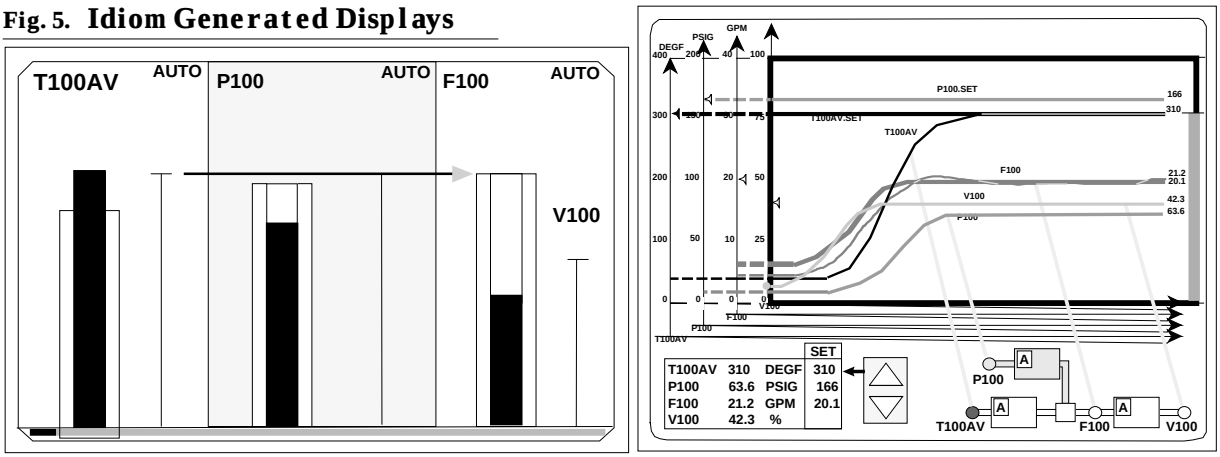
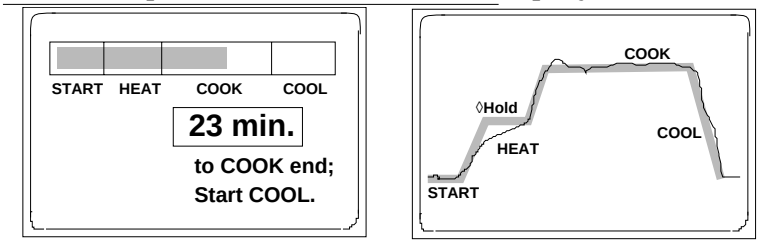


Figure 6 shows two similar policy determined, operating displays, for the Ramp statement. The benefit, in

Fig. 6. Ramp Statement Generated Display



each case is more uniform displays and automatic evolution to any new display technology.

INFORMATION MODELS AND OBJECTS

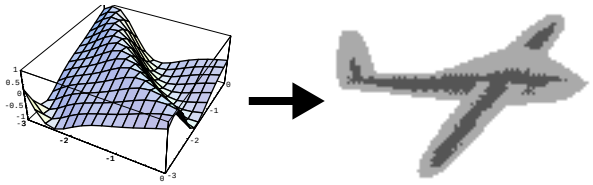
Everything should be as simple as possible, but no simpler. Einstein

Modeling is a triteness of the teaching of our field and a rarity in practice. Good modeling requires talent. The good scientific models that confirm the feasibility and utility of fundamental technical models, like the Information Models called for here, were not the incidental application models of some simple analysis, but those authored by people like Newton and Einstein. The essence of these models in their application is the simplicity of their basic expression, supported by a large body of technology for defining the nuances and variations of the model, and an equally large collection of different examples of their use. The Navier Stokes model is simple. But, it is backed up by libraries of mathematical and computer techniques for elaborating and solving it, and applications, like boats, airplanes, and even process components, located throughout the world.

In any given situation, absent an Einstein to create the Process Control Information Model that we need,

Fig. 7. Models are Simple; The rest is Hairy

$$\rho \frac{Dv_x}{Dt} = -\nabla p + \mu \nabla^2 v + \rho g$$



we can still understand the seriousness of the effort, and the cost of failure. In our approaches to standards and advanced technology we make models which are too complicated and inflexible. The role of the Einstein has been to get both simplicity and flexibility. Either failure involves deep and hidden costs, causing expensive and inadequate control solutions, and a massive Catch-22 to progress. Does any of this sound like the software that we are used to? It should.

