EHB[II] at Foxboro

Background:

The following items from my website ((http://homepage.mac.com/ebristol/) provide general background on me and my attitudes to process control:

Bibliography:

- http://homepage.mac.com/ebristol/PDF/EHBBib.pdf.

History:

- http://homepage.mac.com/ebristol/PDF/PrcssCntrlHstry.pdf
- http://homepage.mac.com/ebristol/PDF/ISABrstlHstry1.pdf

History of Process Control

Process Control, in modern terms, dates back to the beginning of the century; in other terms, back to the beginning of civilization. But I have sometimes said that there were processing industries and process controller industries but no process control industries. Process Control is what happens when processing companies get together with process controller companies.

Initially Process Control took the form of independent "boxes" patched onto the processing equipment. Depending on the industry, these boxes were pneumatically powered or electrically (in the power industry). Foxboro controllers (and sensors and recorders) were pneumatically driven. There was also a distinction between the controller manufacturers which tended to be generalists and the sensor specialists (e.g. Beckman).

An important characteristic of "boxes" is that the manual controls had to be built in. This was important because a danger in digital control was a loss of focus on good manual interfacing design, as the interface became separated and managed on its own.

Relevant Foxboro History

Foxboro was a mechanical/pneumatic company which had one advantage in refineries and chemical plants of being fire safe. E H Bristol[I] represented a prototypical designer, but eventually Hoel Bowditch represented the elegant possibilities, plus other clever gadgeteers. In the 50's electronics really took hold with a slightly more flexible form. And then systems in integrated panels.

The instincts of the earlier leaders were slightly stretched by this technology, leaving Foxboro with fewer product inventors, one being Everett Olsen. But this was nothing compared to the digital era, where a succession of leadership was stressed by loss of the intuitive connection.

The computer programmer was completely out of tune with the mechanical product designer. This was a great shame because they had much to teach each other about the future opportunities of process control. My view is that the industry still has a long way to go to achieving its proper potential.

As it Developed; My View

Process Control is a complex industry by having many divers elements and attitudes. The complexity is usually derived from the large numbers of divers parts rather than from inherent sophistication of the whole. It is true that some of the processing technologies are highly sophisticated.

At the same time individual practitioners often approach Process Control in a simple minded way. Characteristic of the best people is an extremely broad/general perspective, often at a cost of limited expertise in useful specialties.

A Process Control plant is like a town or city which the ideal application design must

address not only in terms of the needs of one user, but the needs of each stakeholder. The usual designer is more specialized. The classic plant may take years to actually find one's way around.

The classical Foxboro was a product oriented company, where the application/ systems (Shinskey-style) expertise was established to support the sale of the products. Moreover the different application aspects called for different mutually unrelated experts. As we went to electronic and digital systems we lost this product point of view and came more and more to emphasize the applications and systems over a grasp of the product expertise facilitating them.

What is the difference?

- A product is designed once to be manufactured and sold in many standardized copies, so that the initial design cost is amortized over the many copies. Refined design elements, protected by patents and other strategies are used to create an especially attractive design character and protect its commercial expression. Refined manufacturing techniques are developed to reduce cost and further protect the design.
- A system is designed to achieve lowest cost design and implementation of the single system (with at most few copies) by using simple standardized design practices, and inexpensive standard parts. Within this perspective, practices and parts need to be non-proprietary, to the extent that this is required to allow the entire user community to use the same basic underlying application thinking.

Systems expertise thus emphasizes fast inexpensive one-off design, potentially implemented by "anybody", and de-emphasizes unnecessary refinement.[1] Product design expertise emphasizes refined best possible design for the largest body of potential users. Kept at a proper distance there is an enormous opportunity for synergy as application designers develop opportunities which can be generalized to more broadly useful product possibilities.

Placed too close together, these are distinct, generally incompatible, attitudes; the practitioners usually do not understand the differences. Moreover the entire digital community has been mostly oriented about standardization across all manufacturers; proprietary properties are usually insignificant.

In time, the resulting resistance to inventive product design was further exacerbated in our business by the later managerial emphasis on succumbing to "best practices" (of competitors).[2] The resulting business would be ideally more that of a consultancy or a contractor, reducing support for quality product design.

My goal was to redefine more appealing (Jobs/Apple like), proprietary systems support products (EXACT™ being just an initial low level example) with expanded or better process control function. Most recently my goal has evolved to address the difficulty the dif-

[1] Exacerbating the differences, the systems design environment makes systems designers impatient with the limitations of product prototyping and the amount of time required for full product design.

[2] In my ideal world there was also the problem of competing with my colleagues on each other's innovations. A marketeer once talked to me about any adverse effects of the competitor's different innovations. We then agreed that this was not all bad; more than two worthwhile innovations at a time was unlikely and these would just split the market, before the passing years sorted out the difference.

ferent control engineering and user personnel[3] have in understanding each other and working together to create the necessarily complex systems.

This goal called for products whose concepts made systems application designs and usage more standard, self-documenting, and widely understandable, and made the different application dimensions more universally accessible. Admittedly, the results might seem to my clients to be incomprehensible, because it usually depended on change or automation of traditional practice to broaden it or improve it.

This would not be about eliminating different workers but permitting them to better understand each other and work better together and with less redundancy of effort within projects and from job to job. The strategy derived special computer languages which lent themselves to more easily understood application programs.

