Putting Knowledge-Based Concepts to Work for Mechanical Design

Ian C. Campbell, Kris J. Luczynski, and Steve K. Hood

CAMES (computer-aided mechanical expert system) is a tool for automatically designing material-handling equipment used in pulp and paper mills. Operational since 1988, it has designed over 700 machines in a total of 5 classes and, consequently, has influenced our principles for deploying all knowledge-based systems. Because it is being extended to other machines within our group of companies, it is becoming more closely integrated with other knowledge-based and conventional systems.

Principles for Deployment of Knowledge-Based Systems

The deployment of knowledge-based systems in our corporation follows some principles that are still evolving; the development of these principles is more important than the development of any individual system. The principles are that (1) we will develop knowledge-based applications that are directly in line with our corporate goals, strategies, and critical success factors; (2) knowledge-based applications must be introduced into our corporation slowly but surely, so that they neither are rejected by our organization nor destroy valuable aspects of our culture; (3) the corporate computing environment must encourage knowledge-based development; and (4) knowledge-based application must be under the direct control of the line operating manager who will most greatly benefit.

Prior History and the Current Environment

This chapter continues the case history of several expert systems, all intended to improve our company's ability to give well-rounded responses to customer needs by improving mechanical and electrical engineering specifications, engineering productivity, design quality, and proposal preparation. Our ultimate knowledge-based–engineering goal is to use less human effort to make our corporation more responsive to customer needs, thereby ensuring a more profitable corporation. Previous work on a programmable logic controller (PLC) system (CALES) is described in Campbell and Bukshteyn (1990). CAPES (computer-aided proposal expert system) is a proposal-preparation system that is dependent on the CALES and CAMES systems (so we know the proposed system is preengineered and, therefore, can accurately be costed and priced). An early version of CAPES is partially deployed. These knowledge-based systems are in direct alignment with our first principle and our corporate goals, strategies, and critical success factors (which are not included in this chapter for space considerations).

All these knowledge-based systems are being moved into an environment with a parallel Common Lisp object system (CLOS) on the corporate computer. This computer is a four-processor Sequent S27, which is the corporate fourth-generation language platform, running Informix. Integration between CLOS and Informix is by standard query language (SQL) calls. To implement the Information Technology Department's (ITD) first strategy—to contribute to "cross-functional and geographic integration"—CAMES and CALES are being ported to the corporate computer system, a Sequent S27 with 5 Sequent S3 computers operating an enterprisewide 56 kilobit/second networking environment. This architecture, supporting standards that include the X Window system, UNIX, TCP/IP, CLOS, and SQL, will help solve many integration problems between sales, engineering, business, and knowledge-based systems and is part of the implementation of ITD's basic mission. This hardware design partially implements our third principle, that the corporate computing environment must encourage knowledge-based development.

The Original Problem Defined

In 1986, some apparently intractable business problems were starting to hurt mechanical design sections of the company. Our company is located in a small western town, and skilled design engineers for pulp or paper machinery were hard to find. A continued corporate strategy of selling highly customized solutions to large, complex problems in the pulp and paper industry helped with our sales success but created internal problems and the need to improve engineering productivity and quality. These problems seemed intractable because inexperienced engineers, like all professionals, tend to make errors, and we simply could not afford to retain a full complement of human expertise from one business cycle to another, when the whole relearning cycle would start again. Therefore, we saw a need and an opportunity to transfer engineering knowledge between business cycles by using a knowledge-based system. We hoped that CAMES could make an impact on mechanical engineering productivity during the current business cycle (1985–1991). However, even if it did not, we thought CAMES would at least pay for its own development. Ultimately, our company would be able to handle more work with the same number of engineers, or if we were forced to lay off engineers, and business subsequently improved, we would not have to hire as many inexperienced staff members during the next business cycle.

Description of the AI Technology Used in the Application

This section describes the technology used in the CAMES system. We examine our decision to use Lisp and our initial problems with object-oriented Lisp, the design concept and the selection of the first machine to design, and initial problems.

The Decision to Use Lisp

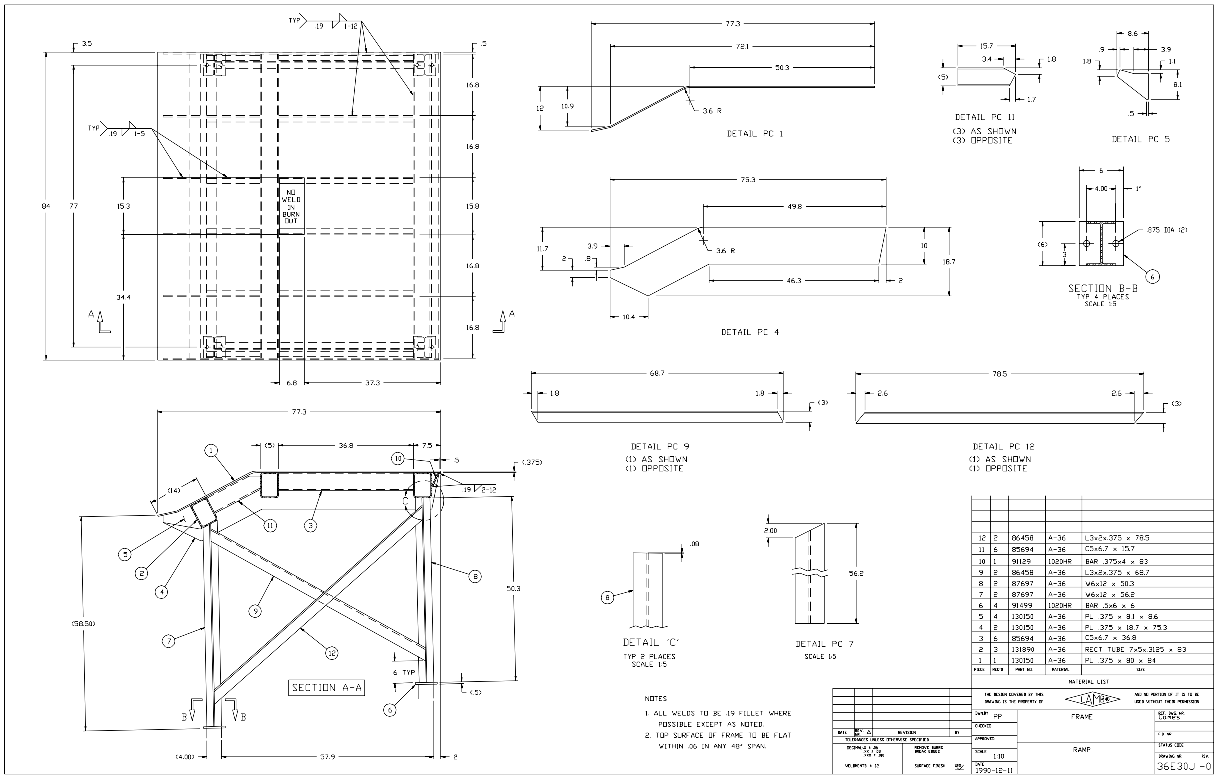
One lesson learned from our CALES experience was that the Lisp environment contains a powerful development tool. We were prepared to attack problems in a rapid prototyping way without any clear plan of how we would handle the later parts of a large problem. We believed that if, as electrical or mechanical engineers, we could handle the engineering problems, then as Lisp programmers, we would be able to encode our mental processes after reasonable work. We knew we could not or would not explain our expertise to computer programmers, but we could and would explain it to a computer. This method avoided the serious problems of an uncooperative prima donna or an arrogant expert, the expert who is too busy, the expert who is inconsistent and has to be told so, or conflicting experts getting into a fight. We were all these things, so we just talked to ourselves a lot.

