Presented at the
WBF
European Conference
Mechelen, Belgium
13-15 November 2006

PO Box 12277
67 Alexander Drive
Research Triangle Park, NC 27709
USA
+1.919.314.3970
Fax: +1.919.314.3971
E-mail: info@wbf.org
www.wbf.org

Using Basic Control to Make Recipes Simple

Francis Lovering
Automation Consultant
ControlDraw Ltd
Southsea,
England
+44 2392 716857
+44 2392 716858
fl@controldraw.co.uk

KEY WORDS

Recipe, Control, Basic control, Engineering

Background

This is about the nuts and bolts – the control systems that a plant operator works with. About the Process Control, and how to improve it.

Over the last few years I have been seen many Batch Control Systems during and after their development. More than just seeing them, I have delved into the depths of them, in particular the batch procedures and the physical models.
This has been done with a number of large projects implemented by major Process Automation Suppliers in Europe and the USA.
Most reviews were done by looking into Detailed Specification Models produced by the suppliers, but some were reviewed by code inspection and/or by reviewing Office Specification documents. Confidentially prevents listing them here, but they have been mostly in Pharmaceuticals.

These systems have been both DCS and PLC based, but this paper is not about DCS versus PLC. The one thing most of the large systems had in common was the use of some kind of batch manager. An 'S88 Compliant' one (not always the same one,) linked to the controllers via Phase Logic Interfaces.

This paper is about how the systems are deployed, that is the way the applications have been built, the capabilities of standard products, although that may also be relevant.

I have looked at what these applications have implemented as phases, operations etc in the Batch Managers and at the equipment and control module levels in the controllers - and at the Control recipe procedure/equipment control separation.

Many times I have found that the suppliers seem to have made the procedures much more complex than seems necessary, especially for Recipe Designers. One reason seems to be that they have failed to provide a physical model that helps to make recipe development easy.

That is the area that this paper covers, with emphasis on how the physical level can be used to improve the situation.

First, what does the S88.01 standard say?

S88.01 standard

Here are a couple of points from the standard.
The standard is clear about many things. In the Introduction it says

"….this standard provides a standard terminology and a consistent set of concepts and models for batch manufacturing plants and batch control that will improve communications between all parties involved; and that will

- reduce the user's time to reach full production levels for new products;
- enable vendors to supply appropriate tools for implementing batch control;
- enable users to better identify their needs;
- **make recipe development straightforward enough to be accomplished without the services of a control systems engineer;**
- reduce the cost of automating batch processes; and
- reduce life-cycle engineering efforts.

It is not the intent of this standard to

- suggest that there is only one way to implement or apply batch control;
- force users to abandon their current way of dealing with their batch processes; or
- restrict development in the area of batch control."

Now, that is clear enough.
Note the use of the words May – the standard itself points out that is it does not provide a fixed set of rules or a software design.
It works mainly to promote a level of understanding and a common verbal language.

It hopes that this "will make recipe development straightforward enough to be accomplished without the services of a control systems engineer".

Has it succeeded?
Well, the question is more about whether practitioners have succeeded – my impression that many have not!

'Making recipes simple' is more than just making it easy to design the Recipes, it is also about making them easy to operate. In S88 terms not just making the Master Recipe simpler, but the Control Recipe too.

S88 also provides definitions of basic and procedural control, a couple of key points of which are:

"**3.4 basic control**: Control that is dedicated to establishing and maintaining a specific state of equipment or process condition.
NOTE – Basic control may include regulatory control, interlocking, monitoring, exception handling, and discrete or sequential control."

3.18 equipment phase: A phase that is part of equipment control.

"6.2.2 Define general recipe procedural elements
The define general recipe procedural elements control function creates, maintains and makes available for subsequent use, the procedural elements that are used as building blocks in general recipe and site recipe procedures…….importantly, modular process actions, process operations, process stages, and/or complete procedures that are frequently reused tend to make recipe transformations at lower levels much easier to accomplish and recipes more consistent."

3.34 phase: The lowest level of procedural element in the procedural control model.

Now, these lower level procedural elements (typically recipe phases) should be the things that the recipe designer has to build at their disposal with which to build each general recipe.
And one other point from S88. Both Procedural and Equipment phases exist.

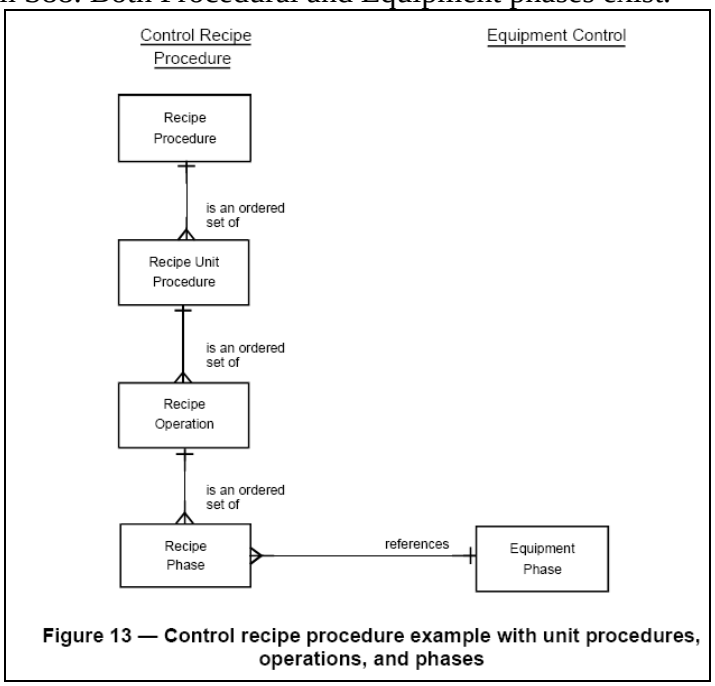


Figure 13 — Control recipe procedure example with unit procedures, operations, and phases

