UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
Más que un estado sólido
https://revistas.udistrital.edu.co/index.php/visele

VISIÓN ELECTRÓNICA

VISIÓN INVESTIGADORA

Metodología para estandarización de componentes SCADA bajo normas ISA

ISA Standard-Based SCADA Component Standardization Methodology

Ingrid Johana Donado Romero.[1]***, Juan Carlos Villamizar Rincón.***[2]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 01/09/2017
Recibido: 23/11/2017
Aceptado: 13/01/2018

Palabras clave:
Estados
Modos
Normativa
Objeto
Programación
Transición

Keywords:
States
Modes
Normative
Objects
Programming
Transition

RESUMEN

A continuación, se presentan los resultados de un trabajo investigativo desarrollado con el propósito de realizar un aporte a empresas dedicadas a la automatización de procesos industriales como la elaboración de productos alimenticios, fabricación de diferentes insumos, generación eléctrica, entre otros, con miras a alcanzar una mayor calidad y competitividad en sus desarrollos de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés). Teniendo en cuenta lo anterior, la investigación se centra en la etapa de control, ya que incide directamente en la programación del controlador lógico programable (PLC, por sus siglas en inglés). Se obtiene una metodología conformada por ocho procedimientos de desarrollo estandarizado de funciones o también llamados objetos de programación basada en normas ISA y en buenas prácticas de ingeniería. En particular, se seleccionó como modelo un proceso de producción de papel en el cual se realizó la identificación de los objetos mínimos requeridos y su programación sólida, dando como resultado bloques de funciones eficientes —con más de una docena de valores asociados— aplicables a cualquier tipo de proceso industrial en el que se cuente con procesos automatizados.

ABSTRACT

This article presents results of an investigative work developed with the purpose of making a contribution to companies dedicated to the automation of industrial processes such as the elaboration of food products, manufacture of different inputs, electrical generation, among others, in to achieve a higher quality and competitiveness in its developments of Control and Supervision Systems "SCADA". The present investigation focuses on the control stage, since it directly affects the programming of the Programmable Logic Controller, PLC. It specify a methodology consisting of eight procedures for the standardized development of functions -also called 'objects' of programming- based on ISA norms and good engineering practices. Particularly, was chosen as model a paper production process in which the minimum required objects were identified and finally their solid programming, resulting in efficient blocks of functions -with more than a dozen associated values- applicable to any type of industrial process in which there are automated processes.

[1]Ingeniera electrónica, Universidad Pedagógica y Tecnológica de Colombia; especialista en Automatización de Procesos Industriales, Universidad Tecnológica de Bolívar; magíster en Ingeniería Electrónica, Universidad Pontificia Bolivariana. Correo electrónico: ingriddonado@unicesar.edu.co

[2]Ingeniero eléctrico, Universidad Industrial de Santander; magíster en Potencia Eléctrica, Universidad Industrial de Santander. Correo electrónico: juan.villamizar@upb.edu.co

Citar este artículo como: I. J. Donado y J. C. Villamizar. "Metodología para estandarización de componentes SCADA bajo normas ISA". Visión Electrónica, algo más que un estado sólido, Vol. 12, No. 1, 14-21, enero-junio 2018.
https://doi.org/10.14483/22484728.13402

1. Introducción

Las empresas de automatización en Colombia deben estar a la vanguardia de la estandarización para poder competir en un mercado cada vez más globalizado; en esta vía, Indutrónica del Caribe S.A.S., es una compañía de ingeniería que provee soluciones integradas en las áreas de informática, telecomunicaciones, sistemas inteligentes de transporte, automatización y energía, así como servicios de asesoría, consultoría y direccionamiento tecnológico.

No obstante, en el compromiso y la búsqueda de transitar continuamente por la ruta que conduce hasta la excelencia, de innovar, de buscar continuas mejoras en el proceso de producción, se han identificado factores como la falta de uniformidad en sus desarrollos, es decir, el hecho de que cada desarrollo tenga el estilo propio del programador o que no exista una metodología única de programación influye directamente en que no se pueda seguir escalando hacia la competitividad, ya que no es tarea fácil dar continuidad a un proyecto iniciado por otro ingeniero, o bien, encontrar en qué punto va el desarrollo. Particularmente, por ejemplo, no existen librerías completas de elementos que puedan ser usados cada vez que se necesite desarrollar un sistema SCADA, siempre es necesario comenzar de cero.