Personal history

I grew up in Foxboro as a shy, non athletic kid, with many personal technical and sundry interests, spending summers at Maine camps (which didn't change this basic character even as it gave me a broader outlook) and visiting relatives in the midwest. The final three years of high-school were at Deerfield Academy (also not affecting the basic character but adding breadth: athletics added skiing and soccer).[4]

By MIT I was able to hold my own and take up fencing as the top epee-ist. Unfortunately a Deerfield eduction made my initial MIT experience too easy [I didn't buckle down (from their point of view) until being kicked out my junior year and spending a first senior year at Beloit College (where I won the Math award)[5]]. A second MIT senior year followed, but the MIT benefits came into play only years after. I am now sufficiently outspoken to be an embarrassment to my kids.

One of the Deerfield teachers (and father of Dick Hatch, a later Foxboro colleague) had observed that good schools were made more by good students than by good teachers. While I had outstanding teachers at both Deerfield and MIT (and Foxboro!), I was largely independently taught. And I do believe that students (and their parents) do more to make a good school. Good teachers are then attracted (and vice versa). I admit that there have been exceptions in the world.

At Foxboro

After MIT (in '58; class of '57), and Army Ordnance Summer Camp, I worked six months at Daystrom Electric on Navy sonar research [and my first "computer" (the paper tape driven IBM610)], and six months in the Army as a second lieutenant. I came to Foxboro in November '59, doing electronic product design in research. Research, under Mead Bradner was what Bruce Hainsworth called an invent-ory! I shared an office with Jim Graham where I could see Hoel hold sway, with his several assistants. Later Hoel became my frequent test victim for explanations of these weird digital opportunities.

My first significant project was under Everett Olsen, suggested by Bill Howe, on a proposed square rooting integrator. The design

---

[3] And other stake holders like sales people and managers.

[4] The headmaster, Frank Boyden (as in the Library) was notoriously a Foxboro boy!

[5] A Foxboro biased history doesn't do justice to my mother's side! While one grandfather co-founded Foxboro, the other was President of Beloit and other Colleges and Chancellor of Universities in the mid to "wild" west.

was interesting (based on number theory) but impractical, beaten by a more direct design I should have anticipated. A first paper resulted, with an academic review (a single sentence: "I like it.") untypical of the later academic battles!

After several years, I had had more digital experience (on an IBM 1620 and CDC160), and related to others outside of research. I moved over to work in Sam Goodhue's group, whose individualistic engineers (e.g. Greg Shinskey, Manny Freitas, etc.) supported special projects.

About this time the pioneering first desktop programmable calculator came out, called the "Mathatron". This used large core (1/4" magnetic cores!) memory.[6] My comment: "Cute toy, but I'll never use it". So they put it on my table and within a week I was doing all kinds of poor man's computing, including, later, some of the first experiments that led to EXACT™.

I got to know the designer, Bill Kahn, based in the old Waltham Watch Co., as a committed user, trading ideas on this precursor to personal computing. The programming was similar to the earlier IBM 610, implemented in a form I called Learn/Do programming, where the user entered a calculation, which the calculator then recalculated on command.

This strategy has been used with robots and in other places including my later operator interface language.[7] A later influencing personal computing and language exposure was the BB&N, Joss-based, TELCOMP system, which "lost it" to the inferior BASIC.

Several experiences made me feel that an electrical engineer/mathematician, even with good experimental intuition, really didn't belong alone in the field. This may have been selfish and unfair; the application engineers, particularly on the early digital jobs, were often sent up the creek without a paddle. But in my later prototyping/demonstration days I often found myself in endless repetition. Better to use the effort to demonstrate that others could make "it" work too.

Among my jobs for Sam were feedforward systems with Greg. One of his techniques was to trim model parameters with feedback controlling the critical measurement. Only one parameter could be trimmed, usually a gain or bias. Why not be able to trim several? I invented, and later published and patented, the robust Two Parameter Feedforward Adaptive system, remarkable in several respects:

- In a usually highly mathematical subdiscipline (adaptation/self-tuning, like EXACT™), this could be implemented pneumatically!
- It implemented a Gauss-Seidel solution, switching feedbacks based on the direction of feedforward measurement changes.
- It set my rough direction in the later invention of the much more powerful, multivariable Moment Projection EXACT™ extension.

Regrettably, following my above strategy, the final field trial was carried out under an

---

[6] Magnetic core computer memories, collectively invented by An Wang at Harvard and Jay Forrester at MIT, were hand strung, usually out of very small doughnut-shaped cores. Quarter inch cores could be strung more easily and cheaply (and flexibly!).

[7] Regrettably in taking the strategy beyond its natural limits they lost out to a more simple-minded design by the more powerful Wang Labs competition (Windows over Mac!), which then also later "lost it" in turn.

experienced field hand, who got the switching wrong (so the trims diverged). I should have been there! Later, when the publisher scrambled sections of the paper, the same subtle error showed up in a review paper by Bud Keyes, a well known competitor.

RGA History

One of Sam's people, John Chang, and I had been talking before about a book on "multivariable control" (by Mike Mesarovic at Case western Reserve).[8] We thought we could do something with this. As a result he got a chance to do a thorough experimental study of a power plant. Then he left to get his PhD. And I was assigned to make sense of the extensive report and measurement response records he had made.

I had a number of simplified false starts at trying to characterize the ways in which these records expressed the interaction of the power plant elements. I then (this was '64-'65) came up with the RGA.[9]

The RGA is a table (array) of numbers indicating the way in which the interaction of the different valve/measurement pairs are affected by the remaining controls. One benefit is that these numbers are dimensionless (not dependent on the measurement units), thus expressing more basic behavior. One of the characters of the array is that a "1" means no effect,[10] and large numbers indicate control difficulty or impossibility.

Shortly afterward Greg was in a refinery discussing a control function that wouldn't work properly. He calculated the RGA and come up with the number 26 to a key element; effectively saying that this was impossible. A believer! The RGA quickly became the starting point of his design procedure and teaching, allowing him to analyze where the problems are, and better develop his common sense designs.

