ISA-88 Model based control for a Programmable Automation Controller based Basic Process Control System

Author:

Timothy S. Matheny, President ECS Solutions, Inc.

Revision Summary

Rev.	*Date*	*Approved*	*Description*
0.10	1/1/2014		Initial Draft based on previous paper describing database based control.
1.00	1/10/2014	Tim Matheny	Released to ISA FPID review committee for review and comment.
1.10	2/17/2014	Tim Matheny	Released to ISA for publication with modifications recommended by the ISA FPID review committee

Abstract

The ISA-88 Procedural control model has been implemented in various Batch Management software packages for years. Control software supporting lower levels of the Physical model, particularly the Control Module and Equipment Module levels have been implemented in a Basic Process Control System as custom programming. Recent efforts, including ISA-88-05 WD01, have proposed standard library objects for Control Modules with the goal of reducing custom programming. This paper presents a revolutionary approach wherein a process cell is exclusively described in a data model. Identical Basic Process Control System programming is used for multiple, very different process cells, eliminating custom programming. Also included are presentations of three successful implementations using programmable automation controllers, on three very different process cells, one producing a high-acid food product, one producing a dairy based food product and one producing industrial flavorings.

1. Introduction

The ISA-S88 standard defines a strongly layered approach to developing a batch control system. Within layers, software objects should be reused for control of similar elements. To form a control system, objects must be related to each other. In a traditional implementation, these custom relationships are defined by developing custom control software.

Programmable Logic Controllers were developed to provide a means of allowing control engineers to develop simple software that replaced relays and other electro-mechanical elements of an industrial control system. Programmable Automation Controllers added additional capabilities and languages allowing control system engineers to develop much more extensive industrial control systems. As flexible as programmable automation controllers are, they are typically applied to a specific system with a specifically developed program, so memory and storage resource demand has remained relatively low. Conversely, programmable automation controller capability, particularly in terms of CPU power and memory, has recently increased relatively rapidly.

The ISA-88 Standard proposes a model for controlling a batch process. Batch management software packages have implemented only the higher level portions of the procedural model. The diversity of function and use found in equipment and control modules has led control engineers to model these in custom software. Because each control engineer did as he thought best, systems varied greatly in function.

Developing and maintaining a custom program is expensive. Integration firms and system vendors have developed libraries of reusable software objects in an effort to reduce this expense. ISA-88 Part 5 was started with a goal of increasing reusability by defining standard interfaces. The panacea remains an object based program that would be configured to control a wide variety of processes by building a model of that process—similar to current batch management packages.

This paper proposes that basic, coordination and procedural control software for an S88 control system can be accomplished through model based control in a programmable automation controller. It presents several successful example implementations.

2. The ISA-88 Standard

Building a structured model requires a structured standard. The ISA-88 standard for batch control (hereinafter, S88) provides such a structured model. S88 part 1 defines models and terminology. The standard is available from the ISA website, www.isa.org.

Development of a generalized control system, that is, one with reusable programming that can be configured for varied applications, is dependent on existence of standards which define language and approach. S88 provides the language and general approach required to develop a model based, programmable automation controller based, batch process control system.

Application of S88 to continuous process control systems has been a matter of passionate debate. The model based control concepts discussed in this paper are very applicable to continuous process control systems.

2.1. The Process Model

S88 breaks down processes according to its process model.[1] Processes are made up of one or more Stages. Stages are made up of one or more Operations. Operations are made up of one or more Actions.

In programmable automation controller based batch process control systems, the Process Model is generally implemented in the Batch Management System software, therefore is only of reference interest in this paper.

2.2. The Physical Model

S88 breaks down physical assets according to its Physical Model.[2] The highest levels of the physical model, the Enterprise, Site and Area levels, are generally beyond the scope of a control system although batch management software may support definition of an Area model. The lower four levels are the Process Cell, the Unit, the Equipment Module and the Control Module.

2.3. The Equipment Entity Model

S88 defines equipment entities as a collection of equipment grouped to accomplish a control function or functions. Equipment entities include their physical part and their control part. S88 describes relationships between various equipment entities in its Equipment Entity Model.[3]

---

[1] ISA-88.01-2010 Batch Control Part 1: Models and Terminology, International Society of Automation, Page 29, Figure 2.
[2] Ibid, page 32, Figure 3.
[3] Ibid, page 34, Figure 4.

