1 Abstract

Re-engineering and reuse are now becoming mature areas within software engineering. Their application to industrial control systems has, to date, been somewhat limited by the lack of tools specific to this specialised domain. In particular a robust, open and standardised implementation model has been lacking. This has recently changed with the increasing availability of IEC 61131 based tools, and in particular, a number of vendors supporting IEC 61131 based programmable controllers.

The paper intends to present a system integrator's view on the re-engineering of legacy control systems using modern IEC 61131 based controllers. In particular, it will examine the processes undertaken to re-engineer the existing Auto Boiler Control systems for a series of coal-fired units ranging in size from 350 to 500 MW on behalf of a large electricity generator. The paper will examine how a library of functions, developed for the initial refurbishment, was reused and extended through the second refurbishment, and our experience of subsequently translating a significant portion of the library to a different IEC 61131 platform for the third refurbishment.

The advent of IEC 61131 and other controller tool advances have appeared in parallel with a move towards a more adaptable engineering approach to systems implementation within the power industry. This has seen customers looking for much shorter implementation timescales, flexibility during commissioning and installed systems that are open to future maintenance and expansion. The paper will go on to discuss the roles that IEC 61131 and other controller advances have played in achieving these goals.

2 Re-engineering – Examining the hype

What is re-engineering, and how does it differ from what we've all been doing for years?

Re-engineering is a term that has recently come to the forefront of software engineering, not least because of the issues surrounding Y2K. It is a term that originates from the re-working of business systems, but has equal importance when considering any software based system.

Re-engineering is the re-implementation of a software system to make it more maintainable.

Often the existing system is a legacy system, that is one that uses outdated techniques for its implementation, obsolete systems for its operation and support, and often lacks the process infrastructure (e.g. documentation, in-house knowledge, configuration management, etc.) for its ongoing maintenance. In many cases these legacy systems are critical to the efficient operation of the business, and the costs associated with their maintenance and/or their inflexibility have reached a point where that efficiency is compromised.

© 2000 The Institution of Electrical Engineers.
Printed and published by the IEE, Savoy Place, London WC2R 0BL, UK.

Normally the re-engineering process will take the existing system (often described in source code only) as the basis for the specification of the re-engineered system (i.e. it will reverse engineer it).

Maintenance of these systems normally involves three processes:

- Resolution of operational problems (normally latent defects)
- Addition of new functionality
- Accommodation of a new operational environment (e.g. new hardware)

In some cases preventative maintenance also features in this list. Here maintenance is performed with the aim of reducing future maintenance costs.

In re-engineering systems we therefore aim to produce systems that allow these maintenance activities to be readily performed. It is this goal that distinguishes re-engineering from the re-implementation of a copy of the legacy system.

So how does this process differ from what we have done in the past?

It could be stated that the production of a maintainable system should be a goal of all re-implementation projects, past and present. There are, however, a number of key areas that should be considered in the production of a maintainable system.

- The needs of the maintenance and user community should be paramount in the re-engineering of the system. It is quite likely that these people are a different set to that involved in the original system development.
- In re-engineering the system, use should be made of current best engineering practice wherever possible, this is particularly the case in the area of software engineering.
- Open standards should be adopted if available and appropriate.
- The use of automated tools to enable the re-engineering process, and to support the future maintenance of the system should be considered.
- The ability to learn from and re-use elements of the systems being re-engineered.

So what role does the IEC 61131 standard have to play in this process?

- The standard represents a significant step forward to opening the programming of control systems to a commonly understood format (it defines the boundaries).
- It provides the basis for cross-platform compatibility and future systems availability (hopefully different vendors systems will support a common core set of features and will go on providing compatible systems well into the future).
- The frameworks it defines allow the use of significantly improved engineering practices, particularly in the area of software engineering, when compared to the previous generation of controllers.

In summary then, re-engineering is about learning from the past with a view to improving on it for the future. The IEC 61131 standard provides a tool for the systems engineer to use in achieving that aim.

3 Project Backgrounds

This section provides some background to the three projects being examined in this paper. All the projects were undertaken by Instem for TXU Europe Ltd (formerly Eastern Generation) across three of their coal fired stations. Each contract was let separately on a tender basis. These contracts followed on from a number of initial Soft Desk contracts for plant control and sequencing. It was during these contracts that the original methods and library were developed. The stations involved were as follows:

- Ironbridge Power Station
  - Located near Telford, Shropshire
  - 2 off 500MW Units and Common Services
  - C & I Refurbishment including APMS Soft Desk
  - Included replacement of existing CUTLASS based control systems
  - Subsequent separate contract for Unit 2 Low NOx conversions
  - Uses Moore's APACS controllers

  Within the scope of the Ironbridge project, the authors took responsibility for the review and replacement of the existing CUTLASS auto control functions. These covered all aspects of the Modulating Auto Boiler Controls, including mill control, feedwater control, furnace draught control, steam temperature control, load control, boiler pressure control and turbine ancillary control. In undertaking the work, the existing CUTLASS control philosophies were analysed by reference to the source code. Where appropriate these control strategies were then updated or replaced to reflect the clients need for a more advanced control solution.

- West Burton Power Station
  - Located near Retford, Nottinghamshire
  - 4 off 500MW Units
  - CUTLASS replacement
  - Followed from a previous contract for APMS Soft Desk
  - Uses Moore's APACS controllers

  This extension to the West Burton project covered the replacement of the existing CUTLASS controls (hardware and software). The authors took responsibility, in conjunction with the station engineer, for the review and replacement of the existing CUTLASS auto control functions. In undertaking the work the existing CUTLASS control philosophies were analysed by reference to the source code. The system was commissioned on a loop by loop transfer basis with the units live.

- Drakelow Power Station
  - Located near Burton on Trent, Staffordshire
  - 3 off units totalling 976 MW
  - Firing controller for Unit 12 once through super-critical 350MW unit
  - Uses Mitsubishi Q series controller

The Drakelow project was a standalone project to implement a new firing controller. Initially the control was implemented across 2 out of 5 mills. The authors took responsibility for the design of the new control philosophy in conjunction with the station engineers.

What is a soft desk replacement scheme?

The scope of the schemes can vary significantly but in essence they involve the replacement of an existing hard desk (hardwired indicators, switches, meters, alarm fascias, etc.) with a complete computer based HMI. This is probably best described by the use of the classic 'before and after' shots. Shown below are pictures of the Ironbridge control room before and after a soft desk refurbishment.