Objects provide special mechanisms for supporting information modeling:

- Methods: special functions which encapsulate the Object's data from unintended access,
- Polymorphism: the ability of differently structured Objects to appear similar on the outside, and
- Inheritance: the ability to base the structure of more complex Object variants, on simpler Objects.

Encapsulation is a negative concept, like the restrictive and inflexible standards which we usually impose, and not new to Objects. Polymorphism and Inheritance are more positive, permitting an essential basic Object abstraction, to be specialized to many different detailed situations, each expressed in its own derived class Object. As in the case of the elegant math model, the base Object can emphasize simplicity over rigid specification; the environment can refine and elaborate the simple analogy to any needed level.

The classic math models are not all based on one mathematics, and our needed Information Models will not all be based on Objects. The TUI and Idiom examples are not essentially related to Objects. The next example involves a simpler structure than Objects. More generally, the models are more likely to be based on larger relations between entities which may, incidentally, but only so, be implemented as Objects.

SIMPLE MODELING; THE ALARM PROBLEM

Even simpler than Objects are lists. We use them in many ways, as: Categorizations, Directories, Vectors, Matrices,…. When Directories are nested, lists become the basis for hierarchical structures, which are another very simple modeling vehicle familiar to all. For an example of simple lists used for deeper Process Control Information Modeling, consider a basic Process Control Problem: Alarms. This problem includes many dimensions. But one of these is to give the operator the greatest high level view of the distribution of alarms in the process in such a way that he has the most control over his information.

The operator should be supported like an airplane pilot on a clear day, able to see for miles, able to select his direction of sight and level of detail, with a simple change of visual focus. The pilot can see large and small elements: cities and buildings. He can organize his view in many ways: in terms of the many towns and cities, or the roads running through these cities, or the storms and other planes on their own courses. From this wide ranging perspective he can support any decisions about his own course.

The control application vocabulary is centered about the process variables. Using Category lists we can define a wide range of alarm/variable groupings, forming a much wider operations vocabulary. An organized set of Categorizations can be very powerful, based on:

- Process subdivisions: Reactor, Distillation tower, Chlorine plant (the traditional process hierarchy).
- Fluid streams: Product, Utilities, Steam, Lights, Heavies.
- Situations: Startup, Normal run, Heating phase, Pump problem.
- Operational Impact: Safety, Equipment jeopardy, Product quality, Production show stopper.

Each of the above Categorizations is "orthogonal" to the others. Consider how this simple modeling concept can be applied in five ways:

1. To allow the conventional hierarchical display selection organization.
2. To allow the operator filtering of displays. Figure 8 shows four lines, corresponding to the above Categorizations. Each line highlights one or more selected Category, indicating an operator selection. Whichever main alarm display (Log, Trend, Summary) is selected will include alarms only from Categories which have been selected on each line.

