Estruturação usando a norma IEC 61131-3: 7 Passos para o Sucesso

INTRODUÇÃO

Como esperado para os modernos ambientes de desenvolvimento, a norma IEC 61131-3 inclui modernas ferramentas para estruturação de software. As partes essenciais para isto são a linguagem SFC (Sequenciamento Gráfico de Funções) e os Blocos Funcionais (*Function Blocks*) definidos pelo usuário (derivados).
Ambas proporcionam uma excelente forma para decomposição da aplicação de controle em unidades facilmente gerenciáveis. Estas unidades são de fácil utilização e entendimento por diferentes usuários de diferentes formações. As unidades também proporcionam que se estabeleça o elo faltante entre os engenheiros de sistemas, desenvolvedores de software e as equipes de implantação, operação e manutenção. Além disso, elas funcionam como uma importante ferramenta de representação e comunicação para estes grupos de diferentes formações e responsabilidades. Neste sentido, equipes multidisciplinares podem cooperar entre si nos grandes desenvolvimentos para aplicações de controle cada vez mais comuns atualmente, gerando programas mais fáceis de se entender e reutilizar, proporcionando também uma melhor separação das responsabilidades dos diversos usuários, programadores e equipes de implantação e manutenção.

PROMOVENDO A BASE PARA A ESTRUTURAÇÃO

As vantagens da estruturação são muitas:

- Uma maior visibilidade para o sistema, não somente para os tradicionais programadores, mas também para as equipes de implantação e manutenção;
- Uma melhor base para a comunicação interna entre as equipes de desenvolvimento multidisciplinares;
- Uma separação clara entre diferentes responsabilidades;
- Um maior foco no problema de controle real e suas possibilidades de solução;
- Uma base para a reutilização de software.

Como um todo, a estruturação consiste na divisão do problema em partes menores, as quais novamente podem ser subdivididas. Não existe limite para isto: entretanto, não é prático avançar para uma granularidade muito fina, a ponto de aumentar o esforço para integração das partes.

O uso de blocos modulares é associado com cinco princípios:

1. A linguagem de programação deve suportar as unidades modulares de código.
2. As unidades devem ser compostas em tal forma e tal número que elas tenham poucas interfaces e poucas interações entre si.
3. As interfaces devem ser pequenas, necessitando pouca troca de dados.
4. As interações dos módulos devem ser de definição explícita, para aumentar a sua reutilização.

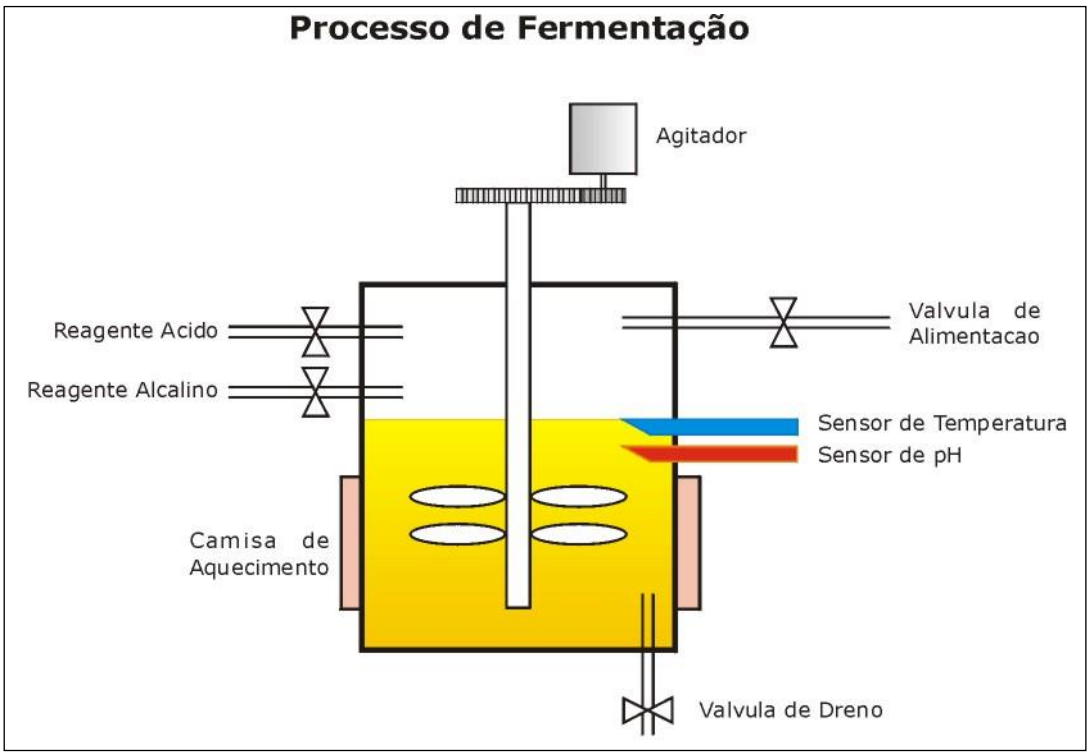
5. Os módulos devem proporcionar o encapsulamento de dados: os dados da aplicação são particionados e cada partição deve ser acessível por um conjunto próprio de funções, o qual deve protegê-la (escondê-la) do uso indesejável.