This paper is most interested in the Equipment Entity Model. This is where the equipment interfaces with the control software. Programmable automation controllers regularly perform much of the basic, procedural and coordination control as defined by S88 for Units, Equipment Modules and Control Modules.

2.4. The Procedural Control Model

S88 breaks down procedural control according to its Procedural Control Model[4] which includes procedures, unit procedures, operations and phases.

In programmable automation controller based batch process control systems, the Procedural Control Model is generally implemented in the BMS software. The customary interface between a batch management system package and the batch process control system is through the Phase, which is the lowest level of the Procedural Control Model.

2.5. Modes and States

S88 allows that equipment entities and procedural elements may have modes and states. The orderly definition of modes and states is critical for developing the control programming for a model based control system.

2.6. Types of Control

2.6.1. Basic Control

Basic control[5] is the portion of equipment control that control engineers are the most familiar with. Basic control establishes a specific state or condition of a piece of equipment.

In programmable automation controller based batch process control systems, basic control is generally performed in the programmable automation controller.

2.6.2. Procedural Control

Procedural control[6] executes procedures. Procedures are an ordered set of steps, actions or activities with a defined beginning and ending intended to accomplish a process objective.

In programmable automation controller based batch process control systems, procedural control is generally performed within the batch management system package. When Equipment modules include procedural control, this is generally implemented in the programmable automation controller.

2.6.3. Coordination Control

Coordination control[7] manages the many equipment assets in a batch. Coordination control includes—

[4] Ibid, page 47, Figure 9.
[5] Ibid, page 44.
[6] Ibid, page 45.
[7] Ibid, page 52.

- Allocation of equipment entities to batches, recipe steps, process actions or other equipment entities.
- Interlocking of equipment entities by other equipment entities.
- Synchronizing of procedural elements.
- Arbitration of requests to allocate or interlock equipment entities or to synchronize procedural elements.
- Propagation of modes, states and control parameters.
- Collection of report parameters.

In programmable automation controller based batch process control systems, coordination control is generally split between the batch management system package and the programmable automation controller. The batch management system provides coordination between procedural elements and allocation at a Unit level. The programmable automation controller provides coordination control at the equipment module and control module levels.

2.7. Software implementations of the ISA-88 Model

Although designed to show information flow through the ISA-88 models, the figure below is useful in understanding the thrust of this paper.

Recipe Entities, depicted left of the heavy dashed line, have traditionally been implemented in batch management software products. Users build a partial process cell model using a tool sometimes referred to as an equipment editor. Users configure phases, or activities, that the configured equipment can undertake. Users develop procedural recipes which coordinate the phases to produce the desired product.

Control of Equipment Entities, depicted right of the same line, has traditionally been implemented in controllers. Users develop a custom program with a custom data table to provide the functions of the "Equipment Oriented Procedural, Coordination, and Basic Control" or Basic Process Control System. The Equipment Procedural Elements provide a structured "mapping" between the procedural control model phases and activities of the Basic Process Control System.