Fig. 8. Operator Controlled Display Filter

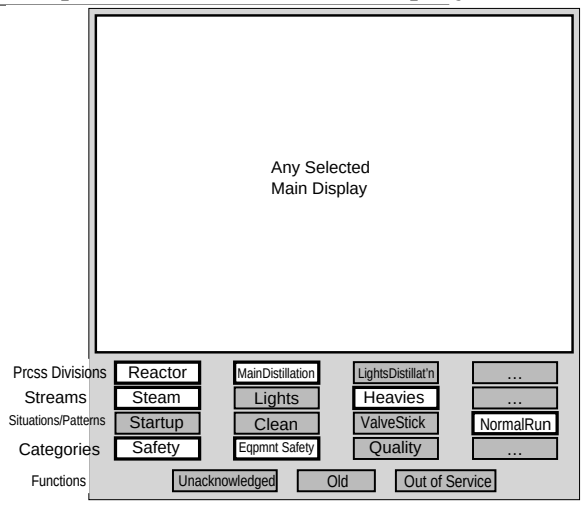
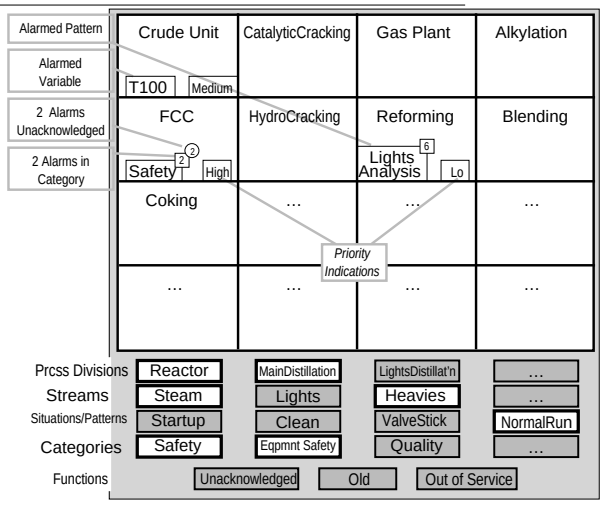


Fig. 9. Matrix with "One Word Summaries"



3. To support "One Word Summaries". These are computed as the name of the smallest Category which contains all of the alarms in a selected area. Figure 9 shows a Summary Matrix, with three examples of "One Word Summaries" each in a separate Indicator box:
   1) "T100": The Crude Unit Indicator is indicating only one alarm; only its name is needed.
   2) "Safety": The FCC Unit Indicator is indicating that all included alarms fall in the Safety Category.
   3) "Lights Analysis": The Reforming Unit Indicator is indicating that all alarms have to do with the particular Situation Category.

   The purpose of the One Word Summary is to provide the best alarm summary possible within a limited information content. As a situation develops, the One Word Summary shows first the initial alarm (as with T100 in item 1), then the name of a small Category, and then the name of a larger Category, etc. As the situation develops the abstraction gets broader. This has the effect of allowing the initial detail to show up at the topmost display level. The operator data load is held constant by varying the abstraction level.[4]
4. To support alarm priorities based on Categories. Priorities become much more meaningful if they are based on named operational Categories, rather than numbers (Safety, Quality, etc. vs. Priority 1, 2, 3, etc.).
5. To support group alarm suppression based on the priorities or control automation, to minimize showers of less pertinent alarms.

This section illustrates the power of a very simple form of Information Model. The earlier Idiom model was equally simple, but that model's meaning is completely dependent on the underlying degree-freedom structure. The Category model lends itself to modeling many situations, with the application meaning wholly determined by the choice of Categories and their use.

CONSTANTS IN THE EVOLVING DISPLAY TECHNOLOGY

A traditional dream of process control systems designers has been the "Brain Cable", a direct connection between the brains of operating people and the process control system. Mechanically the highest data rate

[4] As another example of this usage, users could define one or more Categories of Noise alarms (that tended to be always and irrelevantly on). This would allow the operator to filter out these alarms by de-selecting this Category. But whenever he forgot to do so, if no other alarms were active, the One Word Summary would show "Noise"!

input to the human being is through the eyes. The output most naturally integrated with that input is the change of focus or eye direction. No other vehicle for accessing data matches the ability to simply select it by looking at it.

The console has been our connection to the eyes. The conventional console display most lends itself to a single integrated display, a small process graphic, or layout of small faceplate or similar indicator displays, with a few separate set aside Operator entry lines, summary boxes, and pull down menus. A large data base is examined by moving from display to display. Thus on the small screen, the data is distributed among a large number of called up screens. The small screen is best attuned to manipulating data.

The challenge is to design an Information Model of the display process which captures the broadest display goals in such a way that the porting, or automatic re-compilation to various displays is possible. While the ideal would be to focus on operating goals independent of technology, we also need to look at opportunities suggested by the technologies themselves.

On the face of it, console screens, virtual reality headpieces, and large screens could utilize their single displays identically. But wall size screens[5] and virtual reality displays[6] offer a higher "data rate". Both the large screen and the virtual reality suggest consoles that people might "walk through" to look at the process. (A steering wheel and accelerator could support an analogous perspective with a CRT.) This is another way where the limited physical display, nevertheless, effectively supports an open ended data access. In this case, the virtual world (like the real world) is always all out there. There is no need to keyboard and mouse one's way through the GUI or language to get to desired data. One just has to "walk" to the data one needs. This suggests a quite different data layout, with data distributed throughout the virtual world.