En este sentido, el problema de investigación consistió en especificar un procedimiento de desarrollo de proyectos SCADA basado en estándares, normas técnicas y en buenas prácticas de ingeniería, iniciando desde el estudio de las normativas existentes, la selección de los aspectos a tener en cuenta y, finalmente, el despliegue del mismo para efectos de plasmar un modelo que sirva como guía a la compañía en desarrollos posteriores, además de comprobar su funcionalidad [1–4].

Según la literatura, podría decirse que un sistema SCADA surge como la aplicación e integración de la electrónica industrial con sistemas de control, comunicaciones industriales e informática aplicada [5]; en este sentido, todo partió de la necesidad de automatizar cálculos matemáticos, cuyo desarrollo fue dependiendo de la evolución de la electrónica y los ordenadores, los cuales finalmente se adaptaron para realizar tareas de adquisición de datos y control de procesos. En el caso aquí previsto, la adaptación consistió, por un lado, en el desarrollo de paquetes de *software* aplicados que permitieran la implementación de un sistema SCADA y, por otro lado, en la evolución de sistemas de visualización, control a distancia y transmisión de datos.

Por otra parte, siguiendo las guías sobre diseño y desarrollo de *software* para un proyecto de adquisición de datos típico de control de supervisión, se inicia con la identificación de las entradas y salidas físicas de cada controlador, siguiendo la documentación del *software*, las pruebas y la salida del sistema SCADA [6]. A manera de analogía, conociendo que la metodología seguida por las refinerías de petróleo y gas en general conduce a soluciones basadas en un sistema de control distribuido (DCS, por sus siglas en inglés) para proporcionar todos los procesos y funciones de control de los equipos [7], se opta por otro camino: a través de un sistema SCADA/PLC controlar una refinería de petróleo en lugar del control convencional a través de DCS.

Por lo anterior, se seleccionó un proceso modelo, iniciando con la identificación y la descripción de los objetos mínimos requeridos en la plataforma de desarrollo SCADA, continuando con el desarrollo de las funciones u objetos y la estandarización de las variables a usar para ser llamadas en los nuevos programas. En consecuencia, fue posible especificar una metodología que incluye ocho procedimientos de desarrollo estandarizado de funciones o también llamados objetos de programación, basada en normas ISA y en buenas prácticas de ingeniería, realizando la identificación de los objetos mínimos requeridos para un proceso modelo, controles para distintos tipos de motores y válvulas y, finalmente, su programación sólida.

Como resultado se obtienen unos objetos eficientes, con más de una docena de valores asociados aplicables en diferentes procesos industriales susceptibles de ser monitoreados o controlados

2. Marco teórico

En la automatización de procesos se cuenta con sistemas SCADA que se pueden definir como una aplicación o un *software* especialmente diseñado para funcionar sobre computadores, con el fin de tener el acceso a datos de procesos industriales para su monitoreo y control. Un SCADA combina diferentes acciones en un mismo sistema: monitoreo, adquisición, transporte de datos, comunicación, control y supervisión. El operario no solamente tiene acceso a la información de la planta controlada, sino que también puede accionar actuadores y corregir o cambiar parámetros, todo desde una misma pantalla. Esta variación de variables controladas en tiempo real hace que un SCADA sea un sistema flexible y, a su vez, muy eficiente.

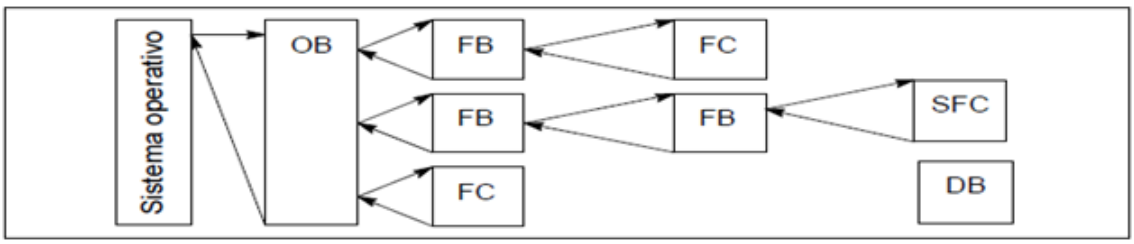
En cuanto a la programación estructurada del sistema de control, usar PLC como equipos electrónicos

programables en lenguaje no informático aprovecha su diseño orientado para controlar en tiempo real y en ambiente de tipo industrial procesos secuenciales. Trabaja con base en la información recibida por los sensores y el programa lógico interno, actuando sobre los accionadores de la instalación.