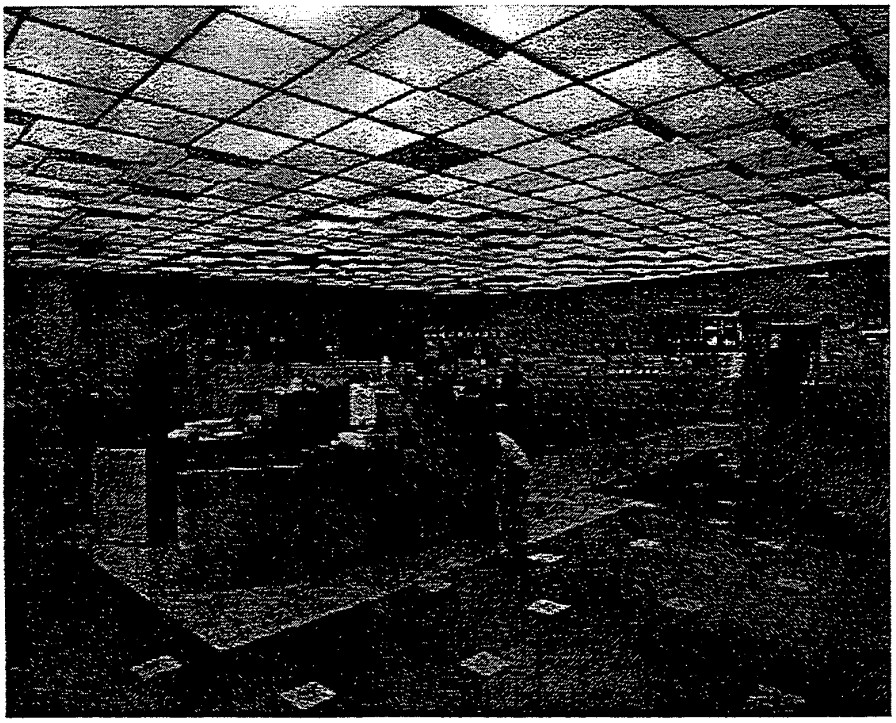

Figure 1 - Ironbridge CCR Pre-Refurbishment (c) TXU Europe Power

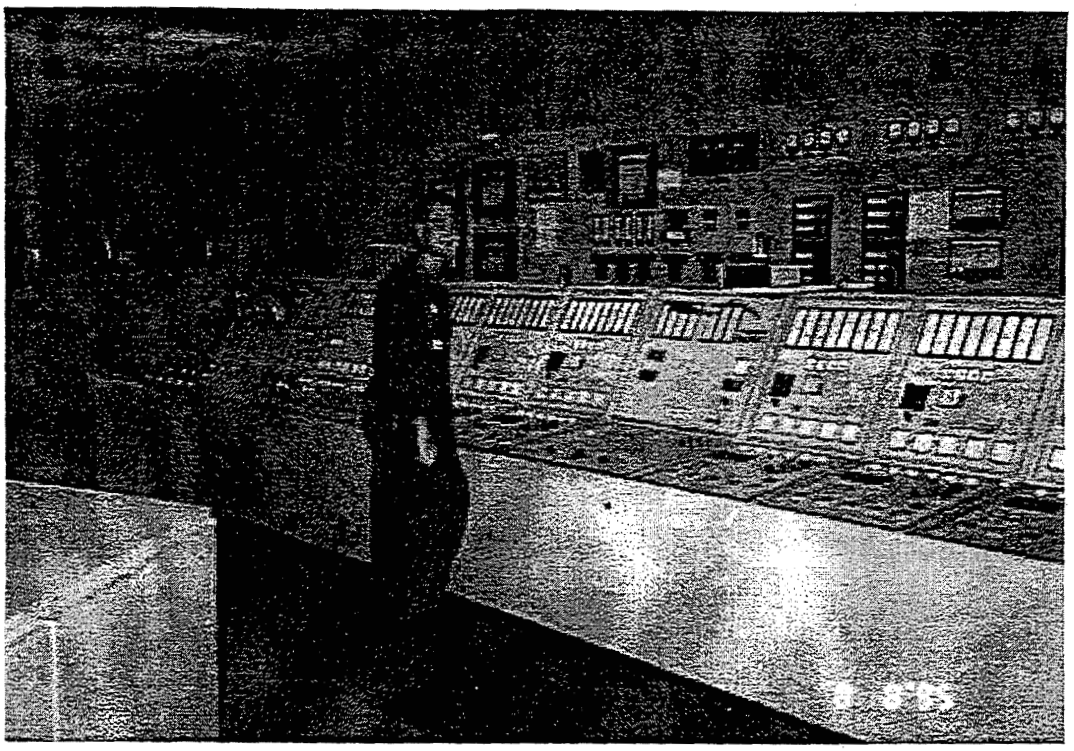

Figure 2 - Ironbridge Hard Desk Pre-Refurbishment (c) TXU Europe Power

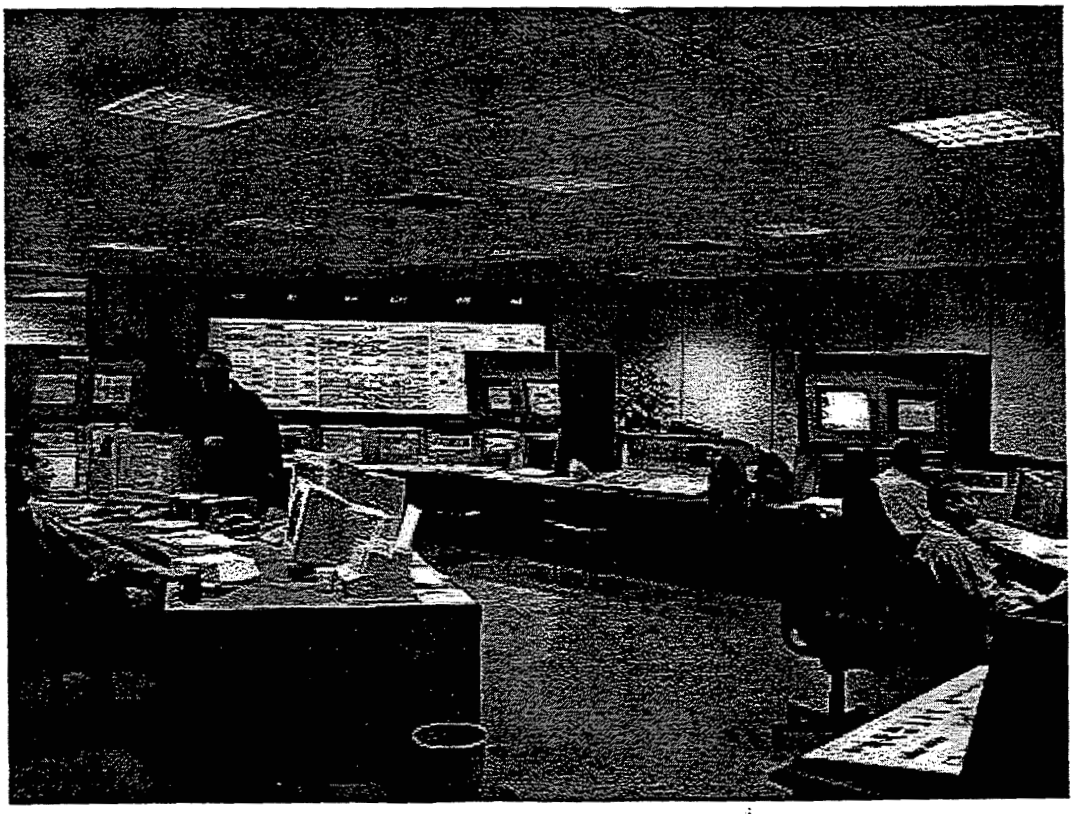

Figure 3 - Ironbridge CCR Post Refurbishment (c) TXU Europe Power

Whilst these pictures provide a visual indication of the impact of the refurbishment, the true scope of the work involved is probably better realised by reference to the architecture diagram for the Ironbridge Unit 1 refurbishment included below.