Over the years more academically inclined engineers have advocated a succession of complex, math-based techniques. These tech-

---

[8] Multivariable Control addresses processes (like most) whose control is made difficult because of interactions between naturally separate parts.

[9] Relative Gain Array, Greg's more dramatic name for **my invention**. I'll never "forgive" him! Had he left it at "Interaction Measure" it would now be "Bristol Array"! Oh well.

[10] This is not entirely correct, confusing academics who didn't distinguish simple cross disturbances from deeper interactions addressed by the RGA and couldn't believe in a measure based only on steady-state effects.

niques would be held to be the latest thing against Foxboro's old fogy ideas. (Even in less standard areas, I have frequently found myself arguing with people's such latest idea which I had thought of and discarded for good reason years before.)

But, when used properly, the simple logical techniques of people like Greg, and Carroll Ryskamp, prevailed. And they are more easily taught, understood, and operated. Ironically, even the math biased engineers lack the basis for the kind of operational understanding that comes naturally with these more direct strategies.[11] Even now (2009), Greg avoids retirement, cleaning up some of these complex designs by RGA analysis and his basic feed forwards.

Many user engineers, even competitors, have found the RGA to be important. And many academic theses, books, and now web references address it. The first book totally on the RGA was by Tom McAvoy (University of Maryland), a long term advocate and friend.[12]

In the early to mid '70's several AIChE based sessions and an Asilomar Conference set out to bridge the gap, making way for the RGA and other common ground. These conferences enhanced both Greg's and my reputations and my broader understanding.[13]

Pattern Recognition Adaptation (EXACT™)

Even before this time I had been working on making controllers that tuned themselves. A number of schemes worked well,[14] but the one I called Pattern Recognition was simplest (even better, working on the PID[15]). This captured the control response shape digitally, altering the tuning to find a better shape, as would a practicing engineer.

At this time I moved back to the research department. Among the people that then worked with me on the Pattern Recognition (on a PDP7 and heat exchanger process) were

---

[11] This was confirmed at an early academic computer aided control design conference when Karl Åström (later) told me that he couldn't relate to the multivariable proposals, Later Tom McAvoy (below) introduced him to the RGA!

[12] I did have a few people (i.e. academics) who I got on with constructively; actually too many to properly list!

[13] I particularly enjoyed a description by Jim Douglas (University of Massachusetts), of his colleagues' reactions to hearing Shinskey talk: "He walks into the room, puts on his black cape and pointed hat, waves his wand at the process and the damn thing works.". This stood out after I returned from my LSU year, and Greg had "moved from his feed forward period (which I understood) to his selector period" and I could no longer understand his designs. From this arose the Idiom concept addressed later.

[14] The first of these was based on similar mathematics to that I later rail about, and I made it work much better (http://homepage.mac.com/ebristol/PDF/AdOdB.pdf) than my later competition. It just turned out to be needlessly complex. I **am** as much a mathematician as engineer.

[15] Proportional, Integral, Derivative; the more refined (but for academic and math-minded control engineers: simple-minded) thermostat-like controllers of process measurements, within what we call continuous process control.

Guzin Inaloglu, a lovely Turkish lady, and a summer student, John Steadman.

At the same time, Doug Payne's (Later a Fox-1 manager) group in Dick Sonnenfeldt's Natick digital computer division was attempting a similar effort (Between us we may have had 10 reasonable proposals.).[16] Later a Gunnar Nilsson worked for Doug and our long term MIT consultant/friend Henry Paynter, and before returning to teach in Copenhagen. Henry and I continued conversations on many subjects over the years.

In the later '60's a field competition was set up with Ernie Cohen supporting the Pattern Recognition; John Greenwood represented the "competition". The future EXACT™ worked fine, given PCP88 control software limitations.[17] We pushed to product-ize it many times over the years.

We had a short failed effort for the Fox-1 in Al Epperly's group, which was too bad because this group did very good work in supervisory control packages for that system. One time I had persuaded one of the marketeers just before he got fired.

I was becoming still more well known for my related work and papers.[18] This fit with the large amount of the academic self-tuning work. I continued to beat the drum into the '80's. At the same time I extended my adaptation experience and technology to cover process nonlinearity in areas not addressed by most people in the field.

Interesting problems with learning instability lead to the Simplex Modeling, which was an elaborate technique for representing processes in nonlinear n-dimensional space surfaces. This tied up a lot of resources (research and computational), eventually to be replaced by much simpler techniques in the '80's.

A point of confusion was my use of the term "Pattern Recognition" to describe the response shape detection. This contrasted with the much more difficult text/handwriting recognition usually applied to the term. I gave a talk to the local IEEE control group; one guy asked a couple of questions and then left!

At a control conference at the University of New Brunswick, a colleague confused the text problem with my simple solution. Greg was to give a major talk (on RGA related stuff). Talking to a well known German lady engineer, I expressed concern on how he would be received. She said: "Don't worry, the academics are afraid that you are right!"

Early Systems/Computer Interactions

Until the mid '70's, with my wide acquaintance of the older engineering management, I often found myself explaining or developing bigger picture computer/systems arguments with them. At a time of conflict with our early

---

[16] On of these engineers was Pierre Belanger, mentioned later, just out of MIT with his PhD, who went on to teach at McGill University, becoming well known in this field.

[17] It took us some years to evolve sufficiently flexible control software, for which I argued vociferously. In the PCP88 self tuning experiments, Ernie Cohen had to hand carry the tuning results from the upper machine FORTRAN program, to the lower machine PID because no-one believed we would need a digital connection!