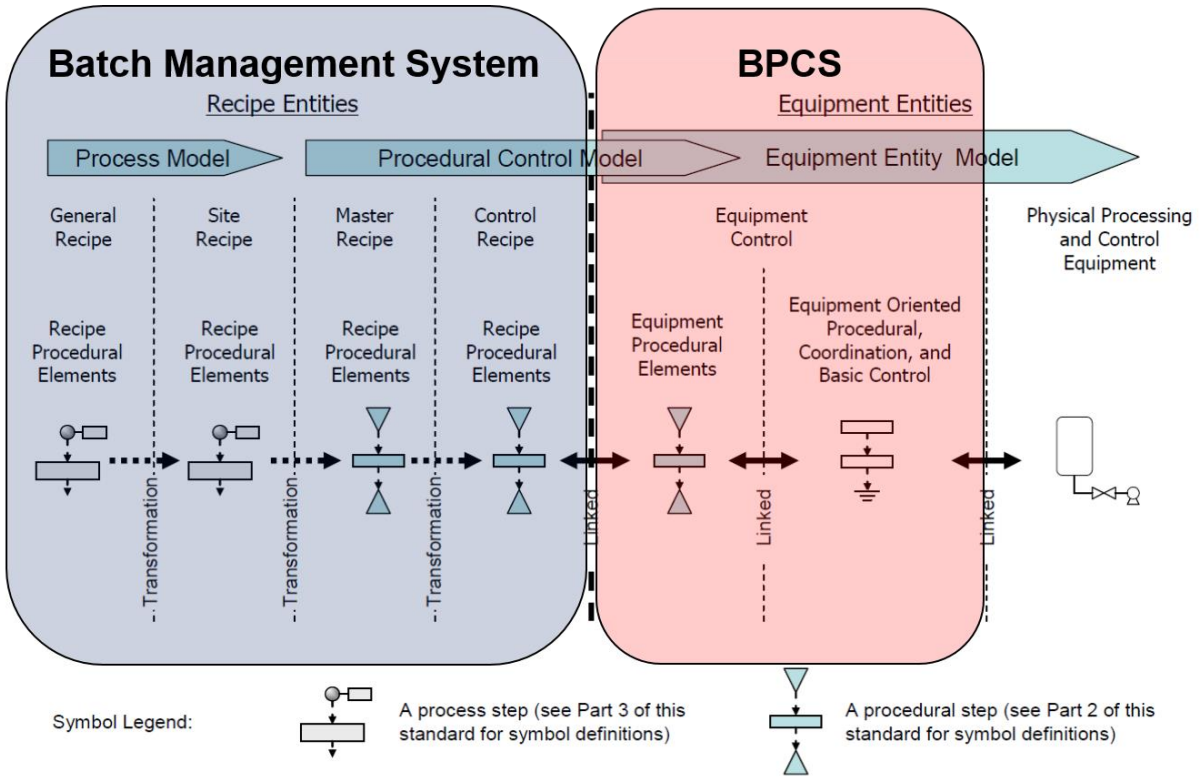

Figure 1-Relationship between the Batch Management System and the Basic Process Control System[8]

3. Implementing an S88 Control system on a Programmable Automation Controller

Implementing an S88 Basic Process Control System using a programmable automation controller platform often requires integration of three major components, the BMS software, the control software, and the HMI software.

3.1. Batch Management System Software

The batch management system package generally encompasses the process model, most of the physical model and most of the procedural model. It generally resides on a personal computer server. It generally communicates with other portions of the overall system using a network.

3.2. Control Software

The control software resides in the programmable automation controller. It includes basic and coordination control for control modules, equipment modules and units. It also contains any procedural control required in equipment modules.

[8] Ibid, page 65, Figure 15.

Procedural control of units is generally implemented in the batch management software as recipes. Control modules cannot have procedural control per S88.[9]

Historically, programmable automation controller-based basic and coordination control are implemented as custom software. Custom software is relatively rigid. If a process designer would determine that a particular control module should be owned differently or interlocked differently, a programmer must modify the codified relationships. Involving the programmer, accurately and completely describing the needed change, and validating the resultant modifications takes time and involves risk.

3.3. Human-Machine Interface Software

Human-Machine Interface for the control system is generally provided by one or more separate devices, which are often personal computers. These devices generally communicate with other portions of the overall systems via one or more networks, often using protocols such as OPC.

3.4. S88 Control System Integration

Modern programmable automation controllers can communicate using networks designed for certain classes of communication. Networks provide the physical integration between the BMS, programmable automation controller and HMI platforms.

Integration between the PC based batch management software and the programmable automation controller based control software is generally accomplished at the phase level. A Recipe Phase defined in the batch management software references an Equipment Phase in the programmable automation controller based control software. S88 also allows integration to be accomplished at other levels.

Integration between the PC based BMS software and the HMI software is accomplished in several ways. The operator may simply use either a thick or thin batch management software client installed on the same computer as the HMI client, using each independently as required. The BMS software may offer modules which can be incorporated into the HMI software application in order to offer the operator access to control software and batch management functions within the same graphic.

Integration between the programmable automation controller and the HMI software is a primary function of the HMI software, generally utilizing OPC services.

[9] Ibid, page 52.

4. Designing a Model for a Model-Based S88 Control System

4.1. Algorithms + Data Structures = Programs

Niklaus Wirth's classic[10] teaches the importance of the data structure in modeling reality. An appropriate data structure models the real world in an appropriate detail and in a way that the algorithms of the program can powerfully manipulate the data but remain simple.

The key to effective model based control is the data structure or process cell model that the algorithms or software will use to perform the needed control.