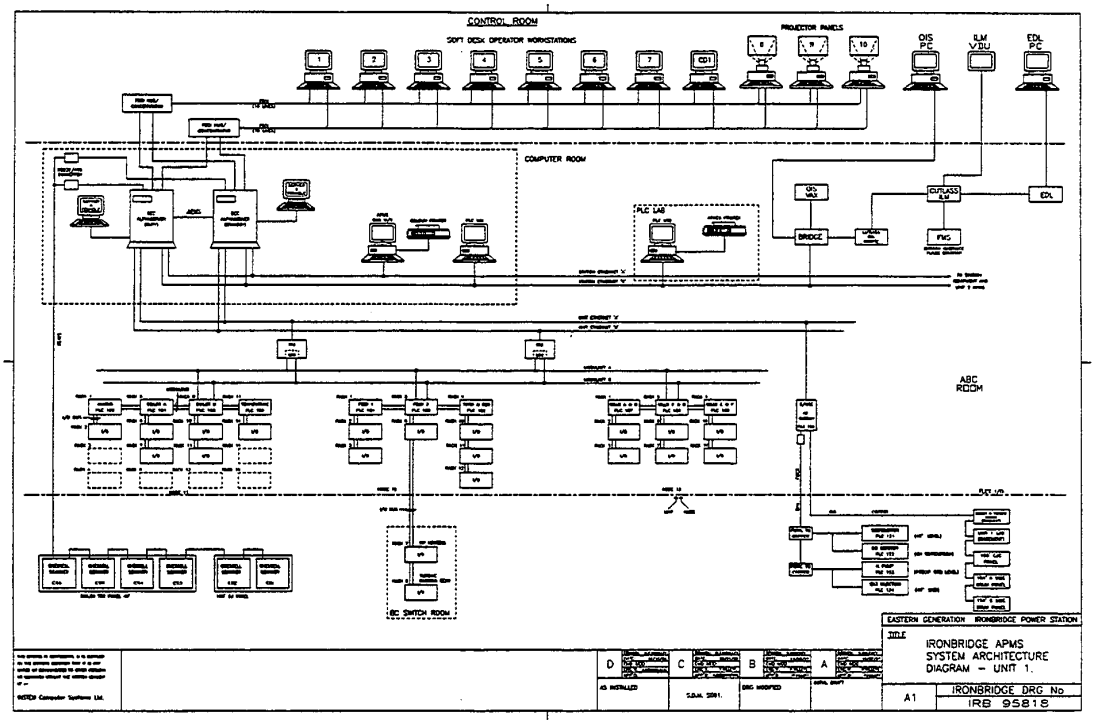

Figure 4 - Ironbridge Soft Desk Refurbishment - System Architecture - Unit 1 (c) Instem

To put all this in context a few facts and figures may be in order.

- Ironbridge represents one of the most challenging soft desk refurbishments undertaken in the UK in terms of its scope, its timescales and its costs.
- It represents one of five stations at which Instem have undertaken APMS soft desk refurbishments totalling 14 completed units representing a total generating capacity of 7.5GW.
- All the systems are based on an APMS top end SCADA system, with Moore's APACS controllers used over the majority.

4 And the products involved?

APMS

APMS is an integrated real-time SCADA system for power plants, developed in partnership by Npower (formerly known as National Power) and Syseca. The application is based upon Agilent Technology's (formerly Hewlett Packard) RTAP product, and in conjunction with Syseca's Open PMS, has been tailored to the power generation market.

Siemens Moore's APACS

APACS is an IEC 61131 based controller. It features the ability to program the controller in Function Block, SFC, Ladder and Structured Text. It is the favoured controller for many UK generators following an extensive survey originally conducted by National Power, and has a number of features that lend itself to this type of control application. These include dual redundant processors, support for user definable function blocks, on-line variable control, real time trending, automated backup and configuration management.

Mitsubishi Q Series

Q Series is an IEC 61131 based PLC. It features the ability to program in Function Block, SFC, Ladder and Instruction List. A Mitsubishi Q Series PLC was already in use at Drakelow and for connectivity reasons this influenced its choice.

CUTLASS

The CUTLASS language was developed by the Central Electricity Generating Board (CEGB) to provide an in-house solution to the need for direct digital control of their many power plants. The language followed from earlier in-house languages of DDACS and Swepspeed, which were used by various regions within the CEGB. The need for several language sub-sets was identified – Sequence control, Modulating Control, VDU handling and a 'General' language incorporating the common constructs found in all high level textual languages, for example conditional statements and loops. CUTLASS was trialled at Skelton Grange Power Station in 1980 and used on many power plants thereafter, including Heysham II Nuclear Power Station.

CUTLASS is a high-level language providing control functions within its own real-time multi-tasking operating system, all written in Coral 66. It has a library of standard control functions. Additionally it supports modular software in the form of subroutines and 'include' files.

The CUTLASS language was ported to several hardware platforms to allow a wide base of manufacturer/hardware choices. These included Dec PDP 11 processors (very common usage), Intel 8086/80286 processors, plus many I/O subsystems including Instem G-Range, Instem I-range, Solatron, Babcock-Bristol, Media, and Burr-Brown. Communications interfaces to common PLC platforms were also supported.

CUTLASS is particularly strong in its handling of BAD data – all function blocks respond robustly to BAD data. Other strengths include the ease of providing bumpless transfer to and from auto control, and the ability to implement incremental PID controllers; this is in keeping with the CEGB policy of using 'stay-put' actuators providing fail freeze transfer to a manual control desk in the event of CUTLASS computer failure. The use of separate sequence, modulating control and general sub-sets maps loosely on to the IEC61131 languages, which aids the migration process.

CUTLASS is still in use at several power stations. The initial problem of a lack of spares has been relieved by the implementation of many Soft Desk/Refurbishment projects. As regards the longevity of the software, the VDU subset has largely been replaced by other systems. It used the traditional screen development methods of pixel placement of boxes, lines, etc. Programming in the VDU sub-set is exceptionally clumsy by modern SCADA standards. The other language sub-sets live on, maintained either by ex-CEGB consultants, software houses or by power station staff themselves. The main limitation of CUTLASS is the inability to view the running programme or its data. Trending can only be provided by configuration of a separate SCADA system, or by writing data via a communications interface, e.g. serial port, and importing and viewing via a proprietary spreadsheet or similar package. In this respect it is difficult to debug the on-line software.

Despite its age - in terms of its functionality, ease of use, and robustness - it is a tough act to follow. Its considerable virtues mean that, on the whole, its end users are very discerning in their choice of replacement systems.

5 How it was done – Re-engineering and re-use in practice

Prior to the Ironbridge project a number of APMS contracts had been undertaken by Instem that had laid down the practices and procedures. Key to these were the use of a central database to control the various development processes and interfaces, and the development of generic software 'devices' to interface to plant items – at this stage the previous projects had been the implementation of sequence, manual and interlock functionality only.

During the Ironbridge project the requirements elicitation phase identified areas within the existing CUTLASS controls that provided acceptable control philosophies and those that required redesign. In each area the CUTLASS code was then analysed to identify detailed control implementation. Wherever possible unnecessary complexity or redundancy was removed. The re-engineering process was a long hand process performed by physical reading of the source code. Within this process, functionality was segregated to identify potential areas of re-use. To successfully achieve this required staff knowledgeable with both the CUTLASS language and with Auto Boiler Control schemes, a measure Instem achieved by the employment of a specialist to work within the modulating team. It was felt in retrospect that this action contributed significantly to the success of this and resultant projects, and that if re-engineering and re-use are to be successfully employed the role of the domain expert should not be under-estimated.