Our answer to the problem of selecting a development language was partly new and partly old: object-oriented programming using Lisp. Implementing this decision was slow but steady and illustrates our second principle, that important knowledge-based ideas be introduced only as quickly as the culture can accept them. We considered and rejected the use of rule-based systems after learning that serious software users were reporting problems of excessive complexity when the number of rules exceeded 1,000 or so. Conventional programming languages were too inflexible. Fearing that new languages would contain hidden problems that we would not be able to resolve, we rededicated ourselves to engineering solutions, not computer problems.

Initial Problems with Object-Oriented Lisp

In 1987, object-oriented systems were supposed to be available, but proven ones were so expensive that they were inaccessible to us. Our method of funding expert system work was by the *skunk-works method*. We think of a skunk-works project as one that is short of formally budgeted money but is allowed freedom from the normal checks and balances of bureaucratic organizations. The enormous benefit of this funding method was that although we could not plan or schedule progress because of the newness of the application problems and the relative strangeness of the technology being used, we did not have to. The price of this freedom was that we could not afford to buy a proven object-oriented Lisp.

We tried to use Object Lisp on an early version of Golden Common Lisp, but Object Lisp was not going to be supported. CLOS seemed too futuristic at this time, and GoldWorks was not available. Thus, we did the best we could and simply started programming in Common Lisp, using Golden Common Lisp. Because of the high costs for multiple copies of Common Lisp and the (to us at this time) specialized hardware, we decided to write a data-entry and initial calculation program in Pascal. This approach allowed multiple users to specify data on their own personal computers and then move this information onto one Lisp system. The ramps (our first) program, eventually grew to over 15,000 lines of Lisp code; the code was purely functional and did not include state-of-the-art ideas, such as case-based reasoning or constraint propagation, and was not object oriented. Figures 1 and 2 show drawings produced by the first ramps program. The reason for this brute-force approach to capturing ramp knowledge was that we were unable to develop a deep sense of comfort with any AI expert in any of these emerging research areas; any move to case-based reasoning, for example, seemed to require that our engineering staff turn over project control to an expensive AI expert who did not deeply understand our engineers or our corporation. For human and business reasons, this



NOTES
1. ALL WELDS TO BE .19 FILLET WHERE POSSIBLE EXCEPT AS NOTED.
2. TOP SURFACE OF FRAME TO BE FLAT WITHIN .06 IN ANY 48" SPAN.

PIECE	REQ'D	PART NO.	MATERIAL	SIZE
12	2	86458	A-36	L3x2x.375 x 78.5
11	6	85694	A-36	C5x6.7 x 15.7
10	1	91129	1020HR	BAR .375x4 x 83
9	2	86458	A-36	L3x2x.375 x 68.7
8	2	87697	A-36	W6x12 x 50.3
7	2	87697	A-36	W6x12 x 56.2
6	4	91499	1020HR	BAR .5x6 x 6
5	4	130150	A-36	PL .375 x 8.1 x 8.6
4	2	130150	A-36	PL .375 x 18.7 x 75.3
3	6	85694	A-36	C5x6.7 x 36.8
2	3	131890	A-36	RECT TUBE 7x5x.3125 x 83
1	1	130150	A-36	PL .375 x 80 x 84

MATERIAL LIST

THE DESIGN COVERED BY THIS DRAWING IS THE PROPERTY OF LAMB® AND NO PORTION OF IT IS TO BE USED WITHOUT THEIR PERMISSION

DATE	REV. NR. △	REVISION	BY
TOLERANCES UNLESS OTHERWISE SPECIFIED			
DECIMAL: .X ± .06 .XX ± .03 .XXX ± .010	REMOVE BURRS BREAK EDGES		
WELDMENTS: ± .12	SURFACE FINISH 125		

DWN.BY PP	FRAME	REF. DWG. NR. Cames
CHECKED		
APPROVED		F.D. NR.
SCALE 1:10	RAMP	STATUS CODE
DATE 1990-12-11		DRAWING NR. REV. 36E30J -0

Figure 1. Ramp Drawings: Weldment.

loss of project control was unacceptable; we preferred a policy of slow, steady, continuous improvement over an expansive, risky experiment that used the best available technology. We knew we did not start with state-of-the art technology in either hardware or software, but this fact did not prevent a slow, steady growth that eventually resulted in implementing our programs in parallel CLOS, X Windows, and so on. Our corporate culture could never have progressed to this point without moving through years of lower evolution. This implementation illustrates principle 2, that knowledge-based applications should be introduced slowly.

First Phase: Design Concept

The first phase of this project was to agree on the design concept. This concept was that any reasonably intelligent technical person should be able to sit at a personal computer and answer specification questions about the machine to be designed; a specific instance of this machine should then be created. For example, if a person wanted a conveyor designed with certain economic, reliability, speed, and weight specifications, we would expect output consisting of complete mechanical weldment, assembly, and installation drawings.