[18] Post-patent papers were a key part of pushing Foxboro to commercialize the different works. Had there been no Pattern Recognition papers there would have been no commercial EXACT™.

smart-ass computer people, I wrote a note suggesting that systems were a necessary effort to apply mass production (of "boxes") to one of a kind applications.

Feeling that the older engineers needed to get a deeper first hand sense of what programming was about, I set up a demo of the basic 30+ word PDP7 RIM loader. In 15 minutes, I took them through the machine language, single stepping it. In this they saw the operation of subroutines and other basic concepts. Bill Howe's comment: "a bag of tricks";[19] but he understood.

In the early '70's, an older consultant, Gradon Smith (a designer of America cup ship gear!) was in an argument with a newer one, arguing against the computer. As he said to me, no one would use computers for such well defined applications as telephone switching systems. I showed him that it was already happening. He later apologized for getting mad!

The First Foxboro Computer Control Products

More recent Foxboro colleagues, unfamiliar with Foxboro's digital history, believe that EXACT™ could not have been applied at this time. In fact, we were probably more pertinently capable back then than now.

Our computer products started with RCA computers developed under Sonnenfeldt (who continued with Foxboro after RCA abandoned the effort[20]), then with the innovative dual computer PCP88 system. The upper computer ran the first Process Control FORTRAN (promoted by another individualistic chemical engineer, Ken Stapleford, who we inherited from CDC). The lower one ran traditional controls, unless it failed, in which case the upper computer suspended the FORTRAN supervisory functions and took over the control job (bumplessly?!).

There followed the Fox-1, -2, -2.10, -3's all of which were programmed for standard continuous control. At any time, we could and should have implemented EXACT™ commercially. Self tuning control had been one of the early (late '50's) justifications for digital control.

One of my battles from research addressed the Lilliput/Blefuscu-ian "Little Block" vs. "Big Block" issue. Our control data blocks were big, loop related, centered about a PID with all flexibility in a large set of parameters and flags (There could be several hundred of these.). This made expressing Shinskey-style controls awkward or impossible. For some groups, loops were easier for operating people to use. [This further relates to the effective standard human interfacing of complex controls; an issue that I resolved, for myself, only with the later Idioms.]

For most of this period, control applications were separated into analog (Sam Goodhue; there was also a separate power group) and digital (Doug Payne, then Al Epperly) groups. Thus Greg wasn't as close to the digital battles as he might have been, to defend the needed flexibility. And it took a while to get him forced digital! But I learned from all of this.

---

[19] Later Howard Rosenbrock, a well known, older English teacher and practitioner said the same thing when I showed him an early MIT language AI thesis (that I had also been showing around Foxboro). Ironical for me considering the "bag of tricks" solving the differential equations making up academic control. And then he went on to carry out his own AI efforts!

[20] Later an outside director of the company; Dick, apart from his RCA/NBC life, had an interesting World War II Nuremberg trial history.

I can remember guarded debates with Paul Griem when we hired him into Luigi Rispoli's Fox-1 group. Paul's glass experience argued for little blocks. A later hire, Liang Liang, now in Shanghai, who I still contact, had implemented little blocks for Bailey; the power industry has complex controls. By comparison to my "Little Blocks", they were probably microscopic; too flexible for us!

Earlier, in another aside, for the first PCP88 test job (Aruba), I was asked, by Bruce Baldridge, to put together plant simulations to test the standard control functions. Our TR-48 analog computers lacked the capacity to do major plant simulations. [I was later in charge of a larger analog/computer (EAI 680[21]) setup in the research department.]

So I put together a little box which allowed one to shunt every computer control outlet to a corresponding measurement inlet. Any of these shunts could be removed to insert some reduced size test simulation. I also argued that, for test purposes, these shunts were adequate test process substitutes; the simulations were not really needed. They took the box with them to Aruba for final test, as a precursor to the kinds of test rigs we later used on all computer jobs.

During the time that Fox-1 was being designed, a committee of myself, Greg, and Roger Ford was formed to standardize our control algorithms, addressing difficulties I had identified and solved. Greg formulated the standard in simple algorithms, simulating the pneumatic controllers we grew up on.

EXACT™, Finally!

All this time, academic work, notably by Karl Åström, of Lund University Sweden, and Pierre Belanger and his student Guy Dumont[22], was heating up. Finally in the '80's Leeds and Northrup, followed later by others, came out with an academic design. I was allowed to go ahead with another prototype, in an Intel 8086 microprocessor, and under the background support of Chuck McKay.

After an initial research implementation, I interacted with development engineers, Dave Richardson and Al Flanigan, who had never worked with computers before. But unlike the computer people, as electronic guys, they understood products. As work continued, with computer bugs and my being away giving those papers, they were beginning to lose patience. Bill Cunningham was helping; he described the program bug problems as continuing to run into "magic numbers".

The project might have been cancelled but I took my Apple II on vacation with a Z80 card (a Microsoft product back when they were a good little company). This card made the Apple II a superior Intel X86 microprocessor development tool to tools at Foxboro. With this I could set up simulated experiments on the working prototype code, which then ran automatically while I biked, ran, and swam.

---

[21] The EAI680 included a Joss based setup language, whose source code became the basis of a lot of my experimentation. Nick Rous, curious over my focus on this during the purchase process, finally concluded: "You feel that if they have this right they have the rest in good shape as well.". Their digital computer had an elaborate indirect addressing capability. We saw that it was possible to set up an infinite addressing loop, and, of course, we had to try it out. There was a large cheer when the machine stopped dead in its tracks.

[22] A continuing worker in the field, now at the University of British Columbia.

The results showed what the controller was actually doing under different conditions.