En cuanto a programas voluminosos e impredecibles, ocasionalmente es recomendable dividirlos en distintas secciones de programa. Las secciones de programas deben ser partes de programa, cerradas en sí mismas, que tengan una relación tecnológica o funcional. Estas partes de programa reciben el nombre de objetos bloques de función; así, un bloque es una parte del programa de usuario delimitada por su función estructurada o finalidad de uso.

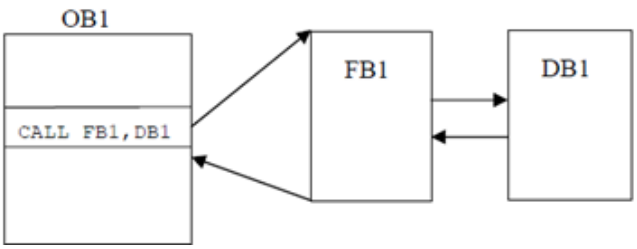
En la Figura 1 se observa la función (FC, por sus siglas en inglés) y el bloque de función (FB, por sus siglas en inglés) subordinada al bloque de organización (OB, por sus siglas en inglés), para que pueda ser procesada por la CPU esta FC tiene que ser llamada también desde el bloque de orden superior; a los FB se les denomina "objetos". Otros elementos utilizados en un programa de usuario son: los bloques de datos (DB, por sus siglas en inglés), las funciones de sistema (SFC, por sus siglas en inglés) y los bloques de funciones de sistema (SFB, por sus siglas en inglés).

Figura 1: Estructura de un programa de usuario [3].



Un FB dispone de un bloque de datos asignado como memoria (bloque de datos de instancia). Los parámetros que se transfieren al FB, así como las variables estáticas, se memorizan en el DB de instancia, las variables temporales se memorizan en la pila de datos locales. Los datos memorizados en el DB de instancia no se pierden al concluir el tratamiento del FB, mientras que los datos memorizados en la pila de datos locales sí.

Figura 2: Estructura de programación de un FB [3].



Por otro lado, para la caracterización del sector industrial, entre los múltiples sectores industriales existentes se selecciona el sector papelero, tomando como referencia ser un cliente que hace presencia en la costa Caribe y que usa en su gran mayoría de procesos elementos como válvulas y motores comunes a muchos otros sectores industriales. En este orden de ideas, se identifican entonces los principales productos fabricados y comercializados: papel *kraft*, bolsas de papel, papel higiénico, servilletas, toallas de cocina, entre otros. Los papeles están fabricados con materia prima reciclada, seleccionada minuciosamente y procesada utilizando equipos de alta tecnología para entregar productos de excelente calidad.

La fabricación del papel se realiza en una máquina papelera constituida por una tela que gira a gran velocidad, accionada por un conjunto de rodillos mecánicos; sobre esta tela cae una mezcla de fibras que forman una capa, esta pasa por rodillos que la succionan y la secan dando forma al papel; posteriormente, este se rebobina y almacena. El sistema está compuesto en gran medida por bobinadores, extractores, sopladores, inyectores, *dampers*, quemadores, succionadores, bombas de recirculación, entre otros.

De lo anterior, se decide realizar objetos para arrancar motores de tres formas distintas: arranque directo, arrancador suave y mediante variador de velocidad. Adicionalmente, se desarrollan objetos para el control de válvulas *on/off* y proporcionales, que también integran un gran número de elementos del sistema de control general en la planta escogida y de muchas otras plantas a nivel general.

En cuanto al universo normativo, se indica que la normalización es el proceso, método o sistema definido exhaustivamente para aplicar, instalar o describir un proceso o sistema específico. Revisando el universo normativo existente, relacionado con la creación y regulación de sistemas SCADA, se identificaron diferentes asociaciones, normas de países o nacionales, normas que han trascendido fronteras conocidas como internacionales. Ahora bien, aparte de las normas también se definen los sellos de calidad, esto es un tipo de certificación que garantiza que el producto o sistema fue verificado y validado por la institución otorgante.