ESTRUTURAÇÃO IEC 61131-3: 7 PASSOS PARA O SUCESSO

Os 7 passos seguintes proporcionam um caminho para o sucesso da programação estruturada:

1. Identificação das interfaces externas do sistema de controle;
2. Definição dos principais sinais trocados entre o sistema de controle e o resto do processo/sistemas;
3. Definição de todas as interações com o operador, ações de controle e dados de supervisão;
4. Análise do problema de controle de cima para baixo (*top-down*) quebrando o mesmo em partições lógicas;
5. Definição dos blocos funcionais necessários;
6. Definição dos tempos de ciclo (*scan*) exigidos pelas diferentes partes da aplicação;
7. Configuração do sistema através da definição dos recursos, conexão dos programas com as entradas e saídas físicas e associação dos programas e blocos funcionais com as tarefas.

A norma IEC 61131-3 proporciona o ambiente certo para suportar todos estes passos. Como? Para entendermos isto, vamos analisar um exemplo real: um processo de fermentação e seu sistema de controle. A figura a seguir apresenta todo o sistema. Você deve analisá-lo como um sistema funcional e não como um objeto de software.



O processo de fermentação consiste de um grande vaso, o qual pode ser enchido (Válvula de Alimentação) com líquido, pode ser aquecido com a Camisa de Aquecimento (a refrigeração ocorre por convecção), pode ser agitado através do motor, e onde os fluidos ácido e alcalino podem ser adicionados no mesmo. Após o tempo de processamento, é usado a Válvula de Dreno para obtenção do produto.

Para criar um programa de controle para este exemplo, vamos seguir todos os passos definidos anteriormente:

1. *Identificação das interfaces externas do sistema de controle;*
    - Realimentação do sensor de temperatura
    - Realimentação do sensor de pH
    - Realimentação das posições da válvula
    - Realimentação do funcionamento do motor
    - Saída para as válvulas
    - Saída para o motor
    - Saída para a camisa de aquecimento

2. *Definição dos principais sinais trocados entre o sistema de controle e o resto do processo/sistemas;*

Neste exemplo não existem acoplamentos com o restante do processo, por motivos de simplificação, o que não representa o caso real. Por exemplo, poderia ser necessária uma conexão com os tanques de onde os líquidos são provenientes, ou uma conexão com o sistema de drenagem, como um sistema de transporte para vasos ou garrafas. Também, poderia existir uma conexão com o sistema de gestão, como o ERP por exemplo.

3. *Definição de todas as interações com o operador, ações de controle e dados de supervisão;*

Para o operador foram definidos os botões de 'Partir', 'Parar' e a 'Duraçao' como entradas do sistema.

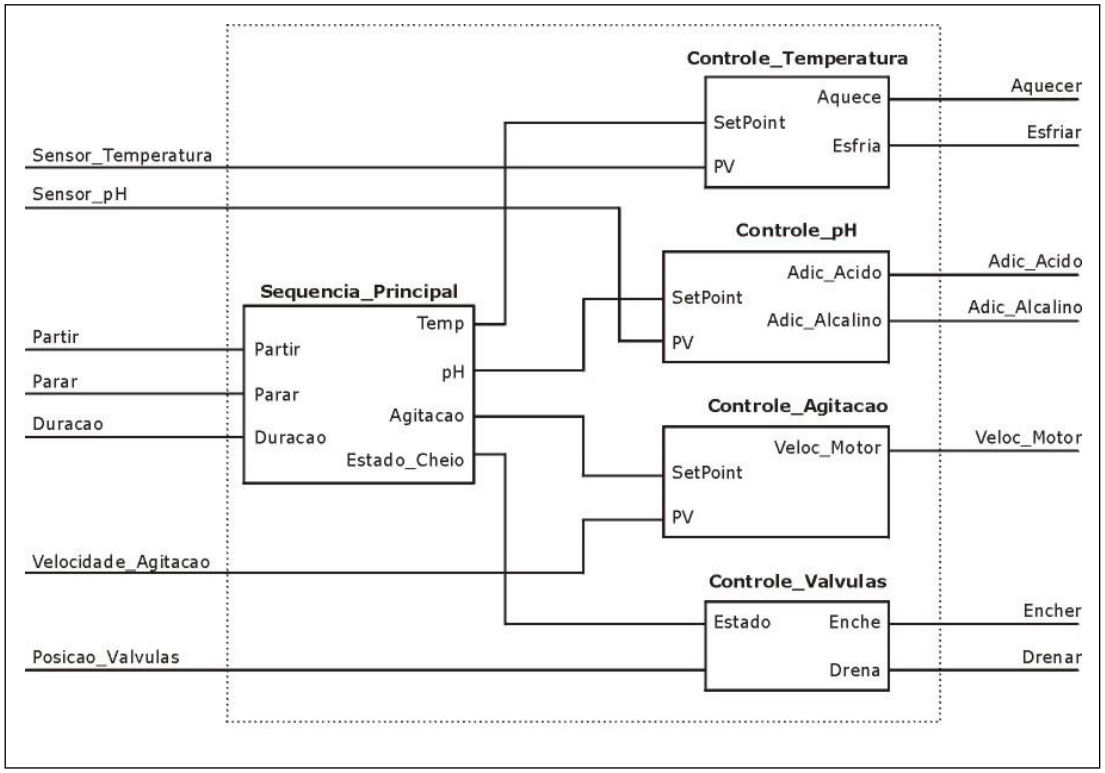
4. *Análise do problema de controle de cima para baixo (top-down) quebrando o mesmo em partições lógicas;*

Podemos identificar 5 funções principais:

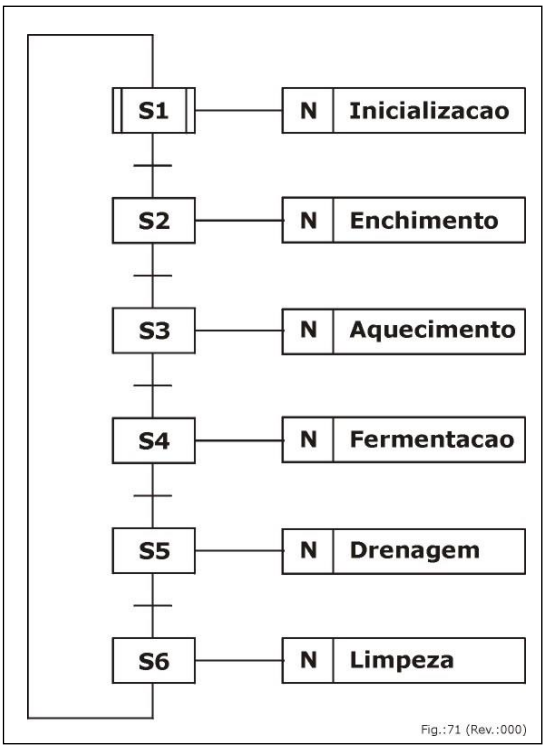
1. **Seqüência Principal**, por exemplo, etapas macros do processo – enchimento, aquecimento, agitação, fermentação, drenagem, limpeza.
2. **Controle da Válvula**, por exemplo, válvulas usadas na operação para encher e esvaziar o vaso.
3. **Controle de Temperatura** para monitoração da temperatura do vaso: modulação do aquecedor.
4. **Controle do Agitador** para acionamento do motor do agitador pela seqüência principal do processo.
5. **Controle de pH** para monitoração da acidez da fermentação, adicionando reagente ácido ou alcalino quando necessário.

5. *Definição dos Blocos Funcionais necessários;*

Usando as definições anteriores e adicionando os Blocos Funcionais (alto nível) às mesmas, nós podemos representá-los na linguagem de programação Diagrama de Blocos Funcionais (FBD). A visão do programa de controle da fermentação poderia ser como esta: (Leia da direita para a esquerda. Na esquerda estão as entradas; na direita estão as saídas).



O Bloco Funcional Sequencia_Principal é conectado com as entradas do operador. Este é suportado pelos demais blocos de controle, os quais são conectados com as entradas e saídas relevantes. Estes blocos adicionais poderiam ser blocos que são fornecidos pelos diversos fabricantes, tais como bloco de controle de temperatura, usando o algoritmo PID. Alternativamente, estes blocos poderiam ser criados pelo próprio usuário. Por exemplo, este é o caso do bloco Sequencia_Principal. Observando este bloco em detalhes, nós poderíamos estruturá-lo usando a linguagem SFC da seguinte forma:



Nós começamos de cima com a Inicializacao: desde que nós não sabemos o estado inicial do sistema ao energizarmos o mesmo, devemos aqui assegurar as posições das válvulas, etc. Então partimos para o enchimento até atingirmos o nível desejado. A próxima fase é o aquecimento até que o processo de fermentação se inicie. Quando isto ocorre, nós passamos para a próxima fase: o controle do processo de fermentação propriamente dito. Após a sua conclusão, nós drenamos a solução, logo após fazemos a limpeza e estaremos prontos para recomeçar do ponto inicial novamente (cima).
Esta decomposição do problema proporciona a todos os envolvidos uma visão clara das seqüências necessárias, além de permitir a modularização utilizando Blocos Funcionais os quais podem ser programados em qualquer uma das quatro outras linguagens da norma.
Ou, de outro ponto de vista, nossa especificação funcional dos requisitos do usuário está praticamente pronta!
O trabalho de programação agora pode ser feito no nível dos blocos de ações do SFC. Estes poderiam se distribuídos entre diferentes pessoas, com diferentes formações. Por isso, a norma IEC definiu 2 linguagens gráficas, Diagrama Ladder e Diagrama de Blocos Funcionais, e 2 linguagens textuais, Lista de Instruções e Texto Estruturado, para melhor atender às necessidades e ao problema em si. Além disso, uma decomposição adicional dos blocos de ação usando a própria linguagem SFC também é possível, se necessário.

6. *Definição dos tempos de ciclo (scan) exigidos pelas diferentes partes da aplicação;*

Neste exemplo nós temos em princípio um único ciclo, o qual nós poderemos executar de forma contínua. Alternativamente, nós poderíamos ter que executá-lo de forma periódica, a cada 20 ms por exemplo. O tempo restante poderia ser usado para seqüências adicionais, como o controle e monitoração do sistema de transporte/envase da drenagem, por exemplo, ou mesmo a monitoração de quaisquer condições de falha e proteção.

7. *Configuração do sistema através da definição dos recursos, conexão dos programas com as entradas e saídas físicas e associação dos programas e blocos funcionais com as tarefas.*

Esta fase é dedicada para o sistema em questão. Ela inclui o mapeamento das entradas e saídas simbólicas para o endereçamento físico dos cartões. Através da representação simbólica é possível criar uma alta independência do hardware. Isto é especialmente válido para a criação de Blocos Funcionais, os quais devem ser independentes do hardware. Através da identificação clara de onde o mapeamento físico é feito, torna-se extremamente fácil lidar com possíveis rearranjos da fiação de campo. Por exemplo, a troca dos cabos de duas entradas digitais significa que alguém deverá editar apenas duas linhas no mapeamento físico. O resto do programa continuará válido, sem necessidade de alterações.
Ainda aqui são mapeados os recursos, indicando quais partes irão ser executadas em quais processadores do sistema. A norma IEC 61131-3 suporta ambientes com multi-processamento, embora muitos dos sistemas atuais ainda usem um único processador para todos os programas.
Por último e não menos importante, devem ser mapeadas as tarefas para os ciclos de varredura e eventos, conforme definido no passo 6. Neste sentido, é possível de se ter múltiplos programas em um único sistema, o processo de fermentação por exemplo, suportado por monitorações e controles dos ambientes suportados, tais como o envasamento ou os níveis da cadeia de suprimentos adiante.

CONCLUSÃO

A norma IEC 61131-3 para programação proporciona uma poderosa ferramenta em todos os níveis. O uso dos Blocos Funcionais no nível mais alto permite uma excelente visão do sistema em desenvolvimento, resultando numa maior facilidade de entendimento e transparência. A norma também serve de guia para modularização do problema de controle. Adicionalmente, a norma proporciona a base para a separação das diversas etapas do desenvolvimento, focando a atenção na criação de códigos reutilizáveis no nível da programação.

Referências

Eelco van der Wal
Diretor da PLCopen
www.plcopen.org

Tradução:
Marcos Fonseca
Diretor da Divisão de Tecnologia da Automação – ATAN Sistemas
www.atan.com.br

www.plcopen.org - Website independente para informação adicional