When I came back with these results, I had the means of proving my point. One of the development engineers, Dave Richardson, expanded this work. ---- And then a whole new team took over including Tom Kraus. He invented his own variant of all of this, leading to the EXACT™ controller. While I regretted the loss of some earlier features, there was clearly a net gain; the result, in my mind, put us an " EXACT™ generation" ahead.

Tom K. and Tom Myron took prototype EXACT™s into the field and achieved spectacular results. In England some of our marketeers got units out to prove them, including one turned over to David Clarke at Oxford, a well known expert, who further proved it.[23]

A well known MIT expert had been involved in an Air Force competitive test of academic designs, and had concluded the problem was impossible; I was able to quote our contrary experience. Tom K. and I went to an academic conference and showed off. [He tried unsuccessfully to set up a shoot out with ASEA (later ASEA Brown Bovari).] As a late starter EXACT™ out-performed everyone.

Afterward, the award chairman of the IEEE group awarding my work asked how my design compared with the academic ones. I told him I had a number of papers showing that the academic designs would fail in practice, except by accident! He said all the field people had said that's what happened!

EXACT™ came out at a time when Expert Systems (systems designed to mimic the behavior of working experts rather than academic theories) were the thing. And EXACT™ qualified as an Expert System. This helped its triumph; our competition then used a related, less ambitious Åström proposal.

A few academics tried to do control according to Expert Systems. But unlike their efforts, EXACT™ was based on an underlying systematic design process, the Expert System character being a computational convenience. Their view seemed to be that because Expert Systems were magic you didn't try to pursue a systematic design. This was regrettable because I would have liked to achieve academic support to continue this line of design, as with the RGA.

More recently I have tried to achieve the same technical success from a more theoretical direction. An adequate theory is difficult and contrary both the perspectives of my Foxboro colleagues and academics. I am too old to follow up my related papers! Maybe.

After EXACT™ came out as a Spectrum "box" and then an IA algorithm, I wanted us to be learning as a company, collectively evolving it. I followed up with several proposed extensions of my own, including more powerful nonlinear algorithms. A Moment Projection multivariable extension, designed with Pete Hansen, was legitimately mathematical as well as an ideal robust fit with EXACT™.[24] It was developed and commercialized in IA by Pete, as part of his own EXACT™ extensions.

His efforts allowed me to resume other work. Unfortunately the company no longer had sales/application interest and the customer was not promoted to continue to recog-

---

[23] I later found that Clarke was already very familiar with much of my controller design work. I find that he is now Director of something called Invensys University Technology Centre (UTC) for Advanced Instrumentation, although I may have seen the start of this.

[24] I did this by combining two methods (Moments and Projection) that control academics found unfavorable!

nize this technology. Any number of times since I have been told of customer surprise at finding that we had this unique capability.

Language Thinking

Paralleling this back through the late '60's I participated in discussions of all of the computer product designs. After attending a seminar by Doug Ross' MIT AED (Automated Engineering Design) group (later Softech) I came to appreciate the value of effective specialized software languages. I put together a block diagram language prototype in two weeks, demonstrating it to the major managers (Bob Silva brought Earl Pitt to see.). This precipitated my little/big block battle. Shortly after, the new Modicon Logic controllers did the same for relay circuits.

Throughout the early '70's (under Earl Pitt, and then Colin Baxter and Chuck McKay), and following the Modicon product, requests for different other (batch, logic) control forms began to come up. Founding participation in the Ted Williams' Purdue Language Workshop let me see different user perspectives. Eventually I co-chaired the TC-3 (effectively Batch) Language subcommittee, which led to the later ISA S88 Standards committee work.

Over time, I came to understand the validity of the different control attitudes, sometimes even when redundant. For example, Carroll Ryskamp had arranged the startup of a distillation tower by arranging controls and constraints so that whatever the state of the tower, it automatically headed in the right direction. A batch man might solve the same problem by sequencing the desired startup. Carroll's approach is entirely general.

The batch approach is cheaper and simpler unless it has to be modified to address problems with the sequence going off track. Either might be best, depending on the overall problem. A Batch plant controlled from Carroll's perspective would be more general but overwhelmingly complex in its own way.

So I began to frame appropriate diagrammatic languages for different aspects of process control.[25] I gave a paper in '75 at a Purdue Workshop related, Boston IFAC workshop where I showed a sequencing diagram which we had prototyped as an active controller in preference to relay diagrams. A well known French engineer came up enthusiastically, saying that they were thinking the same way; I had anticipated the Sequential Function Chart in a more general way![26]

An early '70's research effort was started to define and prototype a new digital control based product system, under Manny Freitas. We used the block diagram language concept with a separate operator interface building language I had invented (There are papers on this also.).[27] This never saw the light of day because of competition with Foxboro's most recent analog product success. I could see the handwriting on the wall and was highly disagreeable with all concerned, particularly my boss, John Bernard.

Development was working on a distinctive computer hardware design now, but for more limited purposes. The design was based on a Unibus structure (DEC's later term for the

---

[25] From the beginning all of my language designs strove for the greatest flexibility and generality possible within their structure.

[26] http://homepage.mac.com/ebristol/PDF/IFACRT.pdf

[27] "A Design Study of a Simple Conversation Language System", Automatic Control Division of the American Society of Mechanical Engineers Winter Annual Meeting, New York, December 5, 1976.

same technology) and bit sliced chip architecture, under Chandra Patel. I worked with that machine to try to carry over the operator language, without commercial success.

I used the machine as a product designer (using all of its features). The more conservative assigned programmers avoided possible hardware bugs, by restricting their usages. Thus I initially contributed more to debugging the hardware! This machine became the basis of our Fox-3 design and of the later control and operator console products.