In identifying the areas of functionality to be re-used consideration was given to the following points:

- Flexibility of the function to meet a variety of functional needs
- Implementations of functionality in line with CUTLASS to allow the translation of design structure and data directly where appropriate.
- The provision of a design structure that could be readily related to high level design document descriptions.
- A modulating function library that would allow future re-use.

This process combined both top-down and bottom-up techniques to identify the most suitable implementation.

The use of generic devices in this way has a number of advantages:

- Development and test effort is reduced on subsequent projects
- New generic items can be documented, implemented and type tested in advance of other parts of the project life-cycle
- Test harnesses and other automation tools can be built around the standard library.
- Where generic devices are a plant interface then initial commissioning activities can be centred upon the device and the I/O interface only.
- Interim and incremental functional releases can be provided to site.

Considering now the role that IEC61131 has to play in this process it is clear that the most significant offering is the ability of the user to define Function Blocks and the associated encapsulation and abstraction properties that this brings. There is, of course, no reason why function libraries cannot be developed without such programming features. Indeed this had been done on past projects by Instem using proprietary PLC features such as CALLED procedures and macros or even just cut and paste. The Function Block concept, however, brings the realisation of a library that can be truly considered stand-alone and to some extent portable. We will come back to these points later in the paper.

6 The second time around

We will now consider the subsequent West Burton project. We will do this only briefly, but in doing so will attempt to comment on the methods adopted and the role of IEC61131 within this.

The West Burton project was tested and commissioned on site. The unit was live during the changeover from CUTLASS to APACS and commissioning was done on a loop by loop basis. This necessitated a number of design considerations:

- Plant devices needed to be quickly proved following transfer to ensure that the operator retained manual control of the device (manual was provided through the existing APACS sequence controllers).
- There was a need to link simulation code to the master pressure and load control loops prior to live commissioning to allow specific confidence testing.
- The system inputs required to be proved with parallel running.

In the development of the West Burton system the level of re-use was as follows:

- Of the 21 functions developed for Ironbridge 9 were re-used on West Burton
- The West Burton system required a total of 14 functions, necessitating the development of 5 new generic function blocks
- Of the 5 new function blocks, 3 functions represented interfaces to plant items

In implementing the West Burton project the following comments can be made about the IEC61131 environment:

- The abstraction of I/O points to tags allowed for rapid configuration, input I/O checking and software comprehension
- The provision of user defined function blocks allowed a substantial portion of the generic library to be directly re-used with the additional benefits of increasing customer confidence, reduced testing and faster development.
- As well as library re-use a number of ancillary functions were also directly re-used (diagnostics, etc), including project tools.
- The encapsulation properties allowed a plant simulation to be built within the PLC, connected to the control code, retained during commissioning, and easily removed after commissioning with low risk to existing software integrity. This was also true of a plant interface device that was developed on site to replace a device found to be unsuitable to controlling particular plant items.
- The encapsulation properties also allowed the upgrade of the load controller to a new design, following initial commissioning. This work had originally been excluded due to the associated development risks to project timescales.

Before leaving this first project we will touch upon some of the things that can be found in many modern PLCs but are not directly a result of the IEC standard. These are as follows:

- The ability to view and directly control variable values within a running controller.
- The ability to trend variable values without recourse to a SCADA system or ancillary devices.
- The ability to access the running PLCs through standard APIs.
- The ability to export code into and out of PLCs.
- The support of duty/standby architectures
- Data quality functions

7 Moving Platforms

So far we have considered the impact of the standard only from the feature set it offers in terms of modern programming practices, e.g. encapsulation, abstraction, structure, etc. Another important aspect of a standard is its aid to portability. In now considering the Drakelow project we will explore the practicalities of transferring aspects of an IEC 61131 design, namely the generic library, between platforms from different vendors, in this case Moore's APACS and Mitsubishi Q Series.

At first glance, the task of migrating a general control system design and associated software library from one IEC61131 platform to another may sound relatively easy. However there were some practical difficulties in moving between the two platforms that the authors encountered. To explain these we need to look in some detail at the differences between the two platforms.

Software Development Environment and Software Organisation

The first major difference in moving between the APACS and Mitsubishi Melsec Medoc platforms is the change in the software development environment and the change in software organisation. When considered at the upper level of the environment, both have hierarchical 'tree' structures, but the user interface is quite different in the two cases.

- At the upper end of its hierarchy, APACS uses a tree structure comprising a Module Tree (a view of the nodes, or 'Resources' accessible on the network), a Network Tree (a view of the Program blocks within the selected Resource), and then Program Blocks which may have nested levels of function blocks beneath them. At the lower levels of the software hierarchy, a 'spreadsheet' approach is used, with functions/elements placed on a spreadsheet page, divided into columns and rows. Between the two extremes there is some duality, with the hierarchy being downwardly visible in either a spreadsheet-view or a tree-view.
- Mitsubishi Melsec Medoc uses a more familiar 'project navigator' approach in which the hierarchy can be viewed/or hidden by clicking on 'tree' elements, similar in concept to Windows Explorer. Its software structure differs from that of APACS, and uses Tasks to control the execution of Programs on an event or interval basis. The Programs are called POUs (Program Organisation Units). POUs contain a header in which variables are defined, followed by applications code on a 'Network' basis.

Clearly, moving between the two environments requires a period to gain familiarity with the new environment, which leads to increased product development time on the first project undertaken.