In our situation, we knew that our equipment consisted of about 50 major classes of mechanical equipment connected by a custom-programmed PLC. Some of these pieces of equipment literally had an infinite number of possible instances, and it was seldom that two instances were designed to be identical. Other machine designs differed in only a small number of degrees of freedom, but all system designs were fundamentally different.

The first important decision dealt with changing the CAD system used for design. In 1984, our company standardized its use of a CAD program, selecting Anvil 1000. This program did not have a published interface and could not easily be driven from another program. Even if it was to be driven, the manual specifically warned against doing it because this interface might change in a later program release. In contrast, Autocad had a standard interface and even had an AutoLisp language built into it. AutoLisp was intended to allow users to do limited things, such as extend commands within Autocad, but was not capable of handling CAMES. We elected to use Autocad, starting a trend that might eliminate Anvil 1000 within our company.

Original Programming

The design program was written in Common Lisp, but the initial input program was written in Pascal so it could run on any 8088 or 286 com-

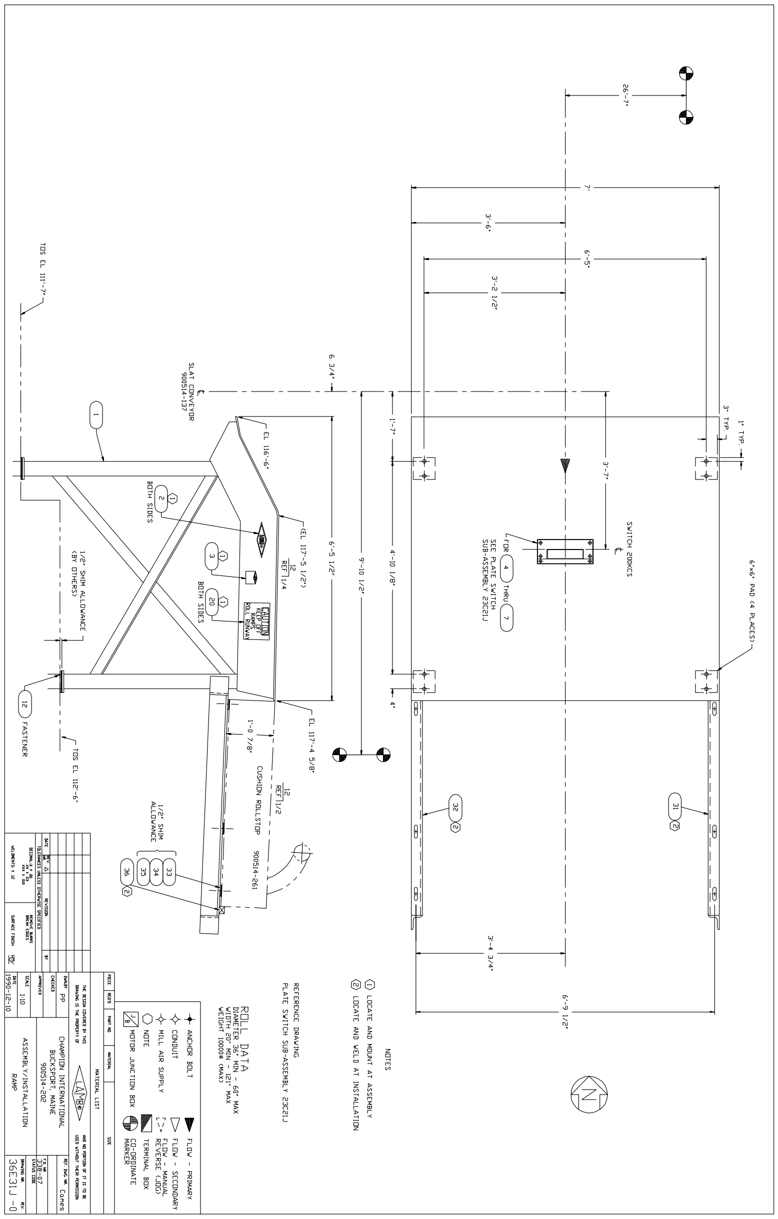


Figure 2. Ramp Drawings: Assembly and Installation.

puter (386s were not available at this time). The initial machine class, called ramps, was started in the fall of 1987. The first ramps were drawn within a few months and were relatively successful. However, to our surprise, we soon started to miss some production deadlines, creating serious problems. The main reason for this failure was that we had not mastered the breadth and depth of the options and complexities of our chosen machine. We elaborate on some of our failures in the next subsection.

Selecting the First Machine to Design

The fundamental criterion for selecting ramps as our first class of machine was economic. The class of ramps is the machine in our company with the greatest number of "same as but different" designs.

All experienced design engineers warned us not to attempt this generic design of ramps. The most skilled engineers claimed that although they looked easy, ramps were the most difficult machines to design because they were infinitely variable; any change in one specification would change almost all the design. Other, less experienced, or bystander engineers, believed that ramps were too easy to test the concept. These opinions seemed important, but what seemed more important was the hope of financial payback; we would handle the technical problems sooner or later, easy or hard. In addition, we believed that it was wise to start with the hardest problem first, so that each different instance is a simplification, not a complication. In retrospect, this approach was our first major error because the deep complexity of options nearly caused project engineers to lose confidence and commitment.

Initial Problems: Drafting Standards

The complexity of the task almost defeated us even before we started handling many variants of ramps. There appeared to be constant change in implied drafting standards. The different ways to write a number all have different meanings on engineering drawings; that is, 1, 1.0, 1.00, 1.000, 1.12, 1.125, and 1-⅛ all represent different concepts. The number 1.13 might not be in the allowable number set at a certain time. There are complex rules about rounding up or rounding down, some of which make little mathematical sense, but they are a part of our company culture. They could eventually be changed but not to accommodate a computer. As another simple example, the decision to record lengths in millimeters, decimeters, or meters or in inches or feet and inches was not primarily mathematical but the result of standards. The standards were not necessarily logical or stable. Humans could (at a cost) adapt to irrationally derived changes in standards, but our program could not. As a result, our business processes were gradually forced to become disciplined to handle CAMES. These changes were desirable in a manual system but were impossible to identify or implement before the use of CAMES. The process of changing business processes as a result of using CAMES is a valuable side benefit.