The chief programmer of this machine was Kevin Diggins, who had been responsible for a highly disciplined machine language approach to batch control applications in earlier "Fox Boxes". He provide me an example of the difference between product and application orientation, when he did his own implementation of my operator language. My design had used sophisticated table oriented compiler techniques, to be more efficient and disciplined.

His program was a long sequence of machine language <If this, do that>s; an innocent's perception of a compiler, and a compiler writer's nightmare. But at the level given it was simple and worked without too much inefficiency. But it would require a programmer to use, not my intended user, and it wasn't oriented to disciplined later product evolution.

During this period John Bernard allowed a language education with two consultants [Jorge Rodrigues (MIT and later Softech) and Carlos Christensen] while I was exploring my language thoughts in an EOL (Experimental Organized Language or Ed's Own Language). I implemented pieces including BEOL (variously Big Ed's Own Language or Beginner's EOL), in FORTRAN to make the ideas accessible.

Binary search made my program run faster than another machine language implemented personal language of one of our programmers; an embarrassment he fixed over night! As part of all this I independently invented Objects (under that same name) for the flexibility it provided to represent control concepts.[28]

And Honeywell came out with their TDC 2000, a μ-processor counterpart to the research block diagram effort, taking over market lead. [EXACT™ coming out at this time

[28] Actually Objects date back into the '60's. I've always been fascinated by the many independent Object inventors also using the same name. Although Foxboro people insisted on saying that Blocks were the same thing. But then Ivar Jacobson once told me that his first Swedish Telephone Objects were called Blocks. Must be one of those academy/industry things.

might have minimized this problem!] We then followed up slowly with our bit slice machine counterpart, and then with a patched together combination of all our control products which we called Spectrum.

ICL

Meanwhile, because I was livid and thus radioactive, they set up a year's ('76-'77) sabbatical teaching at LSU. I taught the normal mathematical control course,[29] recognizing its difficulties in addressing practice, only later resolving how to do better; with Idioms! The teaching experience, together with later participation on a Carnegie Mellon thesis committee,[30] and initial interactions with Greg when I returned, affected my language thinking (and a lot else).

I now wanted a more integrated control language focused on expressing designs in terms of self-documented intent for the support of the process, rather than merely their detailed implementation. In this case, self-documenting meant that the engineer could enter the controls but computer would have total control over the final documentation, "pretty-printing" it in a completely standardized form.

In the years 1978-2000, we programmed a number of prototypes of different aspects of the now named ICL (Idiomatic[31] Control Language). Some of these are included on my website above. Some dated back to my Apple II, others to more recent Mac/PC implementations.

A notable effort was carried out by Naveen Kapoor, a PhD hire and student of Tom McAvoy's, on an IBM AT (PC). He persuaded me to approve use of an early GUI (Graphical User Interface) builder of the sort now common to Mac/Windows programming.

As I was moving from separate specialized graphic languages to an integrated language, true to form, the programmable logic language community was moving to standardize separate specialized graphic languages.

And, true to form, over the following years at Foxboro, I frequently had to argue with people who were now worshipping at this new, (inferior, old hat to the field) now long discarded by me, religion. But when I came back from LSU I was placed as a manager above a group which was now researching to start what became IA.

True to form I got in arguments with these people over the project objectives; we needed many things that they wanted but things that I wanted also. And for me the new system was the raw materials on which a Process Control language based product would be built. For them it would be the product. Management agreed with me but we both lost in the end. And gravity carried IA on thereafter.

---

[29] Math biased engineering teaching comes not just from practically inexperienced teachers, but because it is often the only known, repeatable, timely, and efficient way of making the point to experimentally inexperienced students.

[30] For Gary Powers (not the spy; previously of MIT, who spent a summer at Foxboro) with his student Rakesh Govind. Gary had written an early Chemical Engineering AI thesis at the University of Wisconsin; Rakesh's thesis was on automatic control design, which I rethought as a design language approach.

[31] Named for my invented and implemented Idiom concept which represented a strategy (a kind of specialized Pattern) for automatically combining standardized control elements according to general rules to achieve a specific generally recognized, higher level control objective, as practiced by people like Greg. Idioms expanded into linked block-like structures detailed to fit the local control environment, initially, user editable for added flexibility. They automatically included all the bumpless transfer, back calculation function normal to control. One of my Asilimar academics, and earliest academic RGA advocate, Lowell Koppel, at Purdue, author of the once main Chem. E. control text, was setting out to teach his own notion of Idioms before leaving teaching, using a large IBM system!

Not a manager, and unwilling to continue in this direction, I set out to further ICL and the group leader, Dick Caro, left. Two other leaders (Mal Beaverstock, and then Charlie Cooper) took over in succession, until finally Rick Kornblum was rehired. Rick had the required qualities to manage a nearly impossible project and established many of the basic properties needed (and in the face of rapid computer technology change). But without interest in a more process control specialized product.

While at LSU, in my first semester they let me teach a special topics course on conversational language design, based on a course note book I had written in my previous last months at Foxboro. Later, under Charlie Cooper's pre-IA leadership, with Bill Robinson, an ex-math prof, I taught a course on Objects, this time using PASCAL. Maybe this initiated the Object-based, IA control blocks structure.

[Rick was a pain but no-one else could have succeeded with IA. He deserved better than we (including himself) later gave him. Eventually IA became crucial to all our other objectives, even if my goals were not fully achieved. Ironically, although Rick was never in tune with the kind of language based product I wanted, he nevertheless had been responsible for previously managing the BASIC control language based Fox-3 effort (before leaving in his first huff; Rick was from Dartmouth where BASIC originated).

This was a neat flexible-language, single-application, computer system, like ICL, lacking ICL's focus on clear self-documentation of different Process Control perspectives, and application intent. It was great for the application designer to work with, but the application was hard for anyone else to understand.]

But while we were developing it, the business difficulties, following from our loss of market share, caused us to move many unused application engineers to the (still not so named) IA effort. This meant that we had few real product design oriented engineers. The project moved forward with many difficulties and embarrassments. We needed something to remind the customer of our underlying leadership. EXACT™, late started for self-tuning but timely for IA, provided this, while delaying the language work.

In the Spectrum period we began to support human interfacing research work seriously. This was always important to me, even though I had no conventional competence. The language work emphasized clearly stated control intent. In time it became clear the digital human interfacing could be expressed automatically from stated intents just as well as controls.

So one more function was incorporated into the ICL language thinking. As indicated below, none of this came into being. I gave a related paper, intended to support a human interface session at an ACC conference; they made it the main defining paper instead.[32]

Swinging Door Trend Compression

While the language work was going on, a DuPont engineer was pushing improved digital "trend records". One of the operating tools in the plant had always been circular or long strip paper charts, recording plant history. The early computer memory limits limited the amount of history that could be stored for display. Modern TVs and iPods use data compression to record only the important stuff so much more music and movie can be put in a disk, box, or channel. Dupont wanted this.

---

[32] http://homepage.mac.com/ebristol/PDF/dh.pdf

Our human interface expert, Dick Shirley, came back from DuPont asking for this capability. But the compression that DuPont invented was weak, and you never base your product on someone else's lousy idea. So I set out to get a better design. The first design was called Triangle Trending, constructing trends out of small triangles. The result was much better than DuPont's.

But Dick couldn't see it if it wasn't like DuPont's or proved (exactly, e.g. by mathematics). The second design was called Swinging Door Trending, building trends out of sequences of optimum straight lines. This was maybe a little better still, but it was proven. And the DuPont people got a kick out of their colleagues embarrassment. But we didn't get it into the product line until long after I retired with the patents worn out [and after competitors had stolen it (from a paper I wrote)]. While memory is cheaper now, compression will be valuable to speed search and expand meaningful trending.

ICL SuperVariable

Actually, ICL work continued after Foxboro was sold, with a group of Bristol fellows first preserved, in the Cocasset building, under Ernoe Czaky. ICL had always been divided into specialized Pages to represent different aspects of control, and Operations hierarchically divided to represent different parts of the control function. An important aspect covered in a Definitions Page, was the process I/O (connections to sensors and valves). Traditionally these were represented in the usual inflexible blocks.

The Definitions Page replaced the blocks with operationally displayable tables representing different groups of process variables. Table headings could be defined by the user, so he could flexibly control the display, grouping, and parameterization of the variables, allowing not only parameter but multiple name (synonym) Attributes.

Earlier, the issue of "smart sensors" had come up. And the marketeers had proposed several other simple control logic, cut-cam follower digital replacement boxes. In my view, these tables provided the best basis for configuring smart sensors and field devices within a single language framework.

And then I asked one of the marketeers what the natural limits to computational capability should be for "smart sensors". Well, like the usual sales guy he saw no need for any.

And so I came up with the ICL Footnote, which allowed arbitrary computations to be inserted within the tables, without disturbing the basic table clarity. For field devices, Idiom Footnotes were supported with their set of tuning Attributes. Also supported were conditional execution and sequence control of standard statements or special sequencing statements called Theme Statements.

Variables and their name and parameter Attributes were accessed in a natural notation. The result was the SuperVariable and SuperVariable ICL, a single, integrate-able, language family "box" instead of a collection of random "boxes".

The SuperVariable took the form of a single string of user defined Attributes (parameters), combining header and value (i.e a single line table), with arbitrary Footnote insertions. But the string could contain the multiple Name Attributes, or sections keyed by Name Attributes representing distinct sensors/valves. The "pretty-printing" of the SuperVariable would recognize these sections in separate tables,

which would have multiple line entries if there were multiple identically headed Attribute groupings.[33]

The resulting concept envisioned the main ICL language system, with little SuperVariable ICL sensor or Field Device systems operating under it, feeding it data or accepting control data. The implementation made up one of three+ ICL patents. The method operated scanning the single SuperVariable, every sampling instant, Attribute/Footnote by Attribute/Footnote. It included a stack-like structure to reverse-order, back-calculate any Idiom function needing this. The real time sequenced statements were also supported within the scan.

Fresh from writing the patent, I programmed a Microsoft Graphic BASIC demonstration in a couple of nights at a San Padre Island academic conference to show colleagues Idioms in action. This still works under Windows™, with free form configuration of the variables and controls![34]

In the early stages of the SPC50 (Field Bus) standards effort Paul Griem worked with me to talk a well known customer representative into including the SuperVariable in the otherwise grossly inflexible effort. However more limited strategies of competitors blocked this. We will never get innovation out of even novel standards.

Shortly before Foxboro was sold, still in research, I used the Fellows funding I controlled to get a NeXT computer, with its powerful Graphical User Interface capability. This would allow me to work with UNIX (which IA was using), Adobe PostScript™ (to give me a high quality standard text/graphic for ICL), and support the new design group the company had set up (which was disbanded by Siebe).

There was one oblique benefit. Foxboro was working to set up a joint effort with Sun Microsystem for IA. And whenever they came in, my office and NeXT, were right next to theirs. No-one said anything, but the falsely implied competition must have helped.

When we were sold, and the Fellows group formed and moved under Ernie, I had more powerful access to our network than anyone else, allowing me to use the NeXT to develop requested IA software (not then used).