One of the directions virtual reality is pursuing is shared virtual worlds in networks.[7] This activity is generating whole new conceptual elements that one must include in any model of future displays:

- Agents and Bots (as in Robots): programs, of varying intelligence, that can be turned loose independently in the system (or network) to carry out some requested task without further human intervention.
- Avatars: representations, of any other person or Object (including bots or agents) active in a virtual world interaction, which are the displayed expressions of that person or Object.[7]
- Virtual Reality (VR) vs. Virtual Environment (VE): Combined technologies intended to represent virtual worlds which a user can interact in intuitively, differing in the intended fidelity of the result. VR goes as far as it can; VE aims for greater practicality, only trying to suggest the virtual world.
- Distributed Virtual Environments (DVE), Multi-User Dungeons (MUDs), and Multi-User Dungeons, Object-Oriented (MOOs): Computer controlled environments, representing virtual worlds, in which several people, working on their own consoles, can interact. Each console displays its own view of the virtual world.
- Computer-Generated Force (CGF): In military simulation, a system which generates many coordinated agents, interacting together and with the user.

We don't now imagine operator interaction through the console, but it might be useful. One dimension of moving into the virtual world, is recognizing uses that such a function might play. This can be modeled

---

[5] Now costing $150,000 a pop, but in five years,???

[6] We will ignore the secondary use of the virtual reality display to show three dimensions, and the present day technical limitations of lower resolution and eye fatigue effects in certain displays. We will also ignore a large number of important developing standards, such as Virtual Reality Modeling Language (VRML) and Distributed Interactive Simulation (DIS) of varying degrees of sophistication. These are the transitory developments that our model must see beyond, modeling the constant information requirements to be translated between the different technologies.

[7] Virtual reality systems will include many strategies for making these Avatars interact realistically so that as the user "moves" through someone else's world, the sounds and appearances will reflect the way the real counterparts would act if the reality were "real".

now in conventional displays without waiting for avatars, as email, or as special indicators representing a waiting message or a button to invoke some function.

The wall screen can be used in several ways:

- As one huge process graphic with dynamically operating indicators, or a wide screen process video, stored movie, or virtual reality display.
- As a large screen echo of the console, to be seen from anywhere in the control room.
- As a dedicated display with different regions laid out to represent different classes of conventional screen display, with areas for: video monitoring, instruction videos and procedures, and standard indicators.

In any case, on the large screen the data is usually all there, only infrequently being overlaid.

The large screen presents its special accommodation problems. The Summary Matrix of Figure 9 organizes the process hierarchically for the small screen. A large screen could display the same data in a flatter hierarchy, with more Indicator boxes at each level, including several Matrix Category Summaries in the same display. On the other hand, there may be a maximum size to the Matrix, beyond which it becomes confusing. Policy should then specify the maximum acceptable Matrix complexity.

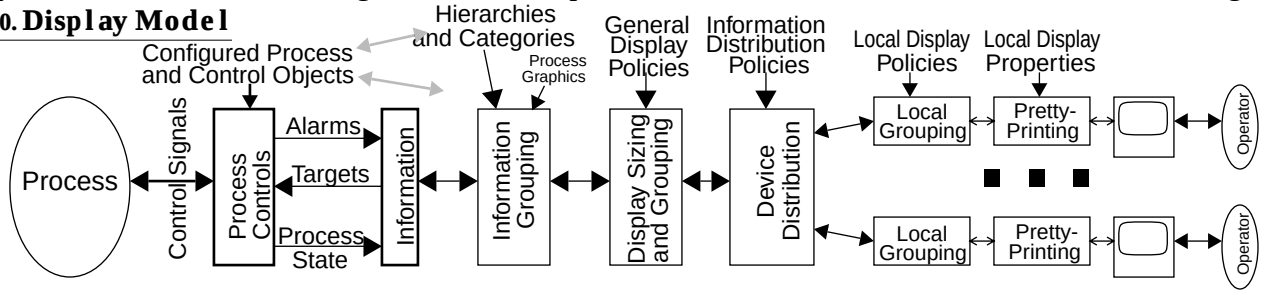
The ability to generate displays automatically to fit the control function, display device, and display policy introduced in Figures 4, 5, and 6 will be crucial to accommodating the above range of displays. Automatic display translation will better bridge the proper use of different co-residing display forms in a same system. Fixed form displays will always mean that each new display environment will have to be hand tooled to each application. But separate hand tooled displays for different co-residing display technologies will be a disaster.