Por lo anterior, para la parte de la programación en el PLC, se trabajó básicamente con dos normas: la norma ISA 88 y la norma ISA106, que reemplazan a su vez aspectos de la ISA 95. Se identificaron y se seleccionaron los segmentos de las normas que harán parte del procedimiento de diseño e implementación de sistemas SCADA que se usará para las diferentes empresas.

Respecto de la norma ISA 88, vale la pena indicar que trata de procesos de control por lotes (*batch processing*) [1], es decir, los procesos de control que tienen un inicio, un desarrollo y un fin; cuando ese fin se encuentra, el proceso termina y se puede reiniciar ese ciclo con un nuevo conjunto de materias primas para obtener el producto final que se obtiene de ese batch processing. Todos los procesos en la industria no funcionan así, hay otros procesos como los procesos continuos, y aunque son dos filosofías diferentes, los objetos tienen que responder a ambas filosofías porque no pueden haber objetos que respondan solo a una filosofía.

A continuación, se explica en qué aspectos debe cumplir con ambas normas. En términos de la ISA 88, quinto capítulo [1], se definen conceptos del control por lotes, de esta parte se trabaja: estructura, equipos, entidades, recetas, planes de producción, información de la producción y arbitramento. Estos seis primeros capítulos no tocan directamente al objeto, mientras que el séptimo capítulo de conceptos de control por *batch* sí tratan de la programación del objeto como tal.

En cuanto a los procesos continuos, la ISA 106 agrega unos conceptos nuevos que se conocen como *promt*, anuncio [2], tal como se observa en la Figura **3**. El anuncio le dice al operador cosas que le permitirán anticiparse a los eventos potencialmente malos que pueden ocurrir y que no deben hacerlo. Si el proceso se da, se pierde por completo y no se puede repetir precisamente porque es un proceso continuo. Allí se observa la diferencia entre estas dos normas.

Figura 3: ISA-106 coplements the ISA-18.2 alarm management standard [4].

Operator Notification Types	Operator is expected to take an action	Operator might need to be aware but is not required to take action
Arises from an abnormal process or equipment situation (ISA-18.2)	Alarm	Alert
Arises from a normal situation (ISA-106)	Prompt	Status Notification

Es necesario entonces que los objetos reporten cuándo están haciendo las transiciones que apliquen. En el presente desarrollo se están trabajando los objetos dentro de la filosofía ISA 106 y se está trabajando con el dispositivo, mas no a nivel *enterprice, sait, plant, plant area, unit*, ni a nivel *equipment* sino a nivel *device*.

3. Materiales y metodología

Las partes del programa de un PLC se pueden estandarizar y reutilizar, no obstante, se pueden trabajar varios programadores en un proyecto al mismo tiempo. A continuación, se describirá una metodología de ocho procedimientos para la programación estandarizada de FB basado en las normas ISA 106 e ISA 88 y en buenas prácticas de ingeniería, representados en el diagrama de bloques de la Figura **4**.

Figura 4: Diagrama de flujo para el desarrollo de la metodología de estandarización para la creación de FB u objetos SCADA basado en normas ISA.

Fuente: elaboración propia.

3.1. *Creación del bloque de función*

Crear FB que contenga el programa que se ejecutará siempre que sea llamado por otro bloque lógico.

- Paso 1: desde el administrador de Simatic, habrá que ubicarse donde se haya creado el proyecto.
- Paso 2: con el botón derecho se seleccionará la opción insertar objeto y se escogerá "bloque de función".
- Paso 3: se definen los datos dentro del FB.
- Paso 4: se edita la función FB del programa.

La dirección de la información deberá fluir como se mostró arriba en la Figura 2.

3.2. Creación del bloque de instancia

Disponer de un bloque de datos asignado como memoria (bloque de datos de instancia). Los parámetros que se transfieren al FB, así como las variables estáticas, se deben memorizar en el DB de instancia. Las variables temporales se deben memorizar en la pila de datos locales. Los datos memorizados en el DB de instancia no se perderán al concluir el tratamiento del FB, mientras que los datos memorizados en la pila de datos locales se perderán al concluir el tratamiento del FB.

Programar bloques de datos (DB)

- Paso 1: desde el administrador de Simatic, habrá que ubicarse donde se haya creado el proyecto.
- Paso 2: con el botón derecho se seleccionará la opción insertar objeto y se escogerá "bloque de datos", ese sería el DB.
- Paso 3: se definen los datos que se van a utilizar dentro del DB.
- Paso4: se edita la función DB del programa.