The ISA-88 Standard offers a model for process control that can be adapted to an effective process cell data model. An effective model goes beyond the lists or schedules of the various elements of the process cell, such as instruments, valves, and vessels, to include relationships between elements. For example, the model should support referencing an instrument as "TT1105" and as "T103 temperature." Further, a particular element, such as a valve, must be part of (have a relationship with) several specific equipment modules. ISA-88 Part 5 draft documents begin to describe standard interfaces between elements. An effective model data structure must assume such interfaces exist in the underlying algorithms.

4.2. Classifying physical assets

A typical data model design seeks to represent many similar entities as rows in a table or array. The model definitions in the ISA-88 standard define similar entities as the levels of its models. In order to more efficiently design and deliver a more effective control system, a control engineer is challenged to classify system elements. Elements of the same class or similar sub-class may be controlled by copies of the same software elements or by using subroutines. It is likely that elements of the same class would share similar representation on drawings and on any HMI graphics.

The Physical Model and the Equipment Entity Model begin to classify entities as Units, Equipment Modules and Control Modules. A complete process cell model will include additional entity classes, but these form the basis from which to begin.

4.2.1. Classifying physical assets by the S88 Physical Model

The first step to classifying physical assets is to classify them according to the basic classifications in the S88 Physical Model: Units, Equipment Modules and Control Modules. The most important consideration is the type or types of control that will be exercised and the range of that control. The three types of control described in S88 were discussed earlier—Basic, Procedural and Coordination. The range of control considers which equipment entities must be controlled in the performance of a particular process task or action.

---

[10] Wirth, Niklaus, Algorithms + Data Structures = Programs, Prentice-Hall, Inc., 1976.

4.2.2. Classifying Control Modules

An equipment entity should be classified as a control module if basic control is performed. Most equipment entities will be control modules.

An instrument or sensor, which does not directly perform basic control, may be a control module if it is shared by a number of process tasks. Alternatively, sensors and instruments may simply be parts of a control module which performs basic control.

Breaking down the S88 control module level for a process cell might give us the following list—

- Valves
- Pumps
- Agitators

Each of these classes might be further subdivided, lengthening the list of process cell control module classes. For example, there are different types of valves such as two-way, three-way, mix-proof, etc. The number of classes required to encompass all possible control modules would seem to be infinite, but in reality is not. In practice, the control engineer can classify a vast majority of the control equipment found in a process, even in a very large process, into a small number of control module classes.

4.2.3. Classifying Equipment Modules

S88 defines equipment modules[11] as a collection of control modules and subordinate equipment modules. An equipment module can carry out a specific number of minor processing activities. Equipment modules are best classified by processing activity. Equipment modules are a team assembled to accomplish a process task or action.

Basic control in equipment modules is generally performed by the control modules it contains. Procedural control in equipment modules is limited to that which is described at or below the phase level of the Procedural Control Model; recirculating when a surge tank level is high and resuming transfer when that level lowers. Coordination control in equipment modules includes allocation and interlock plus mode and parameter propagation for included control modules.

The number of equipment module classes would also seem infinite, but in reality is not. Examples of Equipment Module classes include—

- Transfer to level, volume or weight of source or destination
- Mix, circulate, heat or cool for time or to temperature

Any class may include equipment modules, which are made up of different classes of control modules, different numbers of control modules, different types of volume measuring instruments, etc.

[11] ISA-88.01, page 25.

4.2.4. Classifying Units

Vessels and other major pieces of process equipment, such as filters, are not units, although they often lend their names to units. S88 defines a unit as "a collection of associated equipment modules and/or control modules that carry out one or more major processing activities." A unit may be named the same as a vessel, but actually includes the instruments, valves, etc. that are part of the unit carrying out its function(s).

Units should almost always perform basic control through the equipment modules and control modules they contain. Procedural control is performed on the unit by a recipe unit procedure and in a unit by an equipment unit procedure and/or by operation procedures. Coordination control in units includes coordination of owned equipment entities and coordination with other units plus propagation of modes and parameters.

The subject of defining unit boundaries is far too broad to address here and is of marginal importance to discussing the thrust of this paper.

4.3. Defining Physical Asset Relationships

As a system element is classified, the control engineer must define relationships between that element and other elements. An effective data structure for a process cell model must allow bi-directional analysis of data relationships.

For example, the set of control modules which must be acquired by a given equipment module is an obviously valuable relationship, as is the set of control modules which must be interlocked by that equipment module. Also valuable is the phase that is currently utilizing the equipment module or the equipment module currently using or interlocking a certain control module.

Encapsulating entity relationships in a complex data structure allows a control system engineer to implement features that would otherwise be very difficult or even impossible. For example, the relationship between a control module (pump) and a given equipment module, along with the relationship between the equipment module and a unit, allows the pump control programming to access the material being pumped, that is, the material currently in the unit.

Coordination control is greatly aided by knowledge of entity relationships. For example, a batch management system may request initiation of a phase on a particular unit. The phase-unit combination implies a specific team of entities, an equipment module, which needs to be initiated to take the needed action.

5. Advantages of Model-Based Process Control

5.1. History

ISA-88 describes a model-based process control system. Initial implementations of the models of ISA-88 did not include Equipment Modules and Control Modules. Even in this limited implementation, model-based control proved to be a tremendous improvement over its alternatives.

5.2. Agility

Model-based control offers users tremendous agility. Agility manifests itself as reduced time to implement new products and as more effective use of process equipment entities. By extending model-based control into the basic process control system, that is, to the equipment modules and control modules, agility is significantly increased.

5.2.1. Agility in Custom Programming

Custom programming codifies relationships between control system entities in the programming language of the selected controller. An accomplished programmer thoroughly understands S88 and develops his programming with appropriate granularity to allow the BMS to flexibly control the process cell. A less accomplished programmer is less successful at developing a Basic Process Control System that truly follows S88.

Control system vendors, integrators and end-users have developed libraries of software objects and specialized interfaces, such as Rockwell Automation's PhaseManager® interface, in attempts to push programmers into delivering granular, flexible, agile, Basic Process Control System software. There is no question that these efforts have made gains.

In comparison, a model-based control system presents the model as lists of entities and entity relationships or even graphically. The underlying system programming, or "firmware," must be granular and must tightly follow the S88 standard, on which it fully depends, to be universally applicable.

5.2.2. Agility in implementing new products

Adding process equipment to a process model is as simple as adding a row of data to a table that includes other equipment of the same class or by drawing the additional equipment into a process flow diagram.

Deleting process equipment that is abandoned is as simple as deleting a row or removing elements from a process flow diagram.

Defining a new equipment module is as simple as selecting control module IDs for the set of control modules that an equipment module needs to acquire. This might be accomplished using pull-down lists or by selecting multiple control modules on a process flow diagram.

Interlocking new process equipment is similarly simple. A model builder simply defines the set of control modules that an equipment module needs to interlock.

5.2.3. Agility in asset utilization

Agility in asset utilization is achieved by late-binding of equipment assets to a specific batch and by executing multiple batches simultaneously.

In order for a batch management system to effectively manage batches, it must be supported by a fully compliant basic process control system. Because basic process control systems developed to date have been custom controller programs, compliance with the ISA-88 standard varies by the programmer's

ability and understanding. Even slight deviations from the standard hinder the agility of the entire control system.

As shown above, a model based basic process control system is dependent on a standard, such as ISA-88, for its implementation. It becomes essentially impossible for a model-based control system to be implemented with other than full and complete compliance with the standard.

5.3. Consistency

5.3.1. Function

The system programming or "firmware" of a model-based control system remains the same for all process cells. Process cell control systems are consistent with one another because they run the same programming. The programmer-to-programmer variability found in custom programmed control systems no longer exists.

5.3.2. Troubleshooting

Site issues will fall into three categories, model issues, recipe issues and hardware issues. Programming bugs are an unlikely site issue as system software bugs will affect multiple systems. Model and recipe issues can be tested offline using the simulation built into the system software. Site personnel and support resources can be confidently directed to focus on troubleshooting for hardware issues.

5.3.3. Operator Interface

Great gains in process cell-to-process cell and site-to-site operator interface consistency have been made by use of reuse library objects. Implementing a model-based control system will perpetuate those gains.

A model-based control system does not require a certain look and feel to the operator interface objects. The objects must only be developed to take advantage of the highly structured tags that comprise the runtime model in the programmable automation controller(s).

5.4. Process Orientation

The developer (builder) of a process model based control system will take a very different approach to producing that system than a developer (programmer) of a custom program based control system.

A programmer is focused on programming. Historically, programmers have focused on the controllable aspects (outputs) and observable aspects (inputs) available to the controller for each piece of process equipment. A programmer asks the questions, "Under what conditions should this valve open?" and "Under what conditions should this valve close?" The resultant program tends to be as effective as the programmer is thorough at answering these sorts of questions.