What happens with Typical Implementations

Low Level Equipment Phases and complex Operations

I have found that many of the systems have very complex Operations made from numerous calls to low level equipment phases. Many of these phases, (the building blocks) are small in scope and do a very minor part of the work.
It should be mentioned that this even applies to what I have seen in some (older) WBF S88 Tutorials, and on the Web where major companies explanations of S88 promote the same.

For example entire Equipment Modules and phases are often dedicated to

- Starting or stopping an Agitator.
- Jacket temperature control.
- Prompting an operator. (Message phase)

Should these really be phases?

- They do not Perform a Process action - “run a minor process activity”
- The functions they run can easily be handling in basic control, by passing some parameters and making a simple state request.

Message Phases

Many systems have a messaging phase, with the message text (or an index to a list of message strings) passed to it as a recipe parameter. Sometimes this is the only way of prompting an operator.
This means that in order to have a message ‘mid phase’, the phase with the message in must be split at the point of the message.
I fail to believe that this is the only way that modern systems can handle this.

Tiny Phases

There seems to be a pattern whereby some S88 Batch implementers reduce the phases to the lowest level of modularity so that they can be re-used even within the same operation. Sometimes this is taken to extremes with tiny phases that hardly do anything, perhaps just set one control module. I have even seen a ‘phase’ that sets one Control Module - and with the Control Module that it is to set passed as a parameter.
This ‘phase’ was then used repeatedly to set the all the Control Modules in the same equipment module.
That really is not an equipment phase.
I call it over Modularisation.

Thee consequence is of course that the Operations become much more complex.
Let’s first state

A phase is not the same as a subroutine!

A single step in a phase can invoke quite a complex action in basic control, possibly including sequential control etc. There is no need for Equipment Phases to be so tiny.

Exception Handling Phases

Exception Handling is not generally something that carries out a process action. Almost always it is concerned with physical equipment failure.
Yet it is common to find that Exception Handling is implemented in phase logic.
Often this involves having Re-Entry points in the equipment phases so that a (non exception) phase can continue after an Exception.
Is this really necessary?
It certainly will not make recipe design easy for the chemist if Recipe Design involves designing the Exception Handling.
And what really interests the chemist about Exception Handling anyway? Imagine a Laboratory Recipe. The chemist writes it as a series of steps that a trained technician can run – the technician knows how to do the basic steps, and knows what to do when things do not go exactly right. The trained technician represents good basic control.

Parameters

Search S88.01 for the term parameter and there are in fact 29 references. These all refer to Process parameters.
3.46 Process Parameter: Information that is needed to manufacture a material but does not fall into the classification of process input or process output.
NOTE – Examples of process parameter information are temperature, pressure, and time.
So in this context, parameters are indeed what the chemist should be familiar with.
(I am talking about the parameters in the procedural model – in the batch managers)
Many times I have found that there are many more parameters in the Recipe Phases that just those that the chemist should expect, including many that to me are not just Process parameters.
Now, an Equipment phase, being specifically associated with physical equipment, may also need to have parameters related to the equipment, but these are of no interest to the chemist.
But in many of the implementation there was no way to distinguish between recipe and equipment parameters in the implementations. Consequently all the parameters end up in the recipe.

Equipment Allocation

In S88, Process management is responsible for the identification and reservation of equipment such as Units and Common Resources.
According to S88, the Master Recipe Procedure should be independent from the equipment. That is the Master Recipe defines what equipment is required, but not the specific equipment. And ideally for simplicity of recipe design these Equipment Requirements should be high level things like Units, not a list of all the required control modules for example. Furthermore the Required Equipment should be by type (or 'Class') rather than referring to specific physical entities.

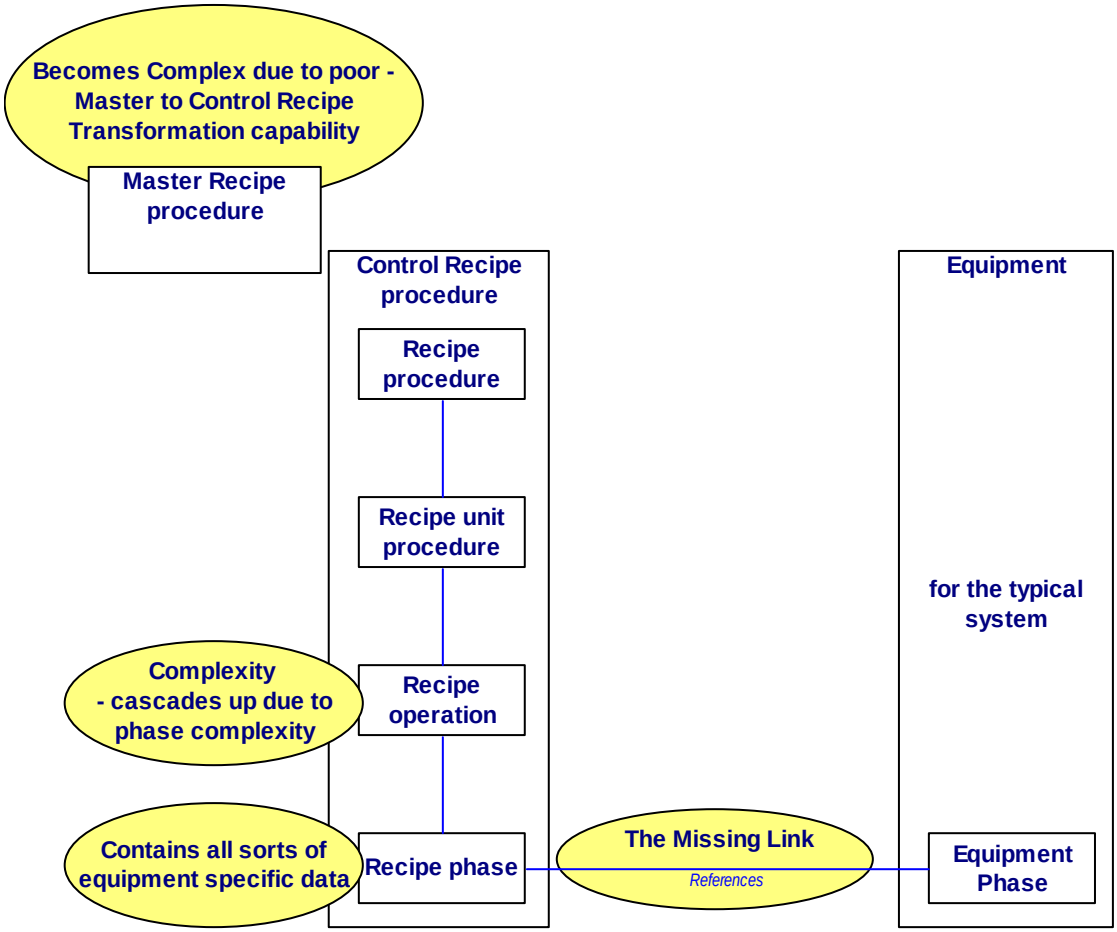
Whilst it appears from my understanding of them that the Recipe Managers are able to work with Classes of equipment it seems that this ideal falls down because there is nothing other than phases to deal with the equipment selections , especially in plant with complex routing.

I have found is that it common for Procedural phases to require 'parameters' in order to determine what equipment it is to run.

And many implementations actually do the Acquiring and Releasing with phases. Now, for one, I don't see why a resource should not be acquired and released within a phase, just as steps. But more importantly in S88 this is a function of Process Management – which does not mean something an individual control recipe does.

These issues alone make the Recipe Procedures much more complex. And I also believe that there are much better ways of handling this.

Summarizing



How can we improve?

Ideally the chemist should be able to construct recipe procedures without having to worry about the physical equipment. And I also think that the chemist should be able to design the operations, by using recipe phases that were provided with the system and without having to worry about the corresponding equipment phases.

The key to this is when deciding what recipe phases provide to first ask what are the minor process actions that a Chemist will recognise.

Add stuff is a real phase

Start or stop an agitator is not in my opinion, it is just something you can do in a step in a phase.

Then ask what are the parameters that the Chemist will need to set? These should be related to the product, rather than the equipment.

Improving Basic Control

Control Modules

Control Modules are the foundations of a process Control System, as S88 says:

"The lowest level grouping of equipment in the physical model that can carry out basic control.
NOTE - This term applies to both the physical equipment and the equipment entity.
A control module is typically a collection of sensors, actuators, other control modules, and associated processing equipment that, from the point of view of control, is operated as a single entity. A control module can also be made up of other control modules.

As it says, a control module can also be made up of other control modules. There seems however to be a tendency for implementers to assume that this then makes an equipment module. It does not.
Control modules can also provide a basic Manual level of operation which is essential for any plant.

Typically, Control modules have many parameters, but only a few are really Process parameters that would interest a chemist.

For example, a Valve Control Module would include such settings as the maximum times it take to open or close, a motor may have a maximum number of starts per hour, a PID controller will have many tuning settings.
Generally the values of these parameters relate to the physical equipment and do not need to change if a different recipe is being run. There will be exceptions (for example PID tuning may change for different materials) but these are not frequent.
So a key point in the analysis is to decide which parameters in the module are fixed and which may require setting from the recipe.

A Control Module can also be a quite smart.
For example an Agitator Control Module can by itself respond to changes in the level in the vessel, slowing down and stopping as level drops, run energy saving algorithms (cycling on and off). A collection of parameters can be used to make the module configurable to meet the requirements of the Recipes – all the Recipe has to do is set up these parameters. You do not need a whole phase to control an agitator, Just a simple command from a phase step is all that is needed.

A Profile Generator is another Control Module level object that can reduce the procedural complexity to a set of parameters.
This is a control module that can generate a sequence of values, such ramp ups, dwells and so on. Typically these provide the Set Points for some types of processing such a temperature controls. Again these just require a collection of parameters from the Recipe and a Trigger.

Complex loops
Consider the typical examples that show two materials being fed by parallel phases. By providing Ratio Control, Control modules are very capable of handling this.
I suggest here that the batch community learns a little from the Continuous people, for whom for example Ratio Control is just one of many types of Complex loops that they know well.

Designing Control Modules

Some Control Modules just monitor a physical equipment entity and provide data about its state. Some actually direct their portion of the process.
Designing this level well can do a lot toward making the system 'friendly'.

Control Modules are State Engines, they work to set the objects that they manage to a fixed state in response to a 'state command'. Even Monitor Only CM's normally have a state command that tells it what to look for in the current context. At the minimum this is Enabled or Disabled.

At this point it is worth mentioning another problem with many Phases –the ones that set the 'standard 'alarm limits. This is not often a good idea, since the Alarm limits for a Control Module should I think be those that apply physically, regardless of the process, and it is easy to create process specific checks at a higher level in the physical model, eg in an Equipment Module.

Exception handling can also be contained within a Control Module, in fact a major part of even a simple CM's functions is to monitor for exceptions. Specifically those that can be identified by physical measurements, such as valve limit switches that show that the valve is not in the state that the command to it has requested.
In addition higher level Exception handling can still be contained in Basic Control as discussed later in this paper.

Equipment phases and Equipment Modules

*"**3.16 equipment module:** A functional group of equipment that can carry out a finite number of specific minor processing activities.*
NOTES
1 An equipment module is typically centered around a piece of process equipment (a weigh tank, a process heater, a scrubber, etc.). This term applies to both the physical equipment and the equipment entity.
2 Examples of minor process activities are dosing and weighing.

5.2.2.3.2 Procedural control in equipment modules
Equipment modules may execute equipment phases, but they do not have the capability of executing higher level procedural elements.

S88.01 clearly states that the Equipment Module level is optional, only the Unit is mandatory.
In 'Typical' Equipment Module Centric designs, the units are split into Equipment modules which run phases, and so the operations must address each equipment module. Typically this is done either by:

A parallel structure where there is a parallel path in the operation for each Equipment module in the Unit.

A sequential structure whereby some Equipment phases are designed to simply set a Control Module state and leave it Running so the Operation works by running in turn a number of short lived Phases.

However the Standard is clear (as shown in Figure 7) that Phases can equally run in the Unit.

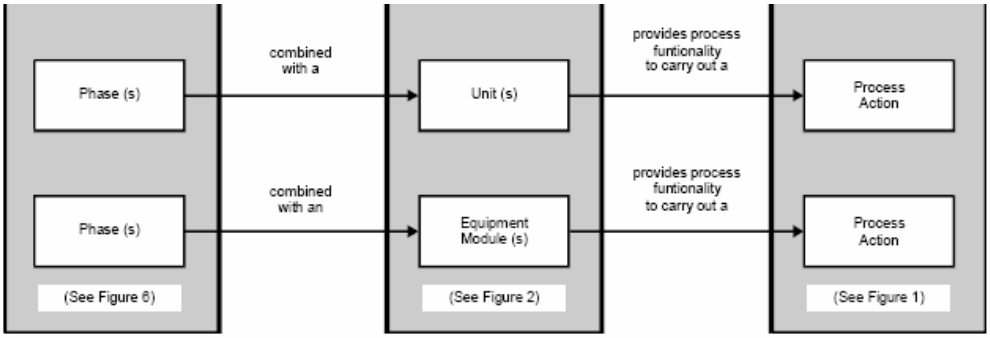


Figure 7 — Procedural control/equipment mapping to achieve process functionality

I have however also seen many implementations where the "Equipment Module" level, although present does not appear to conform to the S88 definition. [1]
Much like the typical implementations they are applied to such equipment as a heater, an agitator etc, in other words a functional group of equipment, and they are made up from control modules.
However, these Equipment Modules do not appear to conform to S88.01 because they do not run their own phases. Although the word 'may' in 5.2.2.3.2 suggests that they do not have to, it seems to be conventional practise that they must.
So, let's for now call them Phase-Less Equipment modules. They are told what to do by a phase running at the Unit level.
Of course only one phase can run in physical entity (whether Unit or Equipment module so this effectively means only one phase running at one time in the Unit. All the multiple activities in the unit are still possible but they are running in the physical model not the procedural.

Now, suppose the Equipment modules do not have such phases, but the Unit does.

A Unit phase could command all the Control Modules in the same unit at the same time.

That is highly feasible and is in fact the other method that I have seen in use. And in projects that seem to have been less troublesome!
It requires that the Basic Control is smarter than in the Classic model.
In what follows, to keep with the convention that Phases run in Equipment modules we will just have one Equipment module in the unit, able to run a number of phases.

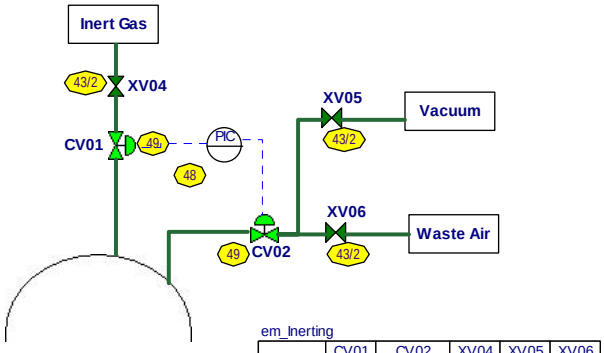
Designing Equipment Modules and Phases

Equipment modules are made up of control modules. Control Modules are of course state machines. And the Basic control of an Equipment module can be too.

Equipment phases have to set the states, set points etc of the control modules in the EM and monitor process parameter for events. Typically in many systems the phase steps address individually each Control Module in the EM.
It is often useful and easier to follow by having Equipment States,

[1] There is some debate about this as with most things S88

The Equipment State Matrix shown below groups the Control Modules so that all that the phase has to do is set the equipment state.
So in the example (for one Equipment Module) below the 5 Control Modules can be set in one phase step simple by setting the equipment module to one of the 3 states



em_Inerting

	CV01	CV02	XV04	XV05	XV06
Idle	Close	Close	Close	Close	Close
Inerting	PID	PID	Open	Open	Close
Venting	Close	Full Open	Close	Close	Open

Now, this can be extended in the Simple Single Equipment module design by having a Unit State Matrix

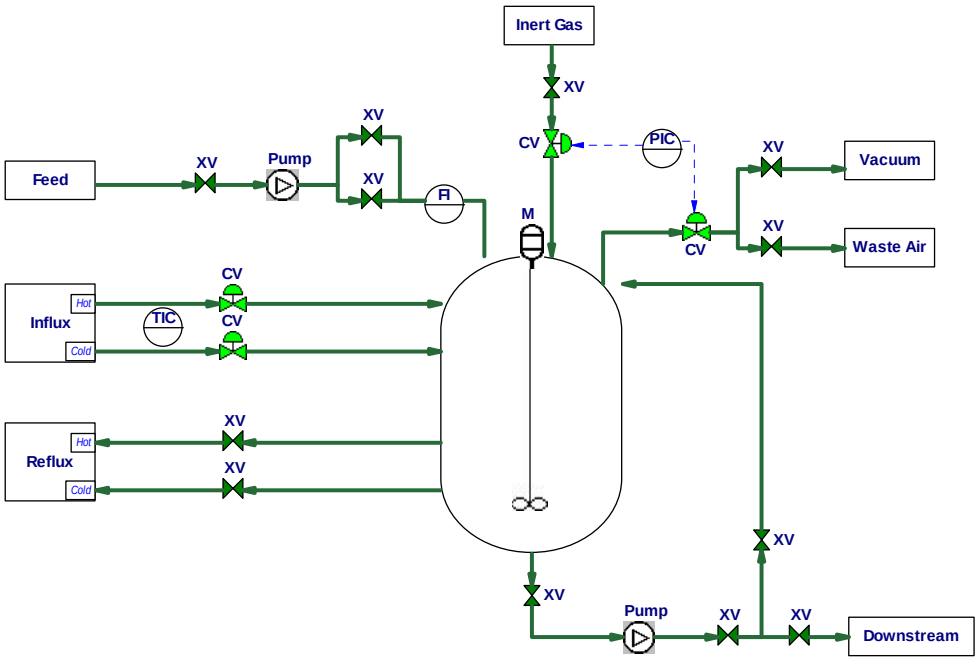
S88 Tutorial Unit

	eqm_Dosing	eqm_Inerting	eqm_Pumping off and Recycling	eqm_Stirring	eqm_Tempering
Idle	Idle	Idle	Idle	Idle	Idle
Fast Dosing	Fast Dosing	Venting	Idle	Run	Idle
Slow Dosing	Slow Dosing	Venting	Idle	Idle	Run Profile
Inerting	Idle	Inerting	Recycling	Run	Run Profile
HeatUp	Idle	Inerting	Recycling	Run	Full Heat
Tempering	Idle	Inerting	Recycling	Run	Run Profile
Pumping Downstream	Idle	Venting	Pumping Downstream	Idle	Idle
CoolDown	Idle	Idle	Idle	Idle	Full Cool

Unit Equipment Modularisation Example

Now let's compare both approaches, using a typical Unit, in fact one from an old WBF S88 Tutorial.

The original unit



Now lets look at how it might be decomposed into Equipment modules and a Unit Procedure that:

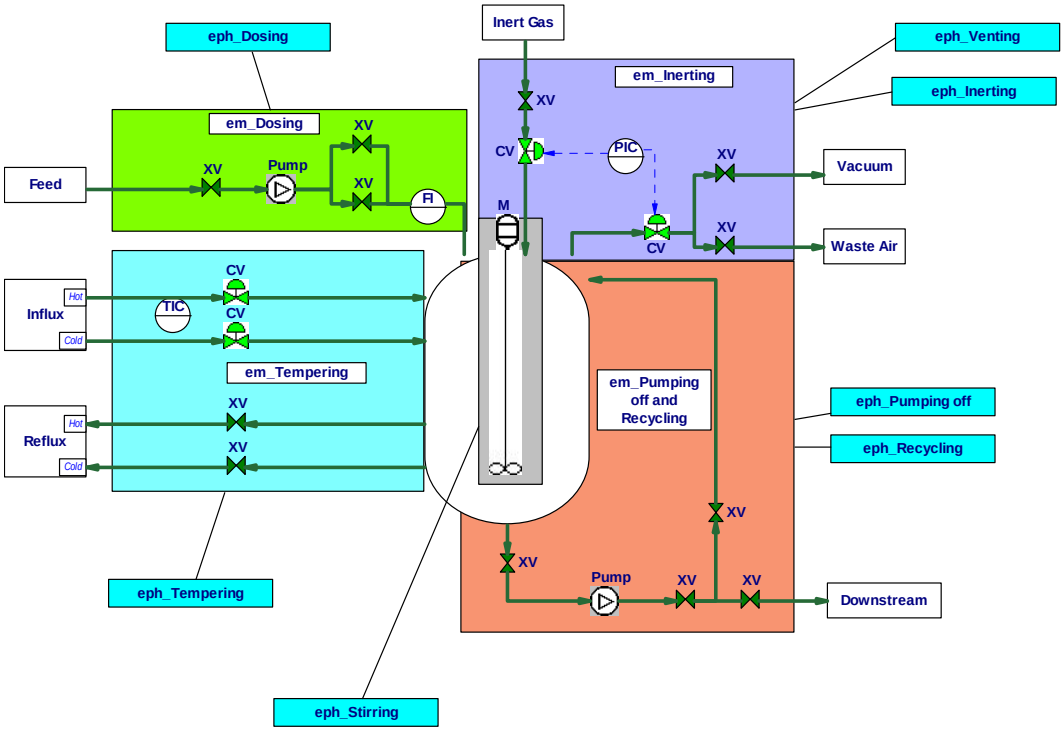
- Begins the batch by adding some Feed (Dosing)
- Inert the batch while stirring
- Continues stirring while subjecting the batch to a temperature profile
- Then Pumps the batch downstream

Two ways of handling this follow, the Classic version and the Simple version.
In the Classic version there are 5 Equipment modules each with their own phases, 7 phases in total.
In the Simple version there is only one Equipment module with 4 phases

- Dosing
- Inerting
- Tempering
- Pumping off

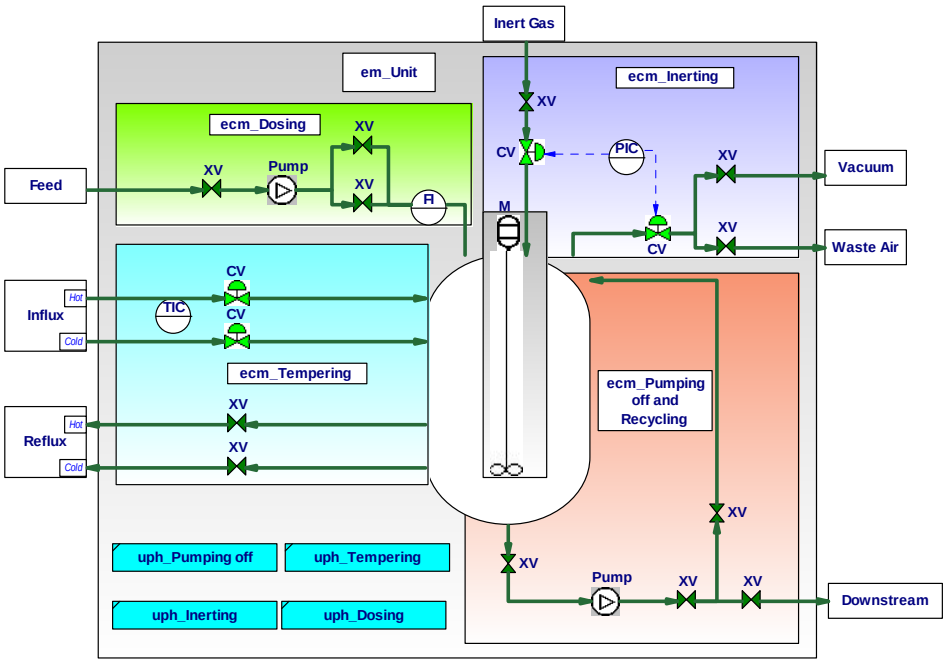
The Typical Equipment module Structure with it's phases

(Phases are prefixed eph_ and Equipment modules em_)



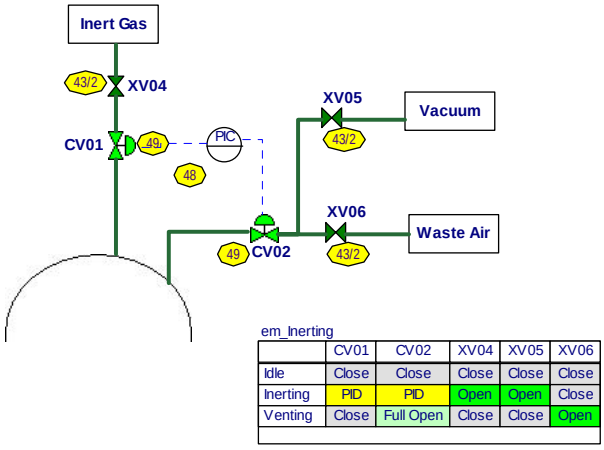
The Simpler Unit, with it's phases

(Phases are prefixed uph_ and the Phase-Less Equipment modules ecm_)



Typical Equipment Module with State Matrix

A state matrix is a means of grouping the required control module states in response to a state request. In the simple example below, the Equipment has three possible states, Idle, Inerting and Venting, each providing a row in the matrix. The corresponding states of the control modules are shown in the columns

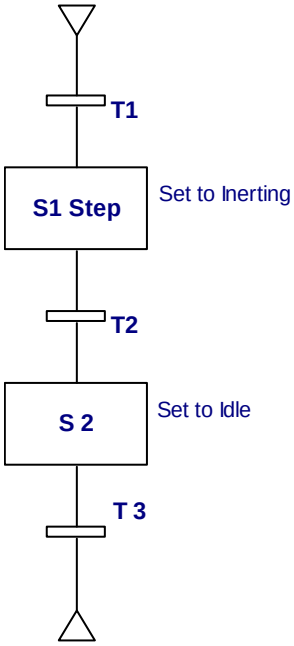


em_Inerting

	CV01	CV02	XV04	XV05	XV06
Idle	Close	Close	Close	Close	Close
Inerting	PID	PID	Open	Open	Close
Venting	Close	Full Open	Close	Close	Open

Equipment phase with Set States

This shows how an Equipment Module level phase can drive the EM into one of it's states. The phase is simplified because it does not have to address each of the CM's in the equipment.



Unit Level State Matrix

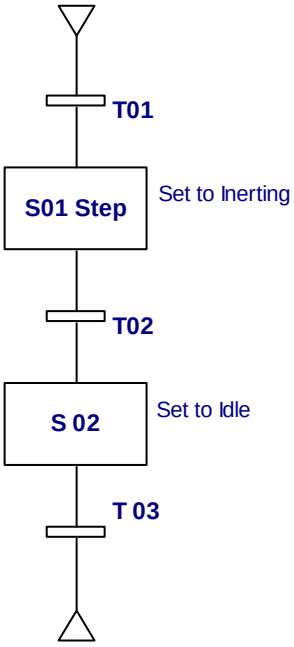
A Unt state matrix is a means of grouping the required control module states in response to a state request. In the simple example below, the Matrix has all the required states of the unit as rows, and the corresponding states of the equipment modules are shown in the columns.

S88 Tutorial Unit

	eqm_Dosing	eqm_Inerting	eqm_Pumping off and Recycling	eqm_Stirring	eqm_Tempering
Idle	Idle	Idle	Idle	Idle	Idle
Fast Dosing	Fast Dosing	Venting	Idle	Run	Idle
Slow Dosing	Slow Dosing	Venting	Idle	Idle	Run Profile
Inerting	Idle	Inerting	Recycling	Run	Run Profile
HeatUp	Idle	Inerting	Recycling	Run	Full Heat
Tempering	Idle	Inerting	Recycling	Run	Run Profile
Pumping Dow nstream	Idle	Venting	Pumping Dow nstream	Idle	Idle
CoolDow n	Idle	Idle	Idle	Idle	Full Cool

Unit phase with Set States

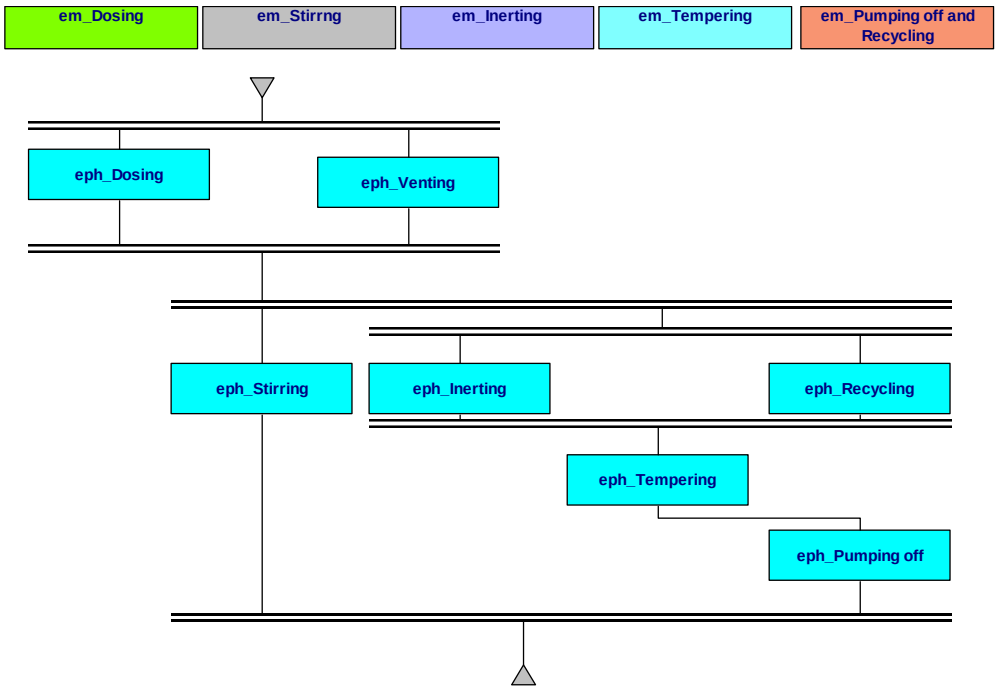
This shows how an Unit Level phase can drive the entire unit into one of it's states.



And the Unit Procedures

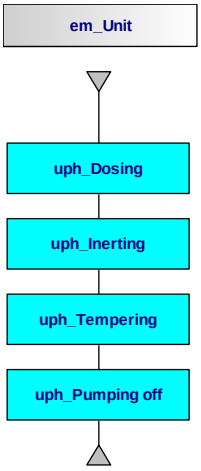
Typical

(Actually this is much simpler then real typical UP's.)



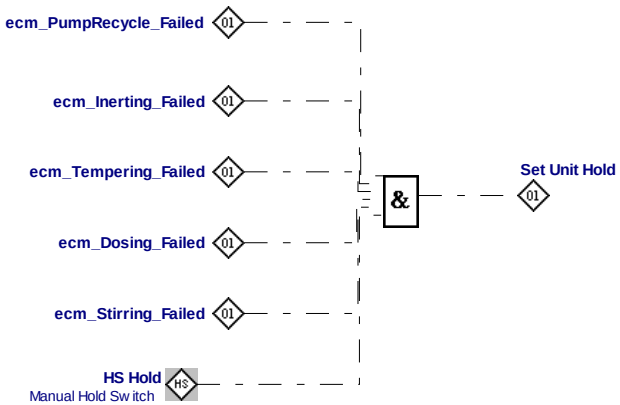
Simpler

This Unit Procedure is much simpler, and chemist friendly



Simplifying Exception handling.

Basic Control has major role here too.
S88 defines events that may trigger exception handling as
— unavailability of raw materials, utilities, or plant equipment when needed;
— product or process problems;
— control equipment malfunction;
— hazardous conditions such as fire or chemical spills.
As far as the systems go, exception handling is nearly always concerned with control equipment malfunction - physical equipment failure that can be handled in the physical model, by basic control. (Product or process problems that do not result from physical failures are rare in well-defined processes.)
Control modules check for physical failures, but you can add much more to an Equipment module or a Unit.
Unit Level Exception Handling Modules can do a lot and need not be that complex.
Typically they monitor the exception states of lower level Control Modules in the unit by aggregating them by, say, OR logic (CM1 has failed or CM2 has failed etc.)



They can respond by driving the equipment to a state – typically one that holds the process in a stable state..

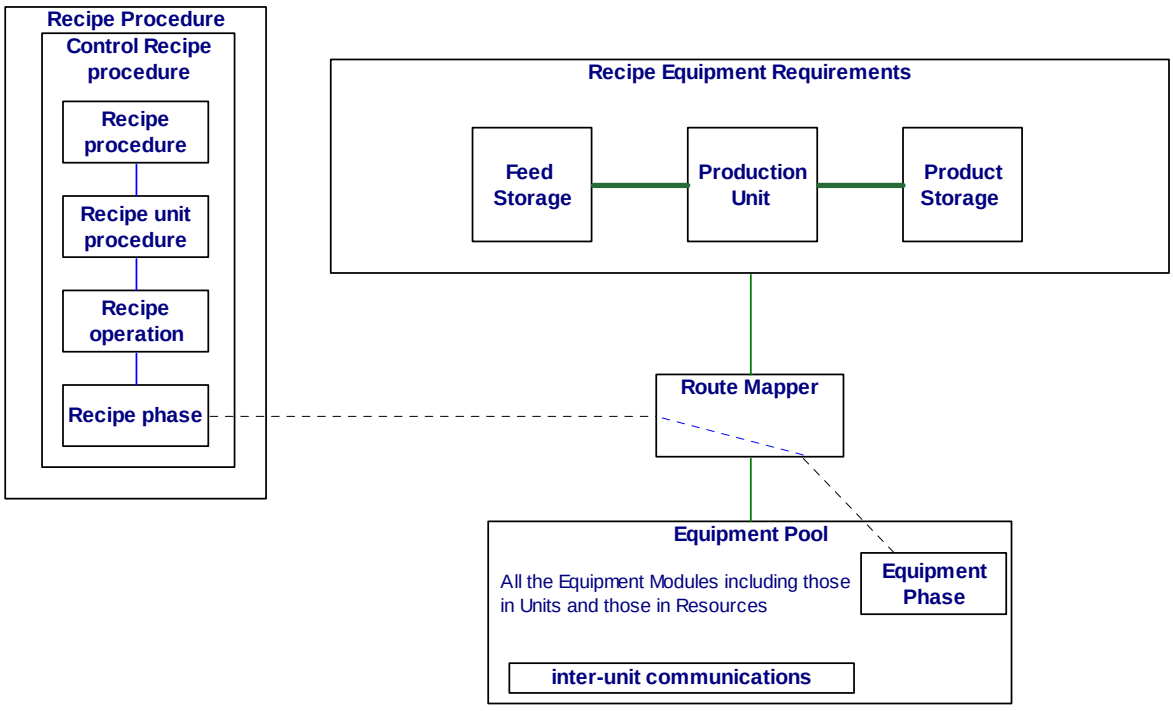
Now, if a phase step is running then it should be doing so until something is achieved. That something will normally be a process event – such as a measurement reaching a value (eg Weight at target,) or perhaps a time elapsing whilst the process is in some state (Mixing at Temperature for x seconds)

Now suppose that a physical failure happens. If Basic Control can respond by putting the process into a hold state then the process event will not happen.
The Phase could just sit still running waiting for the event. (It could even complete if the process event still happens whilst the equipment is reaching the stable state.)
This can remove the need for much of the exception handing that typically makes phase logic complex.

So, designing good Hold Logic in Basic Control can simplify the recipes.

Routing and Mapping Class Based Recipes into the actual equipment

As mentioned above, many systems become highly complex due to the inclusion of equipment selections, routing etc in the phase logic.
I strongly believe that there is a better solution. In essence this means a layer of software that can handle a 'pool' of equipment and direct the phases to whichever specific equipment is allocated. A diagram of how this might be done is provided below.
While this is out of the scope of this paper, it is worth mentioning that there is always a need to establish some connections between the physical entities at the basic control level to handle such things as interlocks on transfer pumps. Such links should not depend on being passed up to and back from the procedural model, but should exist in basic control.



Conclusions

- ***Many Current Practitioners are not succeeding to Make Recipes easy to design***

- ***The problem is not the Standard***

- ***It may be in part the way it is taught***

- ***Improved Basic Control can help***

- ***Designers should improve differentiation between Recipe and Equipment phases***

- ***Suppliers need to produce better solutions for mapping procedural control to the equipment***