3.3. Definición de requerimientos mínimos de programación

En este procedimiento se deben considerar los requerimientos mínimos que deberá contener el objeto a programar. A continuación, se exponen los concernientes a válvulas y a motores.

- Crear FB para válvulas. El FB para válvulas deberá contener las siguientes funciones lógicas: operar en modo automático o manual, una entrada para arranque y una para parada, paradas de emergencia, enclavamientos, indicación de abierta, cerrada, abrir y cerrar válvula.
- Crear FB para motores. El FB para el motor deberá contener las siguientes funciones lógicas: operar en modo automático o manual, entrada para arranque y una para parada, una serie de enclavamientos, paradas de emergencia, activación térmica que permita el servicio de los equipos retardo al encendido y al apagado del equipo, horómetro, alarmas de mantenimiento, indicaciones de marcha, falla y paro, encender y apagar motor.

3.4. Considerar los modos de operación

En modos y estados se trabaja con elementos procedimentales, de las entidades asociadas a los equipos, es decir, se irá a nivel de elemento, llegando al nivel más elemental de cómo se arranca un motor o de cómo se abre una válvula; en este sentido, lo principal es que los elementos tienen o pueden tener modos de operación, estos modos en particular son tres: automático, semiautomático y manual.

En el modo automático, de preferencia en los procesos industriales, se tienen dos comportamientos: un comportamiento procedimental y un comportamiento a nivel de su estructura básica de control. En cuanto al control básico del sistema, se dice que el comportamiento automático de la entidad o del equipo se da porque este está manipulado por el algoritmo de control, es decir, no hay intervención de un operador o agente externo que tome la decisión, entonces ahí se da el automático.

En el comando para automático a nivel procedimental, el operador de un proceso de control en un proceso de producción puede pausar el progreso de algo que está en automático, puede "ponerlo en suspenso", pero no puede forzar las transiciones, no puede hacer que avance de un estado a otro, que pase de un proceso automático a otro, y el algoritmo de control se lo debe impedir porque precisamente el sistema está en automático y la manera correcta de que haya una transición de un punto al otro es que las apropiadas condiciones se hayan cumplido, por tanto, el operador no puede forzar el proceso.

El modo semiautomático, es un modo que no tiene implicaciones a nivel de control básico, solamente tiene implicaciones de transición; la industria de papel es un proceso en banda continua y esto no se utiliza, de hecho, en los objetos desarrollados no se incorpora este modo semiautomático de ninguna manera.

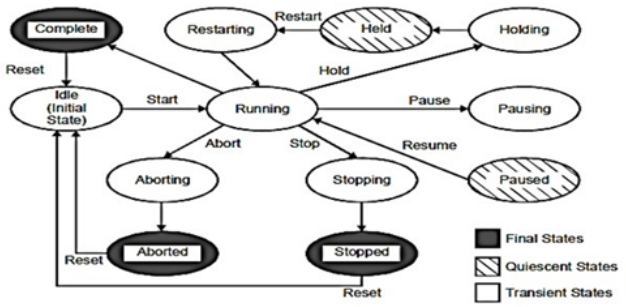
El siguiente modo es el modo manual, este modo tiene implicaciones procedimentales e implicaciones de control

básico; en el modo manual, si se observa desde el punto de vista del comportamiento, todos los procedimientos se desarrollarán a través de órdenes especificadas por el operador, es el operador quien decide qué se hace paso a paso.

3.5. Considerar los estados de operación

Las entidades de los equipos y los elementos procedimentales o los objetos pueden tener estados. Un estado es una "situación" del objeto o elemento que especifica completamente la condición actual del equipo, es decir, cuando se observa el estado del objeto, equipo o entidad, se debe conocer lo que en ese momento le está ocurriendo.

Figura 5: Diagrama de transición de estado, por ejemplo, estados para elementos de procedimiento [1].



Lo importante de los estados determinados en este estándar es que no se requieren estados adicionales a los aquí declarados. Los equipos, las entidades, los elementos, por su parte, pueden cambiar de estado cuando hay un comando y cambian hacia los estados que se definen en el mapa de estados (Figura **5**), pero esto solo sucederá si se definen las condiciones a través de la lógica de control o a través de señales externas que permitan que este estado cambie; por tanto, si el sistema se encuentra en automático y las condiciones apropiadas ocurren, él cambiará; por otro lado, si el sistema se encuentra en manual y las condiciones ocurren también cambiará de estado, ya que en manual no hay algoritmos de lógica del proceso pero sí hay algoritmos de seguridad.

