Ingeniaritza Goi Eskola Teknikoa
Escuela Técnica Superior de Ingeniería
Bilbao

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

INGENIERITZA GOI ESKOLA TEKNIKOA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
BILBAO

TRABAJO DE FIN DE MÁSTER

GENERACIÓN AUTOMÁTICA DEL PROYECTO DE CONTROL PARA MÁQUINAS MODULARES, UTILIZANDO XML, PARA PLATAFORMA TIA PORTAL

Alumno/a *Fernández Alonso Eneko*
Fecha *Septiembre de 2017*
Director/a *Darío, Orive, Revillas*
Curso académico *2016/2017*

Director/a
Darío, Orive

Curso Académico
2016/2017

ÍNDICE

1 RESUMEN TRILINGÜE

1.1 Resumen

Actualmente en el mercado global, existe una tendencia por parte del consumidor a demandar productos cada vez más complejos y personalizados. Esta tendencia supone un reto para la industria y por ello está recogido dentro de los desafíos de la Industry 4.0. Dentro de las alternativas, el concepto de máquina modular destaca como solución para dar respuesta a la flexibilidad exigida, permitiendo adaptarse a las exigencias y demanda de los clientes. Además, la modularización permite alcanzar los objetivos con un precio competitivo. Este concepto supone la división de una máquina en elementos mecatrónicos, que son la conjunción de los componentes físicos (mecánicos, eléctricos, electrónicos) con los lógicos, necesarios para su control (hardware y software). La construcción de las máquinas mediante este concepto modular se realiza a partir de la combinación de los diferentes elementos mecatrónicos, permitiendo así, dada la gran cantidad de combinaciones posibles, un alto grado de personalización. La ventaja de la construcción de una máquina a través de la composición de elementos mecatrónicos, reside en la capacidad de su fabricación en serie.

En este documento se propone una metodología para la creación de cada uno de los módulos a través de proyectos de automatización del software de Siemens TIA Portal. También se describe un modelo para máquina modular con el que realizar de forma automática, a través de un generador de código, los proyectos de automatización, haciendo uso de tecnologías XML. El generador de código se encuentra dentro de una aplicación con una interfaz gráfica que permite al usuario el control de versiones de los proyectos de automatización, la captación de la información necesaria para la creación de las máquinas y el uso del propio generador. Por último, la aplicación genera un documento con las instrucciones que tiene que seguir el usuario para completar el proyecto de TIA Portal, donde la automatización no es posible.

Palabras Clave: Industria 4.0, Modularidad, Sistemas automáticos, Generación automática de código, Sistemas flexibles de fabricación.

1.2 Laburpena

Gaur egun, merkatu globalean, bezeroek gero eta produktu konplexuagoak eta espezializatuagoak eskatzen dituzte. Joera honek erronka itzela suposatzen du industriarako. Hori dela eta, 4.0 Industria erronka horien artean dago. Aukera ezberdinen artean, makina modularra konponbide egokienetakoa da, behar den moldakortasuna eskaintzen baitu, bezeroen beharrak asetuz. Gainera, moldakortasun honek helburuak prezio lehiakorretan lortzea ahalbideratzen du. Kontzeptu honek makina elementu mekatronikoetan zatitzea suposatzen du. Elementu mekatronikoak konponente fisikoen eta logikoen konbinazioari esker eraiki daitezke. Makinen fabrikazioa, moldakortasun kontzeptu honetan oinarrituta, osagai mekatroniko ezberdinak bateratzen egiten da; konbinaketa anitzak direla eta, pertsonalizazio handia lor daiteke. Gainera, makinak elementu mekatronikoen konbinaketaren bitartez eraikitzeak abantaila izugarria suposatzen du seriean fabrikatzean.

Dokumentu honetan, modulu bakoitza garatzeko metodologia bat aurkezten da, Siemens TIA Portal softwarea erabiliz. Honekin batera, XML teknologiak erabiliz eta kode sortzaile baten bitartez, automatizazio proiektuak era automatikoago batean garatzeko modelo bat aurkezten da. Kode sortzailea aplikazio baten barruan dago. Aplikazioak duen interfaze grafikoarekin automatizazio proiektuen bertsioa kontrolatu daiteke, makinak egiteko beharrezko informazioa lortzearekin bateraa, kode sortzaile bera erabiliz. Azkenik, aplikazioak dokumentu bat sortzen du, erabiltzaileak jarraitu behar dituen argibideak azalduz, proiektuaren barruan automatizatu ezin diren atalak burutzeko.

Gako-hitzak: 4.0 Industria, Modulartasuna, Sistema automatikoak, Kodeko sorkuntza automatikoa, Fabrikazio malguko sistemak

1.3 Abstract

Currently in the global market, consumers are increasingly demanding for more complex and personalised products. This trend poses a challenge for the industry, and it is therefore considered as a challenge within Industry 4.0. Framework. As of the alternatives, the concept of the modular machine stands out as a solution that would provide the required flexibility, being able to adapt to the customer's demands. Moreover, modularization allows achieving the objectives with a competitive price. The basic principle for this concept is to divide the machine into mechatronic elements, the merge between physical components (mechanical, electrical, electronic) and the logical components, which are required for its control (hardware and software). Building the machines using this modularity concept is based on combining a variety of mechatronic elements, which given the countless possible combinations, allows for great customization. Another advantage drawn from this concept is the ability to mass produce the machines.

Along the pages of this document a methodology for creating each of the modules through automation projects using the TIA Portal Siemens software is presented. A model of a modular machine is also described, with which the automation projects could be carried out automatically using a code generator, using XML technologies. The code generator is part of an application with a graphic interface which gives users control over the different versions of the automation projects, gathering the necessary information for the creation of the machines and using the code generator itself. Finally, the application generates a document with the instructions the user needs to follow to be able to complete the TIA Portal project steps where automation is not possible.

Key words: Industry 4.0, Modularity, Automation system, Automatic code generation, Flexible manufacturing systems.

2 LISTA DE TABLAS/ILUSTRACIONES/ACRÓNIMOS

2.1 Lista de tablas

2.2 Lista de ilustraciones

2.3 Lista de acrónimos

TIA: Totally Integrated Automation

XML: eXtensible Markup Language

AML: Automate Markup Language

PLC: Programmable Logic Controler

HMI: Human Machine Interface

FB: Function Block

FC: