Exception Handling – A Practical and Manageable Approach

Alistair Ross
Alba Controls Limited
James Watt Building, Scottish Enterprise Technology Park
East Kilbride, G75 0QD
Scotland
Tel +44 (0) 1355 272594
Fax +44 (0) 1355 272907
E-mail aross@albacontrols.co.uk

KEY WORDS
Exception Handling, Permissive, Interlock, HAZOP

ABSTRACT
This paper is based on the practical experiences implementing batch projects on both new build and control system replacement projects. In particular, a technical and management framework specifically for exception handling is essential to ensure a known, proven, safe and maintainable validated batch system is delivered. The practical challenges are significant, projects following the classic specification lifecycle either fail to deliver the quality of information or the safety assessment tasks are conducted in parallel and therefore not available at the early stages. Working methods must be defined to allow the batch control system to accommodate exceptions with a high degree of transparency and flexibility.
WELCOME TO THE REAL WORLD

This paper discusses from a light hearted and honest perspective the implementation experiences and practical issues that Exception Handling within S88 based systems must overcome.

The project has been awarded and the control and software components chosen… oh and by the way production must have the delivery brought forward delivery to summer instead of fall…..and we don’t like some of the things the existing system does, but we’ll get back to you.

I’m sure these are familiar challenges in a batch specialist’s daily life; indeed Mr. Superbatch and his friends have cracked a lot of the hard stuff; we all talk the same language - S88 speak, we have excellent formal physical and procedural design models, we have libraries of standard and proven Control Modules and Phase Logic Routines which can all be re-applied based on the latest object technologies. Is there nothing that cannot be overcome?

Well, yes there is a small problem - customer expectations rise ever higher, management want to get to market faster, the control system must go in early,…but sir…..oh and we are re-working all the recipes for environmental reasons, we’ll let you know just before commissioning,…but sir…..

All sounds either a bit suicidal or a plum opportunity to get some mileage on change control (I never said that). Okay, I have deliberately gone a touch overboard, just to emphasis a very important point.

“Say what your going to do,… then do it”. I really believe in this, but saying all this at once at the beginning is beyond me! Surely the key to delivering agile manufacturing solutions is doing things in parallel and developing practical working methods to enable successful projects to be delivered on time and to satisfied customers.

Anyway, back to action, the scene is set, a strong implementation team has been assembled comprising operators, engineers, management, chemists, approval has been granted and the race is on. It becomes very clear that the Exception Handling definition is well off the critical plan,… can’t we just add this in later’,….. there are a great many other reasons such a scenario should be considered as follows

- Best/Appropriate people/information is not available at early stages; they have key duties to perform on this or other projects.
- Sometimes, where there is an existing plant, the customer takes the view that the existing plant exception details are valid, but after the design boys and girls roll there sleeves up, a different picture comes out and the only solution is to review the exception logic (this is a time-consuming task)
- Even with the best available knowledge, exception handling is very difficult to get 100% right first time. Every project I have been involved with had a common theme; all the brightest people continue to think up new ideas, new combinations of problems, even after a plant has been commissioned successfully. These contributions need to be managed and utilized in a positive controlled fashion. The easiest way for a plant/business/ to fail is to suppress genuine good ideas or innovation just because it’s not in the specification.
- Time to market ⇒ Project Timescales is so fast that there are significant benefits to actually dovetailing the exceptions in later as a concurrent activity. This saves a lot of time and money!

Well, enough of the scene setting, let’s get a move on.
DEFINE THE CHALLENGE

‘Define, define, define’, rigorous up-front exception requirements analysis and definition is important. However, it is crucial to recognise that the exceptions definitions is open for development, corrections and modifications throughout the complete system life cycle. The control, management and implementation of changes to the exceptions must be considered a key part of the batch project. Lessons must be learned from past projects where change was difficult to implement.

So lets define the problem, and then put together a framework for handling the situation.

For the purposes of this paper, lets define the problem as ‘IN THE OLDEN DAYS’; this fictional example is just really used to highlight some potential pitfalls or desirable features.

<table>
<thead>
<tr>
<th>Olden Days</th>
<th>New / Target Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every Phase was hard coded</td>
<td>Re-useable objects are deployed from our phase libraries. Sometimes the same library call for many phases in the same unit.</td>
</tr>
<tr>
<td>Exception Logic was great because we had some common exceptions at the unit level and the rest in tacked on to the phase logic</td>
<td>Sorry, but we have to get the best out of all these object technologies.</td>
</tr>
<tr>
<td>We can find our way around as its easy to get to phase logic and thus exception logic</td>
<td>Nothings stays the same, some things better, others not so!</td>
</tr>
</tbody>
</table>
DESIGN OBJECTIVES

The following points were considered important in developing sensible Exception Handling implementation policies

- **Keep it Simple**
  Should not bamboozle any of the implementation team with over-complex methods

- **Unit Exception Definition Tables**
  Documentation should not be in itself a means to end. It should not choke the efficient project progress or bury team under a forest of paperwork.

- **Ability to Troubleshoot**
  In our case the target platform was a PLC. Any software solution for exception logic should for life cycle reasons be able to be viewed in simple (ladder) graphical status. Whilst Control Modules and Phase Logic may employ and gain the benefits from object-orientated solutions, we have been unable to fully apply the same principles to exception logic.
  The primary exception logic code should be simple and single instance well documented logic. This simple approach allows easy status checking and evaluation by the poor chap on nightshift several years hence. Of course this approach also pays dividends while implementing ‘change’ because it’s so simple!

- **Don’t be Too Smart for your Good**
  Phase libraries and the inherent re-use of phase logic poses a few problems. For example a routine say CHARGE_1 is used for any charging activity into the unit using the target vessel load cells as the measurement device. The routine may be used several times within the same unit for different charge materials. The exception requirements may differ significantly for each chemical material type e.g. chemical charge must have agitator and temperature below 5oC whereas others require ability to add without stirring without temperature constraints or may be rate of temperature change.

- **User Defines Exception and keeps Ownership**
  We must have a simple method of ensuring that the customer and not the system integrator takes ownership of the exception handling. There must be transparency such that the exception logic can be determined and if possible configured without needing a trained DCS or PLC engineer.
  We really wanted to leverage knowledge from the assembled project team effectively.
  (There always seems to be some particularly complex situation or requirement which is solved with a custom solution. Our experience has indicated that over 95% of exception types are straightforward and can be managed. The demanding exceptions can be accommodated and are no reason to comprise the benefits of this approach.)

- **Validation**
  Our last batch project at design stage was coming in at around 250 phases. The validation task to check the exception logic if it were to reside inside each phase would be huge. Any phase modifications may even require re-validation of exceptions.
  De-coupling exceptions from phase logic has significant benefits and allows modular handling.
CONTROL AND DEFINITION

Alarms should be considered as separate entities to Exceptions. This separation allowed the project to proceed with many tasks in parallel enabling the critical project path to be shortened. Phases, Alarms and exceptions could be tackled to some extent in parallel without creating re-work risk and exposure. This was easy anyway as nowadays the bulk of the alarms are generated from the standard library Control Module routines.

In the past (when Exceptions were called Trips) it was indeed difficult for some projects to do a proper job on the exceptions until the Alarms had been defined and laid out. Maybe this is just a narrow view based on previous project methods, but this time we felt it better to leave the HAZOP free to use ‘English’ descriptions to designate exceptions.

Due to the de-coupling of alarms and exceptions, the exceptions could be worked on from the start using more general terms like, no other inlet open, all devices healthy, all instrumentation healthy, base valve closed.

To accommodate many commonly faced problems we defined two types of Exceptions as follows:

<table>
<thead>
<tr>
<th>Interlock Exception</th>
<th>Permissive Exception</th>
</tr>
</thead>
<tbody>
<tr>
<td>These conditions will cause the phase to become HELD under all conditions.</td>
<td>Along the lines of… ‘I’ve started so I’ll finish’, we have a sub-class of exception where in many cases certain conditions will not permit the phase to start. However, once the phase is running then it is desirable to finish the phase task.</td>
</tr>
</tbody>
</table>

A sample exception definition matrix is shown

<table>
<thead>
<tr>
<th>Deutsch</th>
<th>English</th>
<th>ID</th>
<th>Exception Bit Ref</th>
<th>FAILURE (ID Batch)</th>
<th>Beginning of Section</th>
<th>CHARGE_4715</th>
<th>CHARGE_4641</th>
<th>CHARGE_4941</th>
<th>CHARGE_4941</th>
<th>CHARGE_4941</th>
<th>CHARGE_4715</th>
<th>CHARGE_4820</th>
<th>CHARGE_4715</th>
<th>HEAT</th>
<th>TRANSFER_OUT</th>
<th>Section end of</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSS, SSS, OPH</td>
<td>HSS, SSS, OPH</td>
<td>1</td>
<td>0.0 199</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>anderes Zulaufventil nicht geschlossen</td>
<td>Other Inlet Valve NOT Closed</td>
<td>3</td>
<td>0.2 196</td>
<td>P</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>kein anderer Zulauf aktiv</td>
<td>No other addition phase active ?</td>
<td>4</td>
<td>0.5 194</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Bodenaustausventil nicht geschlossen</td>
<td>BOV Not Closed</td>
<td>5</td>
<td>0.4 192</td>
<td>P</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Element oder Ruckmeldungen - Fehler</td>
<td>Device or Switch Failure</td>
<td>6</td>
<td>0.5 190</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gewichtsmessung defekt</td>
<td>W10 Faulty</td>
<td>8</td>
<td>0.7 186</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Niveau &gt; Maximalwert</td>
<td>W10 &gt; Max</td>
<td>9</td>
<td>1.0 184</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Sollniveau &gt; Maximalwert</td>
<td>Wx.Target &gt; Max</td>
<td>10</td>
<td>1.1 182</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Niveau &lt; Minimalwert</td>
<td>W10 &lt; Min</td>
<td>11</td>
<td>1.2 180</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Niveau nicht stabil</td>
<td>Wx not steady after X mins</td>
<td>12</td>
<td>1.3 178</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zuwenig geliefert</td>
<td>Under-charge</td>
<td>13</td>
<td>1.4 176</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>zuviel geliefert</td>
<td>Over-charge</td>
<td>14</td>
<td>1.5 174</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Minimaldurchfluss</td>
<td>Low Flow (WX)</td>
<td>15</td>
<td>1.6 172</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Maximaldurchfluss</td>
<td>High Flow (WX)</td>
<td>16</td>
<td>1.7 170</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Zuliefer - Anlagen</td>
<td>Any Remote Alarm</td>
<td>17</td>
<td>2.0 168</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T10 defekt</td>
<td>T10 Faulty</td>
<td>19</td>
<td>2.2 164</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimaltemperatur</td>
<td>Low Temperature</td>
<td>20</td>
<td>2.3 162</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximalltemperatur 45°C</td>
<td>High Temperature</td>
<td>21</td>
<td>2.4 160</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimale Temperaturänderung</td>
<td>Low Temp rate increase</td>
<td>22</td>
<td>2.5 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rührer defekt beim Startbefehl</td>
<td>Agit Fault (if command to run)</td>
<td>23</td>
<td>2.6 156</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rührer aus</td>
<td>Agit.OFF</td>
<td>24</td>
<td>2.7 154</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rührer Drehzahl zu tief</td>
<td>AgitLOW (or single on)</td>
<td>25</td>
<td>3.0 152</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wäscher in Betrieb</td>
<td>Scrubber/Extraction Running OK</td>
<td>26</td>
<td>3.1 150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transfer Pumpe Fehler</td>
<td>Transfer Pump Fail</td>
<td>27</td>
<td>3.2 148</td>
<td>I</td>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOLUTION PROPOSED

As the wise old sage (or any veteran control engineer) knows

‘It is easy to write complex code, a truly simple solution requires clever design.’

Our solution was applied to different PLC based systems, but could easily be applied with DCS or ‘hybrid’ systems.

- Exception Logic Resident at Unit Level – Not within Phase Logic
  Ensure de-coupling between generic phase logic and phase instance exception logic. The reason was simple we liked our library blocks and could see difficulties arising as each instance could indeed bring new exception functionality which would increase the complexity of the library block.

- Unit Exception logic was based on using single instance logic
  Guaranteed ability to get meaningful graphical status.
  Flexibility – means custom differences or special code can be easily added

- Phase Exception Triggers Defined in Exception tables, not hard-coded
  A structured approach was adopted
  A modular approach was adopted
  Validation tools were available for independent checking
Alarms and Exceptions were handled as BOOLEAN, which made things easy to understand and handle in our target environment. Standard Library routines could then be used to translate the BOOLEAN states into the INTEGER Phase Failure Registers needed by the batch engine. This approach avoided lengthy or error prone repetitive code being used.

In summary, after the exception logic was constructed in the simplest way, the remaining exercise was really a configuration task and not a programming task. Therefore the setting up and checking could be undertaking by any member of the implementation team.

**ENHANCEMENTS, NICE TOUCHES AND SHOWING OFF**

Once the software team and the implementation squad are clear about the objectives and working together it is amazing the number of minor embellishments and aids which appear as follows:

- HMI screens to display actual exception definition tables
- Spreadsheets used to define Exceptions and using minor applications comparison checks could be run on-line against the PLC setup. Colour was then used to highlight differences.
- Since our PLC definition files were protected and read-only we never provided any on-line HMI level configuration features. However, we could easily create PLC data source files for compilation and manually download. This reduced any implementation errors to near zero and brought change control costs to a negligible level.
- We have issued bi-lingual exception logic matrix sheets and left our customer to debate and create the necessary information. Nowadays remote group working using Internet tools and technologies increases project efficiencies.

What I wish were different (or a little better)…

- Would be easier if Batch Packages allowed modification and editing of Phase Failure Enumeration lists without insisting on stopping the plant to re-load the equipment configuration, which is where the enumeration lists are defined.
- Would be nicer if Batch Packages could organise Phase Failure Enumeration Lists better with groups/sub-groups instead of single list, thereby enabling clear and improved use of better exception numbering conventions across units.
- Batch Packages recognised the concept of Permissive Phase Failures separate from Interlocks
CONCLUSIONS

Well, the occasional PLC programmer still has the odd grumble that they have now to look at three places when troubleshooting some situations i.e. Phase Logic, Exception Logic and a HMI Page/Configuration block.

However, we reckon there was

✓ Substantial cost savings associated with project management, change control and re-work efforts.
✓ The ‘impossible’ timescales were achieved
✓ The end-user’s team retains high visibility ownership of the exception definitions.
✓ The on-line visibility of the unit and phase exceptions enables all Plant users to have a valuable reference and training aid. Result is better and safer plant operation.
✓ Life-cycle costs are good….
✓ A well understood system allows the plant to be further developed and properly maintained.

The only sure thing is in life is that things will ‘Change’. The ability to implement change is a strong factor in determining business success. In our batch plant some of the changes could be: a new business opportunity/product, chemist/operator comes up with new products, processes, ideas to improve capacity, safety or quality. The control system enabled ‘change’ in an easy, safe and confident manner. Looking back it seems easy…

….. but that’s the beauty of a successful S88 designed system.