Los estados son posibles en el objeto, como se mencionó anteriormente, y no todos los estados aplican para un objeto. A continuación, se definen los estados (Figura **6**).

Los estados más comunes se mencionan a continuación.

- *Idle*: significa detenido o en espera para realizar algo. En ocasiones se le puede llamar *ready*, porque el equipo se encuentra en un estado estable que le permite pasar hacia otros estados, generalmente hasta el estado de operación.

Figura 6: Matriz de transición de estado, por ejemplo, estados para elementos de procedimiento [1]

Command	Start	Stop	Hold	Restart	Abort	Reset	Pause	Resume
Initial State / No Command End State	State Transition Matrix							
Idle	Running							
Running / Complete		Stopping	Holding		Aborting		Pausing	
Complete						Idle		
Pausing / Paused		Stopping	Holding		Aborting			
Paused		Stopping	Holding		Aborting			Running
Holding / Held		Stopping			Aborting			
Held		Stopping		Restarting	Aborting			
Restarting / Running		Stopping	Holding		Aborting			
Stopping / Stopped					Aborting			
Stopped					Aborting	Idle		
Aborting / Aborted								
Aborted						Idle		

- *Running*: este es el estado al que se hace transición desde el estado de alerta o de espera de comando y llega a correr o a ejecutarse; puede terminar en un estado de complete si se ha ejecutado algo y ya ha finalizado lo que se ordenó hacer.
- *Complete*: este estado no aplica para los objetos de banda continua, porque suponiendo que el sistema estaba parado y se le dio orden de ejecutarse, el motor está andando, no vale la pena decir que ya se completó la tarea porque él no va a dejar de correr hasta que el operador no le diga que pare. Asimismo, pierde sentido este estado en una válvula.
- *Pausing*: significa que el proceso se está poniendo en pausa o espera.
- *Paused*: significa que el proceso ha terminado de ponerse en pausa y se ha detenido. Este estado es muy aplicable para el *bath processing* porque permite mirar qué ha ocurrido con el proceso en determinado punto y reenfocarlo, incluso se puede abortar si el resultado no es satisfactorio. En procesos de banda continua este estado no es muy aplicable.
- *Holding*: operación de contención. Permite poner el proceso en pausa o en espera y terminan cuando el proceso está pausado o el proceso está detenido esto es muy aplicable a *bath processing* porque se puede detener un proceso.

- *Held*: orden para mantener pausado.

Continúan los estados de reinicio de operaciones, paradas y abandono de proceso.

- *Restanting*: reiniciando. Si se tiene el estado en pausa y se quiere continuar, se debe reiniciar, pero sigue siendo un estado consecuencia de una parada.
- *Stopping*: este estado tiene mucho sentido en procesos continuos porque es posible detener el proceso; detener la operación de un motor o de una válvula. En el caso de una válvula, no se le podría decir a la válvula *running* ni *stopping*, a una válvula se le da orden de *open* o de *closed*, por ende, estos estados aplican principalmente para motores.
- *Stopped*: detenido. Después de haber dado un comando de arranque y un comando de parada indica que el elemento está parado. Para válvulas no aplica puesto que la válvula estará cerrada o abierta.
- *Aborting*: parando proceso. No es muy usual en procesos continuos puesto que si al sistema se le da la orden de parar él parará.
- *Aborted*: sistema abortado. Requiere iniciar nuevamente.

Si se observa el mapa de estados (Figura 5) es posible observar que hay unos estados iniciales y otros que son los estados hasta los cuales se puede realizar correctamente la transición.

3.5.1. *Considerar los estados procedimentales*

Considera desde el punto de vista procedimental los estados *idle, start, running, completed, stoping, stoped*, los demás no son útiles para procesos continuos.

3.6. *Programación*

Para el desarrollo de los objetos de programación con Step 7 de Siemens, se inicia igualmente desde el administrador Simatic, donde se desarrollará todo el entorno del proyecto y las carpetas de objetos con sus respectivos símbolos. A continuación, se darán los pasos de cómo se crea el bloque de arrancador directo de motores.