A process model builder is focused on the process. A process model builder will focus on the capabilities of individual pieces of equipment and on how that equipment will be used in the process. The physical equipment, or control module, "lists" that make up the model and the physical relationships of the

equipment, e.g. pipes that connect two valves, will be built by drawing a process flow diagram. The tasks performed by sets of the equipment such as "Add Water" (Phase) and "Add water to Unit 1" (equipment module) will be defined by selecting equipment on the process flow diagram.

This change in focus offers advantage in several ways. The process model builder does not have to be a programmer, he or she can be any process knowledgeable individual. By focusing on the process, the process model builder will make fewer errors and those errors will be more easily correctable without introducing additional errors. By defining the required process tasks on top of the physical model, the process model builder will be more thorough in implementing appropriate permissives and interlocks.

5.5. Validation

In a model based control system, the software is version-controlled system programming or "firmware". Providers of model based control systems will be able to ship validation documentation with specific "firmware" versions. Users will attach providers' documentation to their own documentation for submission to governing bodies.

Model based control systems will implement simulation so that models can be tested offline. Users will test model function against simulation, significantly reducing commissioning effort. Users will be able to test new batch management system recipes prior to validation trials on the physical process system.

5.6. Life-Cycle Cost

5.6.1. Initial Cost

Implementing a model based process control system will be significantly lower cost than implementing a programming based control system. Process designers and product designers will be able to build the basic process control system and develop the product recipes in a similar amount of time it takes today to tell the control system programmer how the resultant control system needs to perform. Today's programming effort will be essentially eliminated.

Commissioning a model based process control system will be both faster and lower cost, so that new products or new capacity can be brought on-line in fewer weeks for lower cost. Errors not identified until commissioning will be fixed quickly and without creating additional errors.

5.6.2. Ownership Cost

Benchmarking of several implementations has shown several areas of ongoing ownership cost reductions and capacity improvements.

Capacity improvements were achieved by improved asset utilization. Users produced more batches by having a process control system capable of managing multiple batches in a single process cell. Advanced features, which are less expensive to implement in the firmware of a model based system, reduced downtime event duration. Recipe-based clean-in-place allowed users to optimize clean-in-place procedures to obtain effective cleaning in less time.

Cost reductions were achieved by reduction in required material and human resources. Material requirements were reduced by reducing bad batches. Human resources were reduced due to increased automation. After implementing recipe-based clean-in-place, cleaning was made an operator function, eliminating the need for a specialized cleaning team.

5.6.3. Security

In a model-based system it is the data in the model that uniquely characterizes the control system. Protecting, including backing up, the control system involves protecting the model definition. Typically the model definition will be stored in a relational database. Robust tools for controlling access and for backing up data are readily available for relational databases.

5.6.4. Guided troubleshooting

A model-based control system can provide detailed guided troubleshooting due to the equipment entity and control entity relationship data included in the model. With more information available to build effective operator panels, operators become effective system troubleshooters.

Experience shows that a large number of problems can be diagnosed and corrected by system operators without maintenance assistance. Another significant number is resolved more quickly when the operator can correctly direct the maintenance worker to the root cause. Cost-of-ownership is reduced when down events are resolved more quickly and involving fewer personnel.

6. 60-Unit, Cooked, High-Acid Food Product Process Cell

6.1. Interest

This process cell is of interest because it is a highly networked cell capable of simultaneously producing a number of batches of a number of different products while cleaning unallocated units. Implementing flexible allocation control using a programmed approach would have resulted in a very lengthy and complex program. Using model based control was, relatively, very straightforward.

6.2. Process Cell Description

This process cell is a highly networked cell capable of producing multiple batches of multiple products at the same time. Cleaning is provided by five source systems. Cleaning is normally done while production is running on other portions of the process cell. The batch management system has approximately 180 product recipes and 10 clean-in-place recipes. All recipes are fully class-based. A clean-in-place recipe for a cook kettle unit applies to all cook kettles. The product of this cell is a cooked, high-acid food product.

6.3. Control System Architecture

A Rockwell Automation architecture with FactoryTalk Batch™, Factory Talk View™ SE and ControlLogix™ programmable automation controllers was used as the system platform.

The batch management system was implemented as a single, stand-alone, server, without redundancy. FactoryTalk Batch eProcedure™ was used for electronic work instructions and to graphically represent recipe status on the HMI computers. Recipe based operator prompting was done using PanelView™ operator terminals located near certain units. FactoryTalk Batch Material Track™, MS SQL Server™ and MS SQL Server Reporting Services were not configured or used in the initial implementation.