We started to realize, as we had with CALES, that we were dealing with a problem in translation. We were translating specifications into mechanical drawings, and we started thinking of a mechanical drawing not as a mathematical fact but as a description in a graphic language. The language has some standards, many dialects, and some major variants. These standards were policed by the department that checked the mechanical drawings, but the individuals in this department had differing views on standards and sometimes seemed to apply different standards to CAMES drawings than they did to human-generated CAD drawings or manual drawings. It took several critical months to work out satisfactory drafting standards for CAMES.

Initial Problems: Deeper Knowledge

This process of defining drafting standards for CAMES ironically was helped by the checking department after we started receiving drawings that were heavily *red lined*, that is, demanded many small corrections in red pencil. These small corrections were invariably proven to be correct in CAMES and incorrect when changed by the checking department.

CAMES now started to receive its first small measure of respect. No engineer or draftsperson normally had the time, inclination, or courage to challenge the checking department; we all just accepted its dictates because it was easier this way. However, with a knowledge-based tool, it was impossible to make superficial fixes. The design knowledge either had to be correct or had to be corrected. After some of the actions of the checking department were shown to be fallible, we started catching errors that would have cost between 10 and 100 times more money to fix in the field under the hostile gaze of a customer concerned about the late startup of his(her) billion dollar paper mill. As a reliable implementer of standards, CAMES became known as a useful tool. The greatest complaint became that "it takes too long and costs too much to transfer any machine into CAMES."

Details of the Operation of CAMES

The CAMES programs design the required machine and then automatically invoke Autocad to produce a drawing package for each machine processed. A manufacturing package consists of all the detailed weldment, assembly, and installation drawings and bills of materials required for a specific machine on a specific project. The system can produce drawings overnight using data entered during working hours or in a few minutes if necessary. In some cases, the Autocad drawing requires manual drafting corrections, but the average time for producing the ramp package was reduced by 50 percent or more. Figure 1 shows an example of a weldment drawing for a ramp. Figure 2 shows an example of an assembly and installation drawing for a ramp. These drawings were completely designed by CAMES and have not been touched by a human designer, although the task would be easy with the use of Autocad.

How CAMES Does What It Does

The ever-evolving design and frequent drafting standard changes consumed more and more programming time. Consequently, an attempt was made to structure the program in a similar way for each machine and rewrite the function library so it would approximate the Autocad drafting commands. This approach, we hoped, would allow users to make certain types of modifications themselves. In this concept, the drawing itself, with its three main views (plan, elevation, and front), is an object. The structure of the program for each drafting view is identical. Critical points (for example, the location of various elements, such as conduits, motor position, bearings) are associated with an object either by formulas or constants. Predrafted machine elements used as drafting blocks are assigned to the particular view by name and location relative to the view coordinates; so, the program always knows where to insert in relation to other components. The program uses a function library common for all the machines. The program maintains its own database of important machine components. If a new element, for example, electric motor, is specified, the program prompts the user for certain information about the new motor and its drafting representation (blocks); after recording this information in the database, the program is automatically ready to use it. A separate database is kept for important customer information, for example, bale dimensions and electric component specifications. The program also tracks the overall drawing database: If new requirements resemble previous designs, the user is alerted. The program uses an interface to Autocad and generates drawings using a set of standard or modified Autocad commands.

The Difficulty of the Design Problem

From the mechanical and drafting point of view, the machines completed to date in CAMES are not the most complicated in the machine line; however, their detailed design is subject to much change and is adapted to suit the other more complicated and standardized machines in the roll bale line. Almost all basic design criteria of the CAMES machines are subject to change, including main geometric dimension (length, width, height) and the position of motors, reducers, switches, and bearing types. These variabilities in the design propagate, to the drawing dimensions, changing patterns that must adhere to certain rules and be easy for the end user to read. For example, machine dimensions must be laid out differently for a motor on the left of a particular machine than for a machine on the right, greatly changing the structure of the dimension lines, not just the data for the dimensions. Problems are also caused when an additional machine is located on a CAMES machine. The design does not change, but the dimensioning must now accommodate both machines without "collisions." The rules governing the dimensioning have become complex and are one of the main reasons for an increase in programming time with design changes.

The Errors Found

The CAMES user answers between 30 and 80 questions pertaining to a particular machine. Most of the questions are simple, and the typing time should not exceed 10 to 15 minutes. However, for various organizational reasons, collecting the data before typing can consume a considerable amount of time. After data entry, the user usually reviews the CAMES-generated drawing in Autocad or plots it, marks the errors, and then corrects them in CAD. Most of the errors are caused by design and standard changes that usually keep ahead of programming. The usual area of error is line collision, for example, dimensions resulting from not updating the drafting blocks. There are few design or calculation errors.

Phase Two: Selecting the Second Machine

After our experiences with ramps, we wanted to design other classes of machines. The specifications for this second phase of our work were (1) the ability to handle interconnected systems of machines; (2) fast, correct manufacturing packages for each machine; (3) a friendly interface and database; (4) straightforward program design and structure; and (5) ease of modification when the design changes.

At this stage, CAMES was considered interesting but not proven. The next step was to automatically design whole systems of interconnected machines. However, the delays and disappointments caused by initial problems in ramp production had caused some hostility in the minds of the managers of our roll product line. Therefore, we decided to develop CAMES to handle whole systems of machines using our second most important product line, pulp baling. CAMES would be used with our most important line if it was successful on the second line. The class of pulp bale chain conveyors was selected as the next machine because it was the most frequently used machine in the pulp bale product line. This machine is designed in a wide variety of lengths, widths, strand spacings, and motor and reducer types. As with ramps, the basic layout of the program was completed in just a few months. Since this time, the program has constantly been modified with changing design standards and various requests for improvement. Figures 3 through 6 show designs produced at this stage of the project. There were signs in early 1991 that at least four to six product-line engineers wanted to learn CLOS to be able to extend CAMES in the future. It is psychologically important to them that they continue to control the basic knowledge, and some seem prepared to learn simple CLOS programs, if necessary, to retain their professional control of the product line.