Configured policies to control the automatic display building will be the key. Certain policies will define universal properties (e.g. optimum character and icon sizing). But different jobs will support different preferences and abilities in the amount of data that they can accommodate. Displays generated under a universal display model will need to support declared display policies adjustable to different users, such as specifications of the amount of information acceptable for display. The One Word Summaries of Figure 9, or the strategies proposed generally for suppressing alarms based on priority (or on causality), illustrate controlled information content display.

Similar configured display policies will also be needed to optimize the distributed information when several different kinds of display are available in the same system. Figure 10 shows a multi-technology, multi-console, multi-display operator display structure, organizing information display requirements independent of the display resource. The Information Model arranges the process and its operational and state data to reflect the natural process structure and control intent. This data is supported by process graphics and other redundant formulations which the system is unable to generate for itself.

User displays develop as a consequence of interaction of the physical display device properties with the configured display policies. The information moves through this structure like passengers streaming to a transportational terminal, seeking available transport. There is the same double benefit to this enlarged

Fig. 10. Display Model



model: whatever can be automated to generate easily configured displays can also be re-compiled to fit new

display technology. With it the operator is presented with broadly predictable display. The industry is presented with a constant application strategy in the face of changing technology.

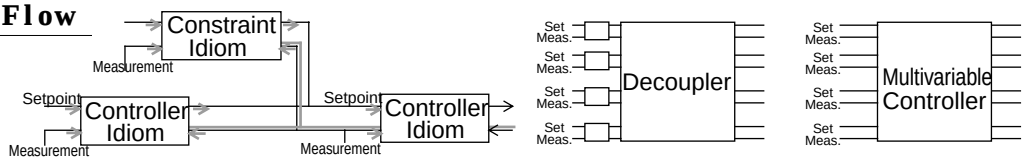
LINEAR NONLINEARITY, THE ANSWER TO AN OLD NUISANCE

God made integers (linearity?), all else is the work of man. Kronicker

Process control, particularly the chemical based process control, has a precious capability to use simple analytic nonlinear models at that higher level of dynamics which truly affect the process operation and economics.[8] The modern emphasis on linear predictive control ignores this capability, and the interest in things fuzzy and neural abdicates it. In those processes with widely changing operation, nonlinear compensation and feedforward can do much more to improve performance than any linear dynamic tuning that we could bring to bear. The problem is that the nice, neat control loops (and regular academic matrices) are hidden by normally represented nonlinearity. But the structural linearity of the cascaded degree-of-freedom Idiom Loop statement can restore the clarity.

The worst source of the complexity is the necessary design interaction of analytic nonlinearities with the normal valve limit and constraint degree-of-freedom behavior (as illustrated by the P100 override of the T100AV controls shown in the Loop statement of Figure 5). The conventional technology accommodates internal computational disruption, using a bilateral flow of information to the controller from the valve side as well as from the measurement side. The Idiom generalizes this concept, as in Figure 11, treating every degree-of-freedom connection as having dual data flow.

Fig. 11. Two Way Idiom Flow



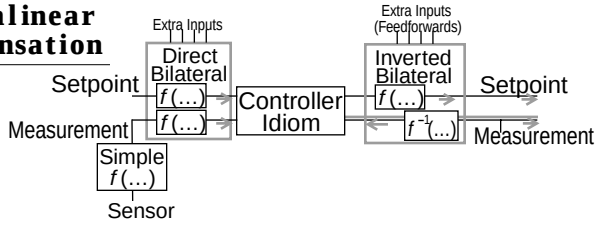
On the Measurement/Setpoint side, both data attributes naturally flow toward the controller, permitting control error based corrections. On the controller Output side, the output attribute flows away from the controller to the valve or secondary controller. The feedback attribute communicates the process state back, providing the controller information about the effects of the down stream process limits and constraints. This information allows Idioms to be designed generally so that they can shift to an alternative control action in the face of any constraint override.