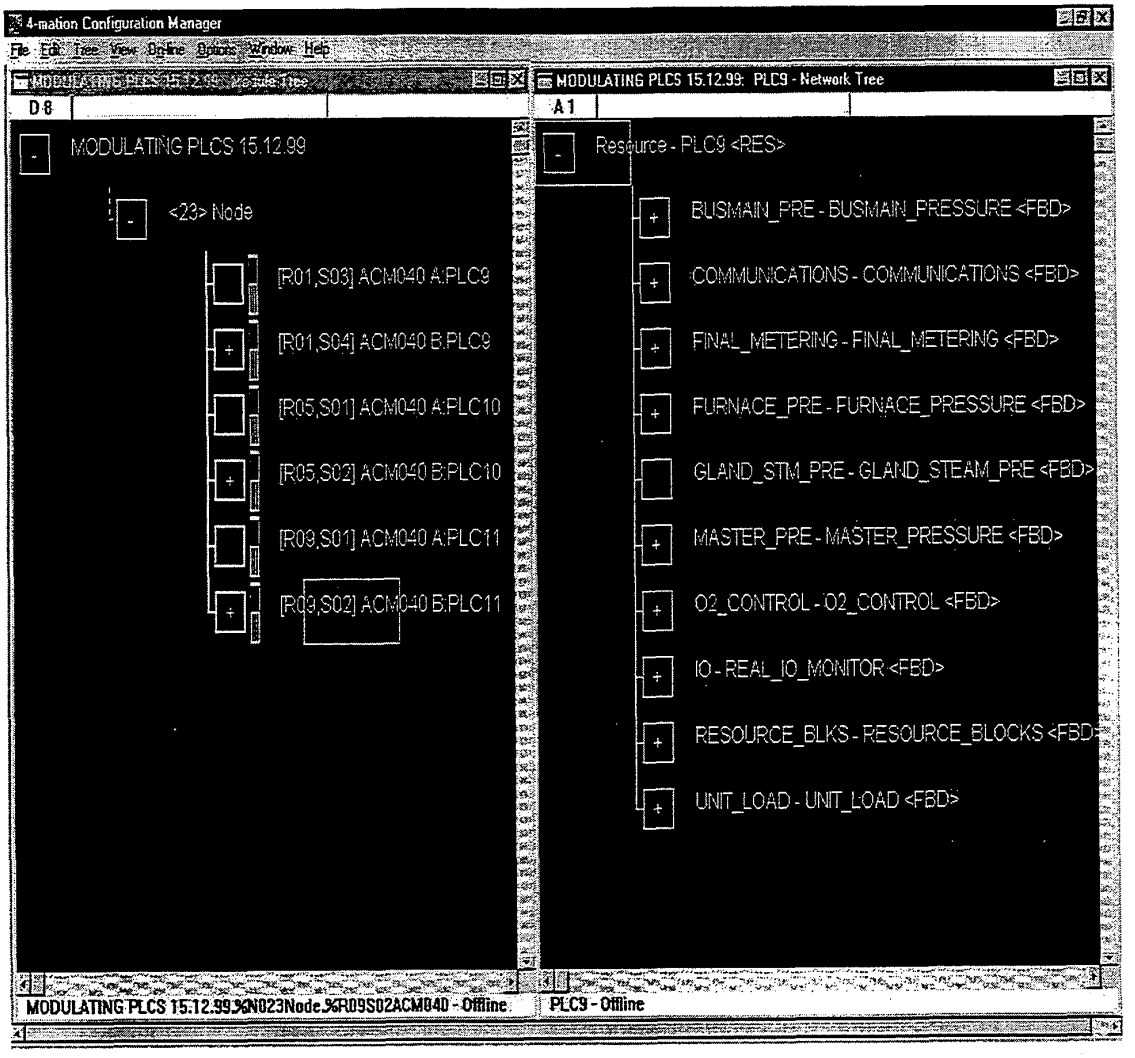

Figure 5 Moore's APACS Module and Network Tree

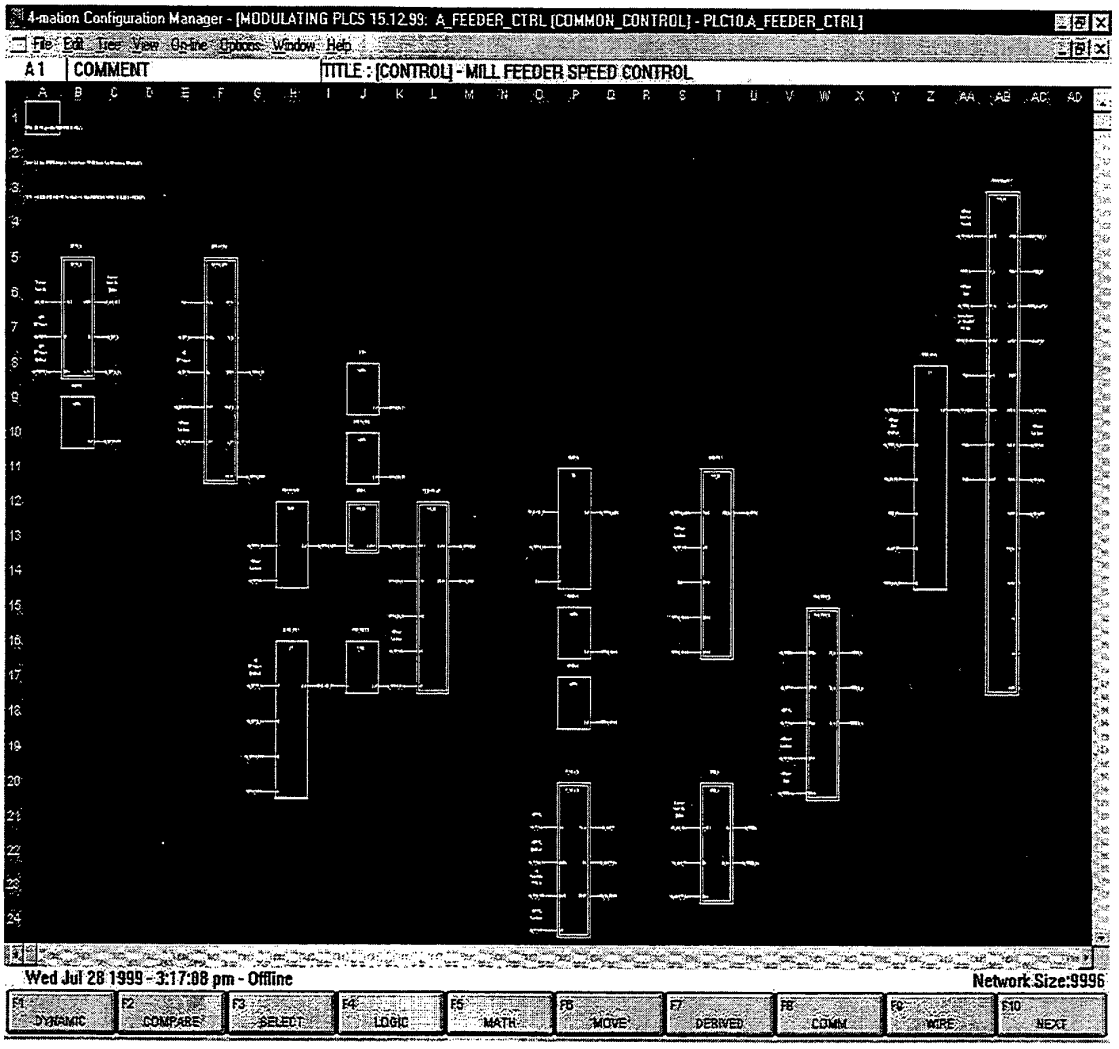

Figure 6 - Moore's APACS Programming Environment

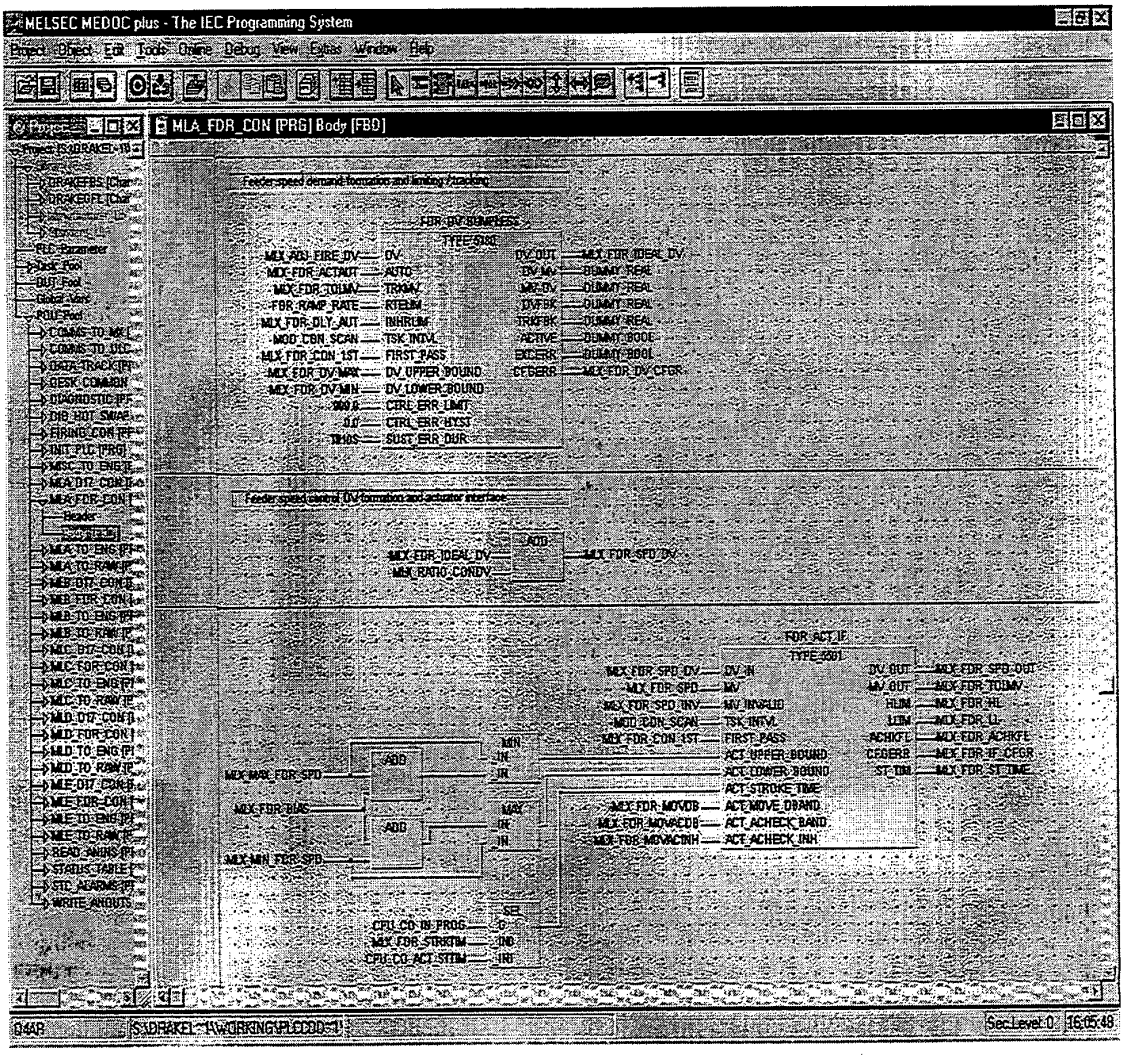

Figure 7 - Mitsubishi Q Series Programming Environment

At the 'program' level, the APACS approach of using a 'spreadsheet' allows the user to view the whole sheet, to zoom into elements on the sheet, or 'double-click' on a function block to view the next level of hidden detail. This ability to hide levels of detail within function blocks allows a comparatively complex structure, for example a control loop, to be implemented on a single sheet, with the ability to view the entire loop's operation, or focus on details within the loop. The ability to zoom in and out in this way is particularly helpful; the user may regain the 'big picture' with a single key press, or mouse click, and then resume a detailed view with a second command. Mitsubishi Melsec Medoc, with its vertical 'network' approach requires a significant amount of scrolling up and down in order to view the program operation. With this type of software structure it is more difficult to maintain a clear view of the overall software structure. This is exacerbated by the requirement that a Network may only contain a single UDFB when implemented as Macro Code, resulting in a greater quantity of Networks (and a greater number of interim variables, hence extra memory usage). This is not a criticism of Mitsubishi Melsec Medoc, but a comment on the more typical 'network' environments in general.

Although IEC61131 does not specify the nature of the software development environment, it does specify the software organisation in terms of Tasks and Programs. In this respect, Mitsubishi's approach appears more compatible with the IEC61131 standard.

Software Run-Time Environment

Mitsubishi Melsec Medoc can be considered as a multitasking environment, whereas APACS is not. Mitsubishi Melsec Medoc may have multiple Tasks, each with their own execution interval (or event condition) and an associated priority. Programs may be allocated to different Tasks and therefore may have differing run rates. APACS executes all its code, in column/row order, at each execution, with the interval globally configured for that particular resource. This can be wasteful in terms of processing time, but the software runs in a more deterministic manner. Mitsubishi Melsec Medoc, and IEC61131 software in general, must therefore be designed to be well structured with robust data exchange (e.g. handshaking) to ensure that the software executes in a deterministic manner.

Availability of Language Facilities across platforms

There were some differences in the language capabilities of the two platforms. APACS was able to support Function Block Diagram (FBD), Structured Text (ST), Sequential Function Chart (SFC), and Ladder. Mitsubishi Melsec Medoc, at the time that it was used for the Drakelow project (late 1999), supported FBD, SFC, Ladder, and Instruction List (IL). Melsec ST was unavailable at the time of the project and was still under development. This limitation did not particularly matter for the Drakelow project which involved modulating auto controls of the boiler and coal pulverising mills. This application uses a library implemented in FBD - a language which is supported by both platforms.

Availability of Function Blocks across platforms

It should be noted that the IEC61131standard only specifies a core of basic functions that should be supported within the FBD standard library. This is limited to data type conversion functions (e.g. Real to Integer), maths functions (e.g. LOG, SIN, ABS), bit-string operation functions (e.g. Bit Rotate/Shift), Boolean functions (e.g. XOR, AND, NOT), selection and comparison functions (e.g. Greater Than, Max Select, Data Select Switch), and Time handling functions. Both platforms support this basic core. Both platforms also provide additional function blocks, many of which are 'higher order' functions that are very useful to the control system designer. These may include dynamic functions such as lead/lag filters, three term controllers, ramp/rate limit blocks, and Auto-Manual stations; logic elements such as flip-flops, and edge detectors; arithmetic functions such as weighted averaging, and piecewise linear function generators; timing and counting functions; and alarm functions. There is no obligation to provide such functions and so each platform may have a widely varying library set.

This above aspect caused a number of problems when moving to the new (Mitsubishi Melsec Medoc) platform, although it must be stressed that this is not a limitation of Mitsubishi Melsec Medoc or its application of the IEC61131 standard:-

- Some of the functions found within both platforms behaved differently, sometimes in important but subtle ways. A detailed evaluation of any new platform is required to tease out these differences, and the importance of prototyping and testing a typical application cannot be underestimated. An example was the TON timer which under Mitsubishi Melsec did not behave as expected when operating in an interval-based program; it would however operate as expected in a free running program. Here the TON block was re-engineered from lower order blocks and added to the project library.
- When embarking on the use of APACS for modulating control, use was made of several of these 'higher order' functions when implementing the user defined function block library. Many of these functions were not present within Mitsubishi Melsec Medoc. The 'missing' functionality had to be constructed from the standard, 'lower-order' functions. An example was the APACS 'RATELIM' block which was used to provide rate limiting of an operator setpoint and the ramping of control error for bumpless transfer of a three element controller.
- Widespread use was made of some APACS 'higher order' functions as stand-alone elements (i.e. not part of a larger library item). It became expected that these would be part of any manufacturer's library. However this was not the case, and these had to be constructed from the standard, 'lower-order' functions. An example was the first order lag filter.
- The corollary is the move to a 'richer' function block set. Here the temptation is to use the 'higher order' functions offered by the new platform to reduce project cost at the expense of portability. This temptation was rejected when moving to the new platform due to lessons learned in migrating other function blocks.