Major Weaknesses in CAMES, Phase Two

A need for constant change and improvement leads us to the major weakness in the existing CAMES system: A Lisp engineer-programmer is required to modify the design engine, and the design engine is not extensible by the domain expert (a non-Lisp engineer). Much initial knowledge about the chain conveyor machine was collected from three relatively inexperienced engineers, who, in retrospect, did not have full knowledge of the whole design spectrum. With each new modification and addition, the design and program became more complex. Our experience (not unique) is that for each stage of the problem, it takes approximately 20 percent of the total programming time to complete 80 percent of the machine, but the remaining 20 percent of the work takes 80 percent of the total time. The last 20 percent of the work renders the project acceptable to the user and cannot be ignored. We also found that it is often economically correct to complete a machine with only about 95 percent of the options captured, leaving the remaining options to human correction in Autocad. This fact exists because the incremental investment in programming could not be recaptured for some infrequently used options. These issues

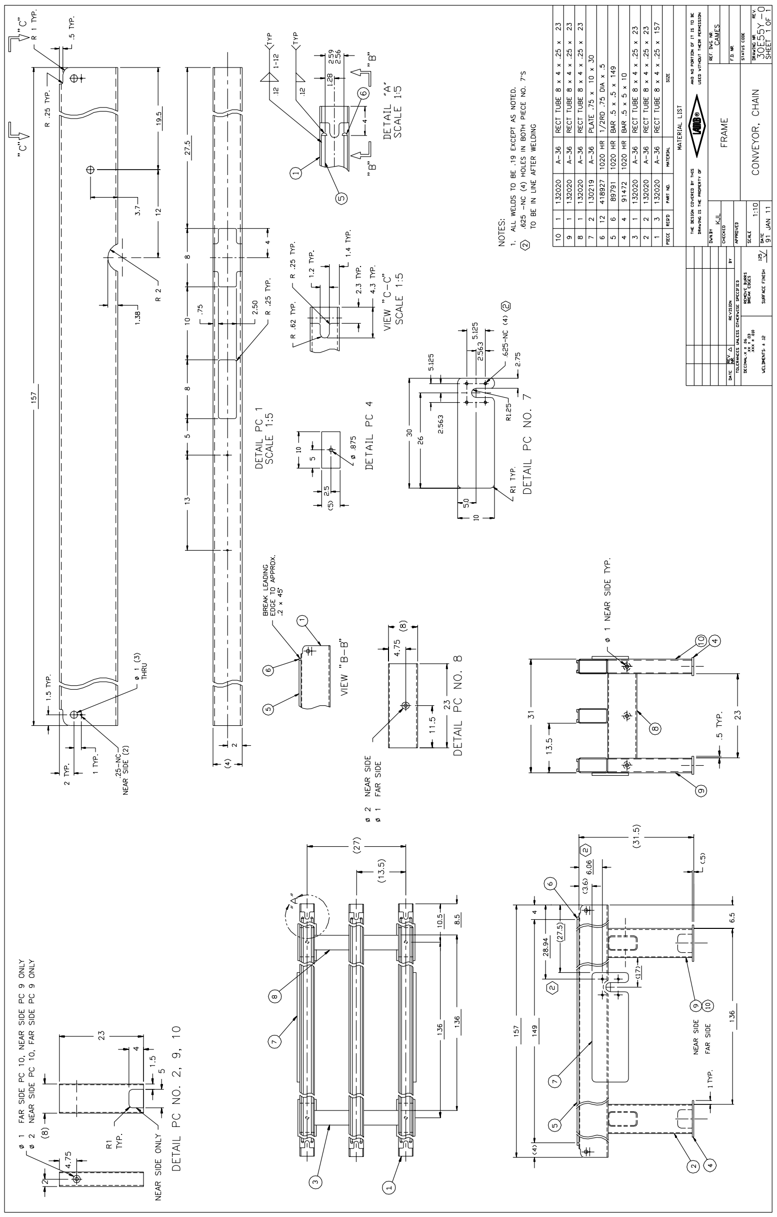


Figure 3. Chain-Conveyor Drawings: Weldment.

would be handled differently if CAMES were extensible by a domain expert and if these problems were in the hands of a project-oriented user and not a "staff stranger" with his(her) own time constraints.

CAMES creates and maintains its own dynamic database of various typical components, for example, motors, reducers, terminal boxes, and some customer data. In this way, we built in some extensibility, but this feature needs to be improved in quantity and quality, probably by using CLOS.

Moving toward Object-Oriented Programming and Extensibility

For this second phase of our work, we wanted to simplify the Lisp code and use a repeating programming structure for each new machine, moving in the direction of the objects and instances used in object-oriented programming. For example, libraries, variable names, and blocks were standardized for easy reusability, recognition, and maintenance. The system consists of three main units: data entry, design, and output. While designing a machine, the system searches its database for characteristics (for example, dimension or quantity) of various machine components. Drawings can be drawn to a specified scale with options for dimensions in metric, imperial, or both. Much of this second phase was done in muLisp. In retrospect, the main difficulty encountered in program development and maintenance is with knowledge acquisition and the frequent changes in design standards. Often, the knowledge that was used for design became obsolete once the programming was completed.

Details of Making the Drawings

Our first approach to making drawings was to represent a machine as a composition of a series of simple elements, build it in a three-dimensional space, and then project it onto three views on a two- dimensional drawing. It worked well for our first simple ramps; however, it became increasingly difficult to build more complicated ramp components or mechanisms with their intersecting lines and complicated shapes. It was also difficult to tell the computer which lines were important enough to be shown on the drawing and which were not. Human rules governing this information are often inexact. When we started even more complex machines, their projections in three dimensions became intricate for certain components. Therefore, we decided to use drafting blocks for each two- dimensional view. These blocks either were extracted from master drawings or were specially