The FactoryTalk View SE installation was on two failover servers. A number of intelligent objects were developed for FactoryTalk View SE in order that displays would be flexible and offer necessary information to the operators. Some intelligent objects retrieve information directly from the FactoryTalk Batch API, allowing display of work instructions on appropriate graphics. Others communicate with the table structure in the ControlLogix programmable automation controllers.

The system utilizes eight ControlLogix programmable automation controllers mostly executing programming for the control modules. The number of programmable automation controllers used in this system was a function of the amount of I/O, distribution of the I/O and segmentation of the process.

6.4. Software Architecture

6.4.1. Configuration Tool

The configuration tool provides two primary functions. First, it assists in building the process cell data model. Second, it downloads model changes to the process controller and the several equipment controllers. A robust configuration tool provides many other functions to assist in accurately building a system configuration.

Importantly, the configuration tool need not be online after downloading the model to the programmable automation controller(s).

6.4.2. Programmable Automation Controller Software

The programmable automation controller software was developed using several of the IEC-61131 languages available in the ControlLogix programmable automation controller. All programmable automation controller equipment control modules include low-level simulation (configurable through the configuration tool) allowing the control engineer to thoroughly test configurations prior to deployment.

6.4.3. HMI Application

HMI backgrounds were developed as typical process diagrams with typical annotations.

Representations of units feature a unit pop-up which gives the operator access to the functions that unit can perform and the equipment that unit controls (equipment modules and control modules). A single unit pop-up global object is used for all units. The specific contents of each unit pop-up are defined in the

process cell model. Operators can fully drill in to control a certain control module without changing to a different background display.

6.5. Schedule

The initial 12 units of the example project started product trials 11 weeks from the integrator's notification of intent to purchase. Water trials lasted only one week due to system flexibility and prior testing using built-in simulation. Production trials proceeded for four weeks. The additional units came online in phases during the subsequent 10 weeks without interrupting production on the first units.

6.6. Cost Benefits

Engineering manpower to develop the system was less than expected, offering the end customer an initial cost savings of approximately $250,000.

ECS provided FactoryTalk Batch as the batch management system software, even though the specification did not require a batch management system. In fact, the end user's historically preferred system integrator bid a higher price based on using that integrator's custom tools, which had been provided for other customer facilities.

Commissioning manpower and schedule was benchmarked to be 70% to 80% less than the end user's experience commissioning systems of similar complexity. This provided not only direct labor savings but, more importantly, quicker commissioning of equipment.

Advanced features reduced both maintenance and equipment cleaning time per week. The owner measured reduced direct labor cost and increased available capacity for product. Savings and opportunity value totaled nearly $2M annually beyond customer anticipated return on investment.

7. 30-Unit, Cooked, Dairy-based, Process Cell

7.1. Interest

This process cell is of interest because it produces a dairy based product, making clean-in-place critical. Cleaning was implemented using batch management system recipes. The end user was able to achieve additional production by adjusting recipe parameters for efficient, effective cleaning.

7.2. Process Cell Description

The process cell is a highly networked cell capable of producing multiple batches of multiple products at the same time. Cleaning is provided by three source systems. Cleaning is normally done while production is running on other portions of the process cell. The batch management system has approximately 50 product recipes and 6 clean-in-place recipes. All recipes are fully class-based. A clean-in-place recipe for a cook kettle unit applies to all cook kettles. The product of this cell is a cooked, dairy based food product.

7.3. Control System Architecture

A Rockwell Automation architecture with FactoryTalk Batch™, FactoryTalk View™ SE and ControlLogix™ programmable automation controllers was used as the system platform.

The batch management system was implemented as a single, stand-alone, server, without redundancy. FactoryTalk Batch eProcedure™ was used for work instructions and to graphically represent recipe status on the HMI computers. FactoryTalk Batch Material Track™, MS SQL Server™ and MS SQL Server Reporting Services were not configured or used in the initial implementation.

The FactoryTalk View SE installation was on two failover servers. A number of intelligent objects were developed for FactoryTalk View SE in order that displays would be flexible and offer necessary information to the operators. Some intelligent objects retrieve information directly from the FactoryTalk Batch API, allowing display of work instructions on appropriate graphics. Others communicate with the table structure in the ControlLogix programmable automation controllers.