3.6.1. *Configuración de componentes y hardware*

Se crea un FB1, pulsando doble clic sobre el objeto o pulsando clic derecho, opción abrir objeto. Se sigue el mismo procedimiento para crear los bloques de función que sean necesarios. Automáticamente las FB creadas quedan guardadas en la carpeta bloques FB que se encuentra en los elementos del programa.

3.6.2. *Variables de funcionamiento del FB1. Arrancador directo con aspectos técnicos y normativos*

De las diferentes entradas y salidas de cada bloque creado, entre las más relevantes en el arrancador directo se encuentran: EN es la entrada básica de la caja o del objeto, es un booleano que permite que el bloque se active o desactive, la cual dependerá de la lógica del programa y determinará si el bloque está activo o no. HOA_ SWITCH_HAND y HOA_SWITCH_AUTO es un *switch* que toma dos posiciones booleanas *switch hand* y *switch auto*, si no está en ninguna de las dos posiciones se encuentra apagado (*off*), lo que impide que el motor arranque y significa que el motor no está en estado *ready*; si está en *hand* activará las funciones que permiten el control manual del motor y si está en *auto* activará las funciones que permiten el control automático del motor.

3.7. *Pruebas y simulaciones*

La tarea consiste en forzar el conjunto de reglas que utiliza cada FB y comprobar su correcto funcionamiento, observando el estado encendido o apagado de sus salidas.

3.8. *Encriptado*

La encriptación es el procedimiento mediante el cual se asegura cada bloque función o FB desarrollado, de esta esta manera el archivo no permitirá su lectura ni modificación interna.

4. Conclusiones

Fue posible establecer una metodología conformada por ocho procedimientos, la cual está soportada a su vez no solo por ISA 106, sino que también rescata sucesos importantes de transición y estados, un ente que estaba consolidado y que duró muchos años en la industria fomentados por ISA 88, por tanto, se desarrolló construyendo objetos que son compatibles con ISA 106 y con ISA 88 a la vez. Por otro lado, cabe mencionar también que los bloques de función u objetos son la base con la cual se construyen en forma general los automatismos de plantas de proceso con varios sitios de operación a nivel mundial.

Los objetos creados pueden ser usados y reutilizados y no deben ser modificados, pues serán los idóneos para responder a una serie de premisas estructurales y normativas, son objetos muy eficaces que deberán

usarse siempre que se necesite un objeto de la misma naturaleza, ya que lo que el usuario programador hace en la compañía es interrelacionar objetos entre sí para lograr un objetivo final.

Haciendo un resumen de valores de los presentes objetos basados principalmente en ISA 106 dentro de los procesos automatizados, se puede decir que se ganan más de una docena de beneficios: rendimiento de seguridad mejorado, reducción de pérdidas, alertas tempranas, incremento en la producción, mejoramiento de la calidad, mejor tiempo de respuesta, aumento de la efectividad del operador, flujo de conocimiento retenido, mejora de los planes de entrenamiento, seguridad de manejo de información y procesos, flexibilidad y facilidad procedimental, estandarización.

5. Reconocimientos

A Indutrónica del Caribe S.A.S., Barranquilla, Colombia, y a la dirección de investigaciones de la Universidad Pontificia Bolivariana, Floridablanca, Santander, Colombia.

Referencias

[1] American National Standard, ANSI/ISA–S88.01–1995, Batch Control. [En línea]. Disponible en: http://www.gmpua.com/GAMP/ISA-88.pdf

[2] Setting the Standard for automation, ANSI/ISA106, Procedure Automation for Continuous Process Operations. [En línea]. Disponible en: https://www.isa.org/isa106/

[3] Indutrónica del Caribe S.A.S. "Curso simatic step 7 nivel basico", [En línea]. Disponible en: http://www.indutronica.com/es/

[4] ISA Symposium, "Water/Wastewater and Automatic Control Symposium", [En línea]. Disponible en: http://isawwsymposium.com/save-the-date-2013-wwac-symposium-to-be-aug\-6-8-2013-in-orlando-florida/

[5] A. Rodríguez, "Sistemas SCADA", Colombia: Editorial Alfaomega, 2007.

[6] S. G. McCrady, "Procedimientos prácticos para el desarrollo de *software* SCADA", 2013

[7] I. Morsi, L. Mohy. "SCADA system for oil refinery control", *Elsevier Measurement Journal*, vol. 47, 2014.