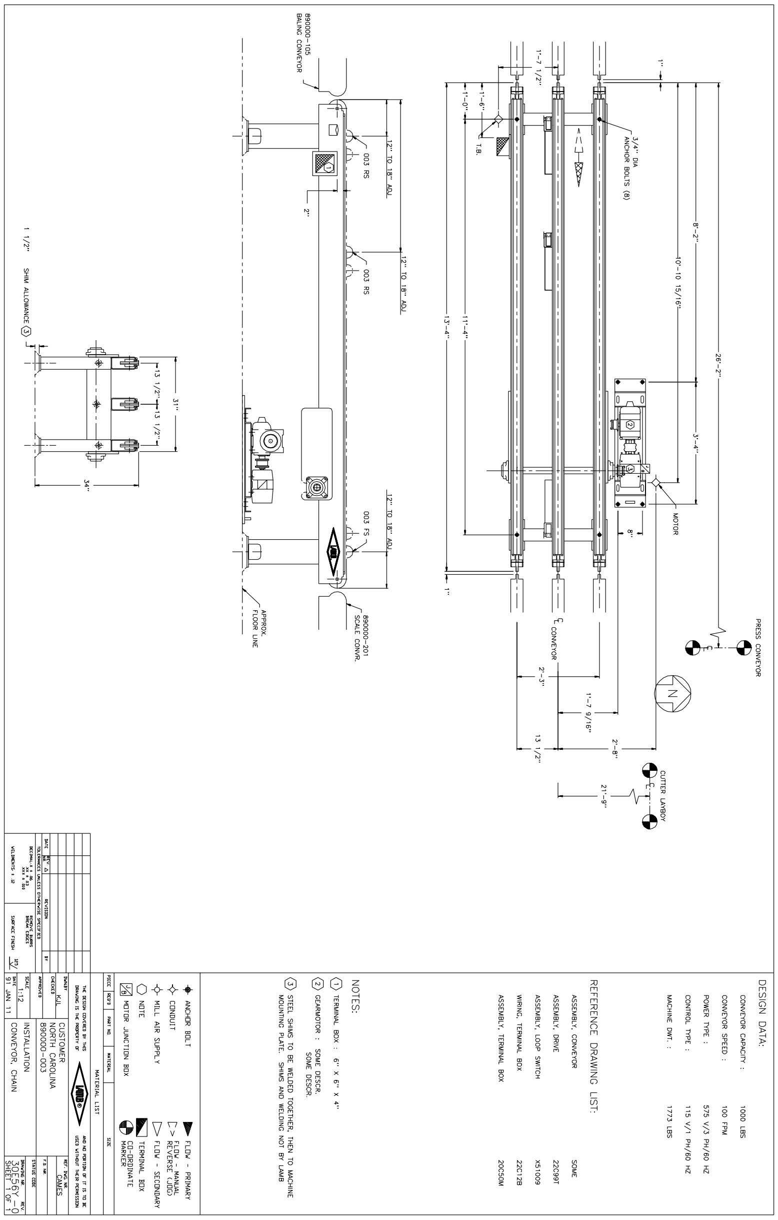


Figure 4. Chain-Conveyor Drawings: Assembly and Installation.

made by our drafting department. For some machines, we overreacted with block representations and had to revert to elementary programming. However, with time and experience, we developed artful skills in which blocks were used to represent certain design details, and functional programming was used to represent other elements of design. We do not yet know how to reduce this art to code.

Innovations Brought by the Application

As this chapter indicated, we believe that CAMES has improved our ability to check errors and improve design quality, increase human productivity for standard machines, increase drafting quality, and allow more time for innovation and creativity and has increased our options for using less skilled personnel. It has acted as a catalyst, even as a driver, for identifying and improving inefficient bureaucratic processes. We hope it will be a vehicle for transferring engineering knowledge between business cycles.

Our users want to continue our CAMES effort because they see personal and corporate benefits; we believe that this continued use is the ultimate endorsement. Some users consider themselves the best machine designers in the world and see CAMES as a natural evolution of their profession. We could not be more flattered.

Conditions for a Successful Application

Users demand that CAMES permit easy modification. However, a user must accept education about how CAMES works and upgrade his(her) general computer skills. For example, a user should have a good working knowledge about the machine being designed, the CAD system used, and some basic DOS skills. Users must actively be involved in the machine-designing process and not expect miracles from CAMES.

The Nature and Estimate of the Payoff to Our Organization

We estimate that by mid-1991, over 1,000 machines will have been designed by CAMES, saving a minimum of 16 hours each and, therefore, yielding a total savings of over 16,000 person-hours during the 5 years of development and use. At a minimum, we claim breaking even for the project to date. However, we don't think of strategic moves in terms of economic payout, and we consider CAMES to be a strategic initiative whose serious contribution will be most significant after the integration of all three systems: CALES, CAPES, and CAMES.

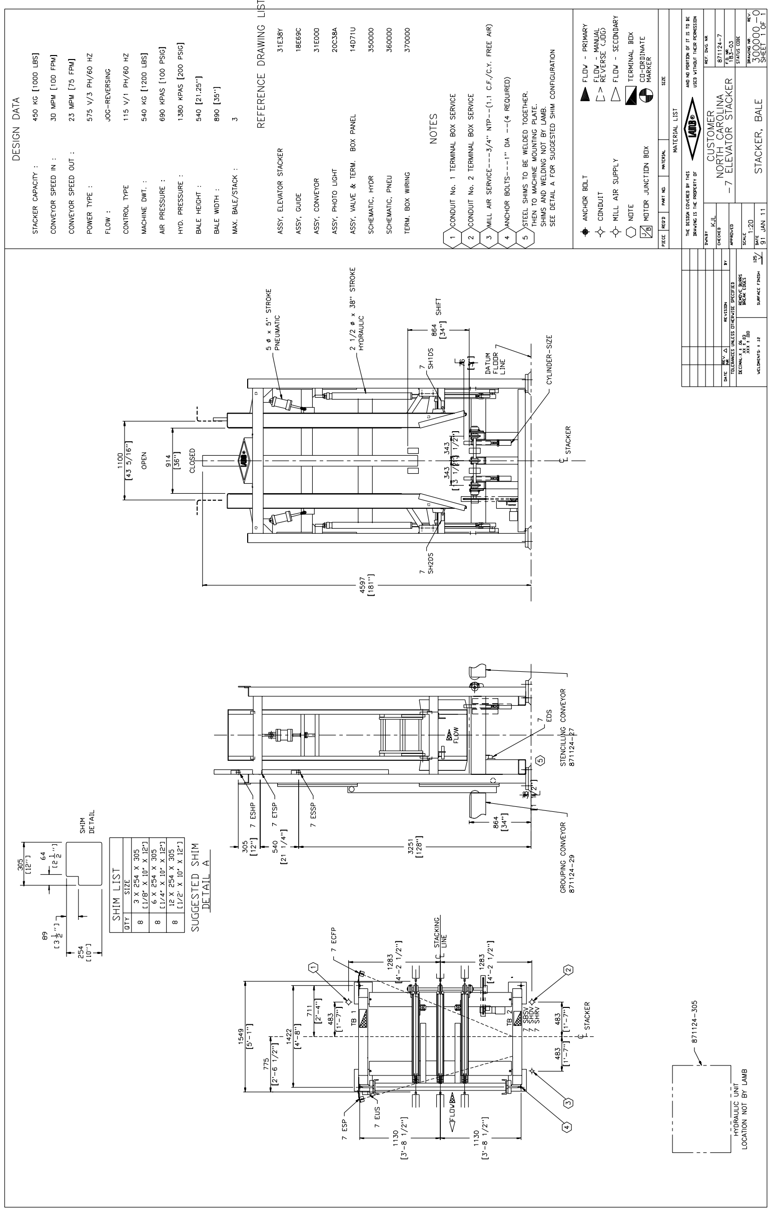

DESIGN DATA
STACKER CAPACITY : 450 KG [1000 LBS]
CONVEYOR SPEED IN : 30 MPM [100 FPM]
CONVEYOR SPEED OUT : 23 MPM [75 FPM]
POWER TYPE : 575 V/3 PH/60 HZ
FLOW : JOG-REVERSING
CONTROL TYPE 115 V/1 PH/60 HZ
MACHINE DWT. : 540 KG [1200 LBS]
AIR PRESSURE : 690 KPAS [100 PSIG]
HYD. PRESSURE : 1380 KPAS [200 PSIG]
BALE HEIGHT : 540 [21.25"]
BALE WIDTH : 890 [35"]
MAX. BALE/STACK : 3
REFERENCE DRAWING LIST:
ASSY, ELEVATOR STACKER 31E38Y
ASSY, GUIDE 18E69C
ASSY, CONVEYOR 31E000
ASSY, PHOTO LIGHT 20C38A
ASSY, VALVE & TERM. BOX PANEL 14D71U
SCHEMATIC, HYDR 350000
SCHEMATIC, PNEU 360000
TERM. BOX WIRING 370000
NOTES
1 CONDUIT No. 1 TERMINAL BOX SERVICE
2 CONDUIT No. 2 TERMINAL BOX SERVICE
3 MILL AIR SERVICE---3/4" NTP--(1.1 C.F./C.Y. FREE AIR)
4 ANCHOR BOLTS---1" DIA --(4 REQUIRED)
5 STEEL SHIMS TO BE WELDED TOGETHER. THEN TO MACHINE MOUNTING PLATE. SHIMS AND WELDING NOT BY LAMB. SEE DETAIL A FOR SUGGESTED SHIM CONFIGURATION
ANCHOR BOLT
CONDUIT
MILL AIR SUPPLY
NOTE
MOTOR JUNCTION BOX
FLOW - PRIMARY
FLOW - MANUAL REVERSE (JOG)
FLOW - SECONDARY
TERMINAL BOX
CO-ORDINATE MARKER
MATERIAL LIST
CUSTOMER
NORTH CAROLINA
-7 ELEVATOR STACKER
STACKER, BALE
871124-7
183-03
300000-0
SHEET 1 OF 1
KJL
1:20
91 JAN 11
5 ø x 5" STROKE PNEUMATIC
2 1/2 ø x 38" STROKE HYDRAULIC
864 [34"] SHIFT
1100 [43 5/16"] OPEN
914 [36"] CLOSED
4597 [181"]
DATUM FLOOR LINE
CYLINDER-SIZE
STACKER
STENCILLING CONVEYOR 871124-27
GROUPING CONVEYOR 871124-29
3251 [128"]
SHIM LIST
SUGGESTED SHIM DETAIL A
SHIM DETAIL
STACKING LINE
HYDRAULIC UNIT LOCATION NOT BY LAMB
871124-305

Figure 5. Stacker Drawing: Assembly and Installation.

As a result of a test conducted in April 1991 (without the benefit of the CALES-CAMES-CAPES integration), several CAMES machines perhaps indicated a reduction in target hours from 32 to 8. However, a process of continuing review was necessary to prove these figures and continue the downward trend to a target total person-hours reduction of 90 percent. This process involves monthly benchmarks and continuous incremental changes to programs and surrounding bureaucratic procedures.

Further qualitative and quantitative benefits can be credited to CAMES. Person-hours assigned to machine packages include design and drafting time but do not usually include plotting or material list completion and vary from 8 to 40 hours for each machine. There are simple cost-accounting problems surrounding the data we collected because the design time for CAMES is so small that operators take responsibility for plotting and bill-of-material tests, invalidating any simple comparison between the budget and actual costs. The design time shrinks, so that bureaucratic time becomes oppressive. Regardless of these problems, it takes only a few minutes for CAMES to produce a manufacturing package. The engineering data necessary for actual entry of the specifications, annual correction of drawings, preparation of bills of material, plotting, and other overhead operations consume a considerable proportion of the time. An obvious possibility is to automate these tasks, starting by using CAPES. In spite of these issues, the accounting data collected to date show that a time reduction of 50 percent was achieved with respect to previous processes. Activity-based cost accounting would reveal more benefits in our opinion.

Some engineers feel threatened by CAMES, as some felt threatened by the earlier proliferation of CAD. When fully implemented, CAMES will actually protect our corporation and jobs in any cyclical downturn of the economy. Some designers and engineers doubt CAMES will be able to process complex machines, but they can offer no theoretical justification for their position. As the developers of CAPES, CAMES, and CALES, we see no theoretical barrier to a complete mapping of customer needs into fully integrated electromechanical designs and preengineered proposals. Our company is in the process of committing itself to this concept of doing business and is considering restructuring and repositioning areas that might use these concepts.

Deployment Times, Costs, and Technology Transfer Problems

CAMES technology was developed in a branch of the Lamb companies in Vancouver, British Columbia, about 250 miles from the main design office. The location caused logistic problems concerning design changes,

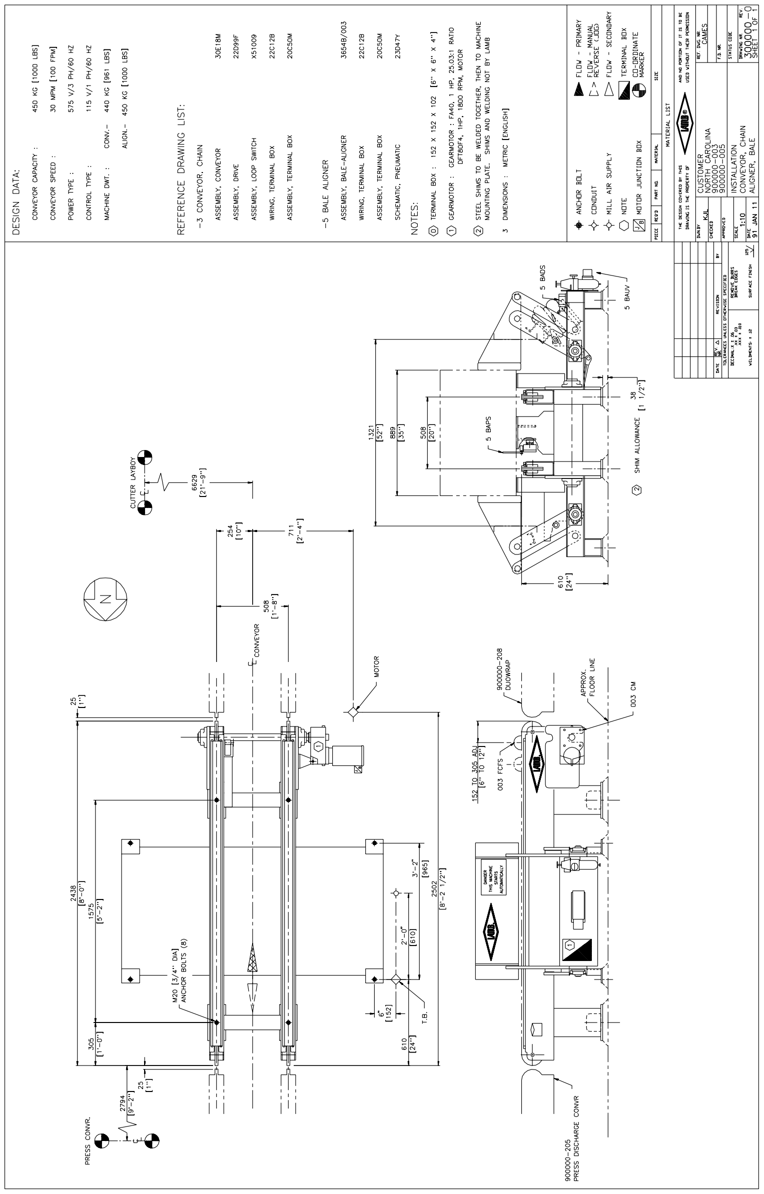

DESIGN DATA:
CONVEYOR CAPACITY : 450 KG [1000 LBS]
CONVEYOR SPEED : 30 MPM [100 FPM]
POWER TYPE : 575 V/3 PH/60 HZ
CONTROL TYPE : 115 V/1 PH/60 HZ
MACHINE DWT. : CONV.- 440 KG [961 LBS]
ALIGN.- 450 KG [1000 LBS]
REFERENCE DRAWING LIST:
-3 CONVEYOR, CHAIN
ASSEMBLY, CONVEYOR 30E18M
ASSEMBLY, DRIVE 22099F
ASSEMBLY, LOOP SWITCH X51009
WIRING, TERMINAL BOX 22C12B
ASSEMBLY, TERMINAL BOX 20C50M
-5 BALE ALIGNER
ASSEMBLY, BALE-ALIGNER 36548/003
WIRING, TERMINAL BOX 22C12B
ASSEMBLY, TERMINAL BOX 20C50M
SCHEMATIC, PNEUMATIC 23047Y
NOTES:
TERMINAL BOX : 152 X 152 X 102 [6" X 6" X 4"]
GEARMOTOR : FA40, 1 HP, 25.03:1 RATIO DFT80F4, 1HP, 1800 RPM, MOTOR
STEEL SHIMS TO BE WELDED TOGETHER, THEN TO MACHINE MOUNTING PLATE, SHIMS AND WELDING NOT BY LAMB
DIMENSIONS : METRIC [ENGLISH]
ANCHOR BOLT
CONDUIT
MILL AIR SUPPLY
NOTE
MOTOR JUNCTION BOX
FLOW - PRIMARY
FLOW - MANUAL REVERSE (JOG)
FLOW - SECONDARY
TERMINAL BOX
CO-ORDINATE MARKER
MATERIAL LIST
CUSTOMER
NORTH CAROLINA
900000-003
900000-005
INSTALLATION
CONVEYOR, CHAIN
ALIGNER, BALE
1:10
91 JAN 11
CAMES
300000-0
SHEET 1 OF 1
CUTTER LAYBOY
6629 [21'-9"]
PRESS CONVR.
2794 [9'-2"]
CONVEYOR
MOTOR
M20 [3/4" DIA] ANCHOR BOLTS (8)
T.B.
SHIM ALLOWANCE
5 BAPS
5 BAUV
APPROX. FLOOR LINE
003 CM
003 FCFS
900000-208 DUOWRAP
900000-205 PRESS DISCHARGE CONVR

Figure 6. Aligner Drawing: Assembly and Installation.

program updates, inquiries, and user training. Now that most Lamb branches are connected through a comprehensive network system, communication between users and programmers was made much easier.

An attempt to transfer the CAMES ramps to the main design office was attempted but was unsuccessful. This failure was because the ramps were processed in a branch office and sent to the main office as a finished package in an established process no one wanted to change. The technology transfer of the chain conveyors and other machines took place in mid-1990 and was generally accepted by users. These users developed a plan to complete additional machines in the course of the next several months. It will take six to eight weeks to complete a new machine package with the system now in use. CAMES development took four years and an estimated six person-years of programming effort to create the existing tool. Because of continued change, the development of CAMES machines will never stop.

A critical move was made in 1990 by transferring the responsibility for the CAMES development to the line manager of the department that will obtain the greatest benefit from CAMES. This move illustrates the implementation of our fourth and most important principle, that the application must be under the direct control of a line operation manager, even if this person has no knowledge of AI or knowledge-based systems. This trend will continue as responsibility for implementing the CAPES, CALES, and CAMES systems is entrusted to line managers as part of a serious corporate reengineering effort by our group of companies, which involves empowering knowledge workers and is accompanied by organizational repositioning, delayering, and downsizing.

Acknowledgments

We thank Phil Parker for his long-term contributions and committment to CAMES.

References

Campbell, I. C., and Bukshteyn, I. 1990. Putting Knowledge-Based Concepts to Work for Generic PLC Programming. In Proceedings of the Second Annual Conference on Innovative Applications of Artificial Intelligence, 108–113. Menlo Park, Calif.: American Association for Artificial Intelligence.