Sometime after, I implemented a demonstration of the SuperVariable ICL execution and "pretty-printing" under my then newest boss, Dick Thibeault.[35] I wanted to show how a mixed collection of variables and Footnotes could be run and formatted automatically: I could vary the font size dynamically with a slider, changing the available line size.

This showed the automatic accommodation of splitting tables into parts, dynamically as the available space changed with font size. Also by altering the headings for different parameters, I could show groups of distinct sensor definitions change as the heading definitions changed to be consistent or inconsistent with each other.

The program addressed: two temperature variables and their averaged value, one flow variable, one valve, two temperature parameters, and four time parameters. It also included: one cascade control loop, controlling one temperature through the flow and valve.

---

[33] A long term Chuck McKay consultant, Robert Solomon, compared the result to the genetic DNA strategy.

[34] http://homepage.mac.com/ebristol/SILDS.html

[35] http://homepage.mac.com/ebristol/PDF/slifol.pdf (ISA 2000: "A Field Device Language in a Process Control Language Family")

A Theme Statement implemented a temperature profile, separate for the two temperatures, the one under control.

The profile included consecutive ramps and holds, carrying the temperatures up, holding, and then down. The Theme Statement included its own Footnotes which allowed one variable to track the other, in one case following a quadratic curve. When the program was run on a simulated process a plotting displayed the two time records showing the control ringing.

Execution time was faster than any then existing competition, with greater flexibility (I don't remember the execution time; the reference paper doesn't seem to quote one; I can't understand why I didn't include it.) easily run in any small μ-processor. The language is so simple that I have no difficulty reading the program after all these years!

Dick left the company shortly thereafter. I was told that one of the reasons he gave was the new management's inability to respond to innovations like those represented in the SuperVariable and this demonstration.

Our group moved to the Neponset building under Tom Kinney and Gene Yon. Gene occasionally spoke for ICL, but my colleagues lack of positive response discouraged me. The SuperVariable ICL would have been easy to create even in the new company.

In retrospect I should have kept this focus, not confusing it with the larger ICL; it was still a better sell and starting point. But I really didn't know how to approach the new management. I had been impressed by George Sarney, as a new president, and talked with him, but found I didn't have a strategy to introduce him to the refinements of Process Control behind ICL.

Again, SuperVariable ICL would have been easier and more direct. Greg had had a similar connection failure before leaving. But as an application innovator he didn't need a company; as a product innovator, I did.

Shortly before retirement I also put together a program which demonstrated the full text/graphic pretty-printing of complex Idiom controls. This also still works, but generates PostScript™ output which would now better be changed to Acrobat™. Related newer ICL documenting demonstration programs now generate Acrobat™ instead.

My Unrequited Loves

From '59, attending a large meeting where Vin Tivy and Dick Sonnenfeldt held sway, until the sale of Foxboro in 1990, I was mixed in most control related aspects of digital and systems related development. Shortly before his death, following EXACT™'s completion, my father wrote a letter to Rex saying he had wished he had supported my early work; we had had long arguments on the direction of research. And yet what I ended up doing was trying to express the kind of product he grew up with in a new environment that was hard for him to grasp.

Rex decided to destroy the letter with the view of not disturbing people (Foxboro now widely recognized my reputation.). And yet we still needed to be disturbed. A lot of things I still consider important were never completed. Of these, my three unrequited loves are:

- A proper mathematical analysis of the PID controller; the very, old hat, simple, generalized (in different forms dates back to the Watt steam engine controls) controller. The PID has no adequate mathematical analysis (neither the practitioners nor the academics believe this).

  Such an analysis would allow a more powerful EXACT™ and one which could

be successfully evolved without the special creativity required by the original.

- ICL commercialized, allowing the full automatic generation of controls, their human interfacing, complete computational flexibility, and a full standardized "pretty-printed" text and graphic documentation. [Engineers are lousy at documentation on their own.]

  Had Foxboro stayed Foxboro a couple of years longer, I would have had the authority to carry it out. Among the difficulties continues to be lack of a standardized text/graphics vehicle, like ASCII text code, to support high quality dynamically created documentation.

  The specialized PostScript™/Acrobat™ expertise I used would be hard to find in standard language builders, let alone among process control people. But in the earlier digital period we successfully hired talent or consultants on a number of high quality language efforts.

  And a product-ized SuperVariable ICL, as prototyped under Dick Thibeault, would have been extremely easy to implement and sell. Even now it would be far superior to any competition.

  All close colleagues know of my deep ICL concerns whether or not they believe in it. At Foxboro's 100th year celebration where Greg, I, and others were invited to a party, I was belaboring the then President.

  When I came to my "proposed solution for all Foxboro's problems", I waved my hand/baton at the others. With one voice everyone shouted "ICL". O well, at least I believe in this as the kind of unifying solution necessary to the industry.
- Multivariable control, clear to users, broadened and simplified; the notion of controlling large processes with many interacting parts; the issue that the RGA begins to address, and of later theoretical extensions. Some of this work suggested that the underlying mathematics was based on a grand tautology that extended well beyond the linear theory usually addressed.[36]

I apologize in this discussion for not distinguishing the different subspecialties of process control. Thus the PID and multivariable control address different dimensions of Continuous Control (like control with a thermostat). Whereas ICL originated with the continuous Greg-style Idioms, but is intended to support sequencing and logic in an equally transparent manner.

The note, as a whole, mentions many people in context. This doesn't begin to cover everyone in Foxboro, industry, and academia that I interacted with usefully. All have contributed, for better or worse.

EHB

3/10/09

---

[36] http://homepage.mac.com/ebristol/PDF/RHPllEd.pdf