Analytic nonlinearity in this structure is designed one way or other to compensate the effect of the Measurement or valve side data so that the basic linear controller is fully compensated to the process character. Figure 12 shows three roles that it may take:

- Simple measurement compensation, where the nonlinearity compensates the single sensor signal or state value to develop the intended measurement value.
- Direct Bilateral compensation, where the setpoint and measurement represent the natural measurement units, but introduce a nonlinearity into the cascade. In this case the compensating nonlinearity is applied in parallel to both setpoint and measurement input signals.
- Inverted Bilateral compensation, where the output represents a nonlinear effect passed to the valve or secondary loop. In this case the compensated output will define the value that the cascaded controller drives the process to achieve. In order to make the resulting feedback compare naturally within the primary controller the compensation must be inverted in the reversed (feedback) path.[8]

Why is this an Information Modeling issue? The easy nonlinearities for our use are analytic equations and breakpoint functions. In this day and age both are easily and accurately inverted. A control configuration tool or language should allow one to configure an Idiom Extender as a breakpoint function or algebraic equation configured as part of an Idiom expression in Direct or Inverted Bilateral form, or applied to di-

Fig. 12. Three Kinds of Nonlinear Extender Compensation



rectly to any variable to compute the appropriate compensation. The duplication or inversion would be implemented implicitly and invisibly. This is how nonlinear control could be a straightforwardly added on to conventional feedback control.

CONCLUSIONS

Process control could now solve all its basic technical problems. Two things stand in our way: control complexity, and a world running in circles. Process control is based on eternal verities, which could be finally captured in organized Information Models. This would straighten out those circles and slow down our part of that world. Unfortunately we have had trouble differentiating implementation from essential goals on which such a Model would be based. Hint: We regulate and constrain our process variables, not play with PID blocks, selectors, or DMCs. Capturing our application configuration in higher level goal models that transcend any particular implementation tool will have three benefits:

- It will make the design more transparent and easy to use by all participants.
- It will decouple process control more from its computer world implementation.
- It will support new ideas as they become available without upending the basic solutions.

One shot at some really high quality computer science thinking might mean that we might never need to look at it again.

Bibliography

[1] E.H. Bristol, “Strategic Design: A Practical Chapter in a Textbook on Control”, ‘80 JACC, San Francisco, Aug. ‘80.

[2] E.H.Bristol, “Rules, Statements, and Idioms”, 17th Annual Advanced Control Conference, Purdue University, West Lafayette, IN, Sept. 30 – Oct. 2, ‘91.

[3] 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.

[4] E.H. Bristol, “Not a Batch Language; A Control Language”, World Batch Forum, San Francisco, May ‘95; also ISA Transactions, Vol. 34 (1995), pp. 387-403.

[5] E.H. Bristol, “Redesigned State Logic for an Easier to Use Control Language”, World Batch Forum, Toronto, May 13–15, ‘96; also ISA Transactions, Vol. 35 (1996), pp. 245-257.

[6] E.H. Bristol, “Batch Object Modeling and Language”, World Batch Forum, Houston, Apr. 27-30, ‘97.

[7] R. Braham and R. Comerford (Editors), “Sharing Virtual Worlds”, IEEE Spectrum, Mar. 1997, pp. 18–51.

[8] F.G. Shinskey, Controlling Multivariable Control, ISA, Research Triangle Park, NC, 1981.

[8] The function being inverted is the function between the controller output and the compensator output; and other extra inputs or feedforwards are free variables which act line parameters in both the function and its inverse. Thus if the function were $y = f(x;(u,v)) = e^{uvx}$, the u and v variables would be treated as parameters; the inverse would be $x = f^{-1}(y;u,v) = \frac{\log(y)}{uv}$; any feedforward compensation occurs outside the inversion.