APACS was the first IEC61131 platform encountered by the authors under this series of power plant refurbishment projects. A review of the available functions revealed that some of the APACS 'higher order' functions did not provide the necessary flexibility or functionality, and these were disregarded and re-engineered as user defined function blocks (UDFBs) for inclusion in a project library. This re-engineering process typically used 'lower-order' functions from the IEC61131 standard set, and this aided portability at a later date. However, the possibility of migration was not consciously planned for, and so the benefits were gained by way of a fortunate accident! With hindsight this would have been a sensible policy to assist portability. It is likely, that if Mitsubishi Melsec Medoc had been encountered first, that the migration to APACS would have been easier - Mitsubishi Melsec Medoc is perhaps more in keeping with the intended 'lower-level' functionality' defined by the standard.

A further point of note is that Mitsubishi Melsec Medoc provides, alongside its IEC61131 languages, its own Melsec function set, which is Mitsubishi hardware specific. This is included for historical reasons to give existing Mitsubishi PLC users the option of using a language that they are familiar with. This requires great care to ensure that a project does not use a mixture of platform-specific languages and IEC61131 languages, such that it impedes future portability.

Manufacturer hardware differences and the effect on portability

Each manufacturer may have a number of hardware specific functions that are used to access the PLC's input/output hardware, and to drive/access its communications modules.

Mitsubishi Melsec Medoc requires the use of manufacturer-specific functions to access its I/O, with the conversion of integer raw data to engineering units (and vice-versa), and signal validation. Furthermore, the user is responsible for the assignment of named variables to memory addresses. If the memory is to be allocated to blocks of functionality in a structured manner, then it is necessary to obtain a broad view of the project at the commencement of the project. This necessitates a top down approach. Adding blocks of functionality during a project can de-structure the memory allocation strategy.

In the case of APACS, there is a greater level of abstraction with the accessing of the I/O. The PLC uses configuration parameters of hardware address, (card, slot), signal type (voltage, current, thermocouple, etc), signal scaling parameters (engineering units max/min range) and signal tag name. The result is named variables, scaled in engineering units, available at the applications level, such that memory address allocation is transparent to the user. This is all performed by the 'system software' within the PLC, and this also includes basic signal validation. The implication is that the top-down approach, essential under Mitsubishi Melsec Medoc, is not necessary under APACS; additional functionality can be added later without de-structuring the overall design.

New library functions had to be developed to handle the Mitsubishi Melsec Medoc I/O. No such functions were required under APACS. It was important to keep these new functions distinct from any additional functionality – they had to be Mitsubishi specific - otherwise this would impede any future migration to a new platform. For example, it would perhaps be tempting to read in an analogue signal, validate it, scale it into engineering units, and add first order lag filtering, all in one function. However, keeping the scaling and filtering as separate UDFBs makes the design more portable in the longer term.

A further major difference between hardware platforms is the manner in which the dual redundant processors are handled by the software. With APACS, other than initial configuration to define the system as dual redundant, the programming and software operation is transparent to the user. The standby processor code 'shadows' the on-line processor, and hence the two processors' memory remain in step. With Mitsubishi Melsec Medoc, the standby processor code is dormant and to keep it in step, the user must implement inter-processor communications code using the 'data tracking' functions provided within the manufacturer-specific function set. This has a knock-on effect on the design of the library functions, which may require additional inputs to ensure bumpless transfer when changing from duty to standby processors.

Other miscellaneous differences which affected the migration

APACS imposes a restriction on the number of input and the number of output nubs a function block may have; the number is sixteen. This restriction is not present within Mitsubishi Melsec Medoc. Fortunately, the library was first developed using APACS, and so the restriction did not cause a problem. However there would have been a problem migrating in the other direction. Such details do not appear to be specified by the standard.

APACS supports the use of variable 'Hold' functions (e.g. HLDREAL and HLDBOOL) in which a pre-set value can be stored and then used within the FBD code. These can be used inside a UDFB, and could be used, for example, to hold controller coefficients for a PID block. The important aspect is that each instance of a UDFB can have different numerical pre-sets held within its HLD blocks i.e. the code is generic but the pre-sets are not. This is immensely flexible. The HLD functions can be accessed and modified using the on-line Variable Control facility. Mitsubishi Melsec Medoc does not support HLD variables and if a UDFB's code is to remain generic, yet use different pre-sets, then these must be passed into the UDFB explicitly as input variables. Hence it is quite possible across the two platforms to have two functionally equivalent UDFBs, which have significantly different external structure in terms of input/output nub count. It was fortunate that Mitsubishi Melsec Medoc did not limit the number of input/output nubs; the number exceeded sixteen in many cases. However in a migration to any new platform this may be a restriction.

Summary – Lessons learned from migrating a library to a new platform

In the development of the Drakelow system the level of re-use was as follows:

- Of the 13 functions used
  - 5 were re-used from Ironbridge
  - 1 was re-used from West Burton
  - 1 was re-used from Didcot (another APACS based system)
  - 5 new functions, primarily relating to hardware interfacing, were required

No matter how thorough the design process, it is impossible to take care of all eventualities. The standard is not specific enough to remove the risk from the migration process. Only an evaluation of all potential platforms, prior to commencing the initial design, will avoid the difficulties that the authors encountered. Of course, this is impractical. However there are some steps that can be taken.

- Confirm the level of IEC61131 compliance claimed by the manufacturer and certified by PLCOpen. Identify any major parts of the standard that are not yet implemented. Most manufacturers are keen to demonstrate their products and may be willing to loan equipment for evaluation to those embarking on major contracts.
- Implement a prototype with any new platform and identify any unexpected or restrictive behaviour of functions – each manufacturer may have a subtly different interpretation of the required functionality. Include physical I/O modules, communications modules, dual redundant processors, etc… in the prototype to ensure that the hardware specific features are well understood before commencing the design of any generic library functions.
- The above statistics on UDFB re-use shows that any migration to a platform without abstraction of the hardware will necessitate a number of platform specific functions to be developed to allow integration to existing library items.
- Construct User Defined Function Blocks (UDFB) only from the 'lower-level' function specified by the IEC61131 standard. This should ensure that the UDFBs are migratable across hardware platforms.
- Exclude any functions required to access manufacturer–specific I/O from any library items that may require eventual migration. If necessary, create a platform specific sub-set of library items to cover the hardware specific roles.
- Confirm the availability within the PLC of commissioning tools such as on-line trending and variable read/write control before finalising the library design and defining the commissioning philosophy. For example, it may be necessary to implement additional, possibly temporary, functionality within the Scada system to overcome the lack of trending facilities.

8 Controlling the Process

In discussing the three projects we have concentrated on the practicalities of using IEC61131 controllers. We have now reached the point at which we should take a step back and look at the impact that the standard has in a wider context. By definition standards are intended to bring about benefits by promoting standardisation, and in turn this implies repeatability. Therefore to make effective use of standards you need to look beyond the single project in isolation.

This may be done informally by using the products and transferring the knowledge on a project by project basis through the practitioners, or more formally by establishing procedures, tools, training, knowledge transfer and infrastructures on a cross project, departmental or corporate basis. It is in this latter area that the greatest benefits for the developer can probably be gained.