The system utilizes three ControlLogix programmable automation controllers, one process controller executing programming for the equipment modules and the remainder as equipment controllers executing programming for the control modules.

7.4. Software Architecture

7.4.1. Configuration Tool

The configuration tool provides two primary functions. First, it assists in building the process cell data model. Second, it downloads model changes to the process controller and the several equipment controllers. A robust configuration tool provides many other functions to assist in accurately building a system configuration.

Importantly, the configuration tool need not be online after downloading the model to the programmable automation controller(s).

7.4.2. Programmable Automation Controller Software

This implementation utilized the programmable automation controller software developed for the implementation described above with modifications.

7.4.3. HMI Application

HMI backgrounds were developed as typical process diagrams with typical annotations.

This implementation utilized HMI application objects developed for the implementation described above.

7.5. Schedule

The 30 units of the example project started product trials 24 weeks from the integrator’s notification of intent to purchase. The system was fully simulated prior to startup. Trials were run 2 weeks after initiation of startup.

7.6. Cost Benefits

Many of the same cost benefits were realized as in the above described implementation.

The customer reported that the largest benefit was received from the recipe based clean-in-place system. Plant engineering personnel were able to follow a practice of adjusting clean-in-place recipe parameters, tearing down equipment, swab testing, reassembly and readjustment of recipe parameters. Following this process, clean-in-place time was significantly reduced from that initially suggested by the process design firm. The reduction provided additional manufacturing capacity for the process cell.

8. 30-Unit Industrial Flavorings Cell

8.1. Interest

This process cell is of interest because it is implemented using model based control without a batch management system package. Operators use configurable controls to initiate process actions. Actions by initiate a single phase or a unit procedure.

8.2. Process Cell Description

This process cell is a simple cell which blends multiple ingredients into industrial flavorings which are sold to be added to various consumer products. Cleaning is manual. First phase implementation does not include a BMS, the operators will execute equipment phases from the HMI to add ingredient quantities according to a paper recipe. The second phase includes the implementation of the batch management system and inventory management system. The third phase integrates the batch management system with the enterprise resource planning system to complete the fully automated system.

8.3. Control System Architecture

A Rockwell Automation architecture with FactoryTalk View™ SE and ControlLogix™ programmable automation controllers was used as the system platform. A Stratus Technologies ftServer® redundant hardware fault-tolerant server hosts all of the virtualized computers.

The FactoryTalk View SE installation is virtualized. A number of intelligent objects were developed for FactoryTalk View SE in order that displays would be flexible and offer necessary information to the operators.

One ControlLogix™ programmable automation controller is used as the equipment controller. A SoftLogix™ controller serves as the process controller. The SoftLogix controller is installed on a virtual PC.

When installed, the batch management system will also be virtualized on the redundant hardware server.

8.4. Software Architecture

8.4.1. Programmable Automation Controller Software

The programmable automation controller software was developed using several of the IEC-61131 languages available in the ControlLogix programmable automation controller. The software of this implementation is an enhanced version of the programmable automation controller system software provided in the first two implementations. Tested programming modules have been refined and packaged as Add-On Instructions in order to facilitate configuration management and to segregate standard system programming from special end-user developed programming.

All programmable automation controller programming modules include low-level simulation (configurable through the configuration tool) allowing the control engineer to thoroughly test configurations prior to deployment.

8.4.2. HMI Application

HMI backgrounds were developed as typical process diagrams with typical annotations.

Representations of units feature a unit pop-up which gives the operator access to the functions that unit can perform and the equipment that unit controls (phases, equipment modules, and control modules). A single unit pop-up global object is used for all units. The specific contents of each unit pop-up are defined in the data model. Operators can fully drill in to control a certain control module without changing to a different background display.

8.5. Schedule

The 30 units of the example project will start product trials 14 weeks from the integrator's notification of intent to purchase. The system is being fully simulated prior to startup.

8.6. Cost Benefits

The cost benefits have not yet been measured.

9. Conclusion

It is both possible and beneficial to implement model based control in a modern programmable automation controller for the purpose of providing a basic process control system. As programming objects are developed to support additional entity classes, system building efficiency will increase even further. Models can be built successfully by any process-knowledgeable individual; a programmer is not required.

The example projects described briefly herein were very successful in terms of short schedule, low initial cost and low ownership cost. Advanced features, many due to exploiting entity relationships, provided additional value.

Deploying a model based control system in a programmable automation controller has proven benefits and will become a mainstream approach to standard based control system integration.