For a development organisation, gaining the benefits that standards such as IEC61131 offer requires investment outside of the traditional project boundaries found within this type of industry. Particular areas that require addressing at some common level are:

- Configuration management of a library
  - This includes both version control and configuration control, the latter raising the point of retrospective action (i.e. the implementation of recent changes retrospectively on previous sites)
  - The library should be considered as comprising the design, its various platform implementations, test cases and test records
- Design management of a library
- Common tool set development
- Product evaluation against the library and tool sets

These requirements can be in conflict with those of the project, particularly where the project is one of the first 'users' of the technology. It is often the case that initial re-use is applied on a speculative basis, and as additional projects are undertaken it is increasingly formalised.

Of secondary importance is the role that the organisation chooses to take in influencing the standard for its future benefit.

9 In summary

An Integrator's Viewpoint – Tim Wright

The IEC61131 standard has brought a welcome benchmark to the area of programmable controllers. In doing so it allows developers to move away from the discussions about which proprietary controller brings the best solution towards considerations such as:

- What is the best way to program the system?
- What can I easily re-use from the previous system?
- How can I use these products to shorten my development times and costs?
- How will these products improve the quality of the delivered systems?
- What tools can I develop to aid this process?
- What infrastructures do I need to support this process?
- What considerations do I need to make regarding library ownership and copyright?
- Can I buy rather than develop aspects of the application?

The IEC 61131 standard set provides only a first step towards commonality in programmable controllers. Areas that are worthy of further consideration are:

- Standardisation in the area of program export and import
  - At the controller level
  - At the function (library) level

  This is an area that the PLCOpen organisation has centred much of its effort upon.
- Standardisation of supporting functionality
  - Security
    - Access to controller
    - Data and program maintenance (backups, etc)
    - Data forcing
  - Configuration Management
    - Controller configuration management tools
    - Export of files at the library level for management under external tools
  - Diagnostic and supporting features
    - Variable forcing functionality
    - Trending facilities
    - Standard APIs
- Further work towards
  - Standard language syntax and functions
  - Abstraction of information (e.g. data quality associated to the I/O tag rather than the I/O or other module)
  - Architectural issues (e.g. duty/standby, networking, etc).

A User's Viewpoint - Craig Powers

I am able to contrast my experiences of using two different IEC61131 systems whilst working for Instem Limited, with my more recent role as a control engineer at Edison Mission Energy's Fiddler's Ferry Power Station at Widnes, Cheshire.

Fiddler's Ferry Power Station has so far installed APMS soft-desks on three of its four 500MW generating units, and it is possible that the fourth unit may be converted in 2001; a decision is yet to be made. The first unit was converted in 1997 when the power station was owned by Powergen plc. The soft-desk refurbishments were carried out by Instem. The refurbishments were similar in scope to those carried out at Ironbridge and West Burton, but with some significant differences dictated by Powergen. They include:-

- Selection of Allen-Bradley PLCs for all manual and sequence control. These were not IEC61131 compatible and were programmed exclusively in ladder logic.
- The CUTLASS auto-boiler control systems were retained. Therefore the scope of work excluded the modulating control re-engineering that is the main subject of this paper. However, the CUTLASS was modified to interface to the soft-desk system (APMS and Allen-Bradley PLCs).
- There was a greater retention of legacy systems, particularly Allen-Bradley PLCs, which were integrated into the new PLC networks.

Allen-Bradley PLCs had been extensively used on the site in a wide range of duties and had given excellent service over many years. There was a significant amount of product knowledge on the site and choosing them for the soft-desk project was a logical conclusion. However there are some disadvantages in using a non-IEC61131 platform:-

1. The soft-desk project involved the automation of many hundreds of valves, dampers, motor circuit breakers, and other common plant items. Despite the vast amount of plant devices to be controlled, there is a relatively small set of plant interface types; for example regulating valves/dampers with analogue position feedback, isolating valves/dampers with limit switch feedback, actuators requiring open/close command signals, and actuators with open/close latching circuits with a separate stop command signal. Hence a relatively small number of generic function blocks can provide the functionality required to control the bulk of the plant.

   In IEC61131, the use of User Defined Function Blocks is supported. If a UDFB requires modification then this need only be carried out once per PLC; the change propagates through all other instances of the UDFB. The Allen-Bradley ladder logic cannot support UDFBs and the effects on software edits are significant – each 'generic' instance must be edited separately. With potentially many tens or hundreds of a particular 'generic' throughout the entire 'DCS' the most minor of modifications become time consuming, tedious and error prone. The above problem manifests itself as soon as the first 'generic' code item is tested, accepted and then cloned. The only aspect that is 'generic' is the design – the code is not - each instance stands alone.

2. The lack of UDFB support allows the 'generic' code to diverge, either intentionally due to partial modification (i.e. only some instances of the generic are changed), or unintentionally due to errors in editing. This jeopardises the successful re-use of generic code.

3. The use of Ladder Logic is not well suited to some applications. For example, well-structured sequences can be difficult to implement in Ladder Logic. Also, the use of Ladder Logic inhibits the future migration of the auto modulating controls from CUTLASS, should this need arise.

In summary, my experience has shown that software development time, project cost and project risk may be reduced by selecting IEC61131 systems. This has benefits for everyone – the system integration companies, the implementation engineers, the system purchaser, and to the end users themselves. The software maintenance benefits to the end-user are considerable.

A User's Viewpoint – Dave Potter – Ironbridge Power Station

"The choice of control system hardware platform to be used on the Ironbridge Process Control Refurbishment project was the subject of great debate prior to the production of the technical specification for the project. Two options were considered; upgrading the CUTLASS system and extending the use of Allen Bradley PLC's in which the station had invested heavily, or implementing a wholesale change to an IEC61131 compliant platform. The IEC61131 route was chosen despite being more expensive in terms of purchase cost, re-training staff, etc. It was perceived that significant advantages would be provided by the programming environment during project design and implementation stages, and for future maintenance of the system. I believe that these advantages were realised in practice, with 95% of the control systems for a 500MW coal fired generating unit being migrated to the new platform in a 12 week outage with only 5 weeks preparation! I have no doubt that the ambitious project time scale would not have been achievable had an IEC 61131 compliant hardware platform not been chosen."

D.A.Potter (Ironbridge - Process Control Section Head).

10 Acknowledgements:

The authors would like to thank the following for their assistance in preparing this paper.

- Paul Goodson – West Burton Power Station
- Dick Wright – TXU Power Europe
- Bob Fairbrother – Drakelow Power Station
- And particularly Dave Potter – Ironbridge Power Station for the supply of the CCR pictures.

11 Useful Links:

- Instem
  www.instem.com
- Edison Mission Energy
  www.firsthydro.co.uk
- TXU-Europe
  www.txu-europe.com
- PLCOpen
  www.plcopen.org
- IEEE Reverse Engineering and Re-engineering
  www.tcse.org/revengr
- RENAISSANCE ESPRIT project
  www.comp.lancs.ac.uk/computing/research/cseg/projects/renaissance/RenaissanceWeb
- Siemens Moore
  www.mooreproducts.com
- Mitsubishi
  www.mitsubishi.com / www.industrial.meuk.co.uk
- APMS
  www.apms.org.uk
- npower
  www.national-power.com
- Syseca
  www.syseca.co.uk/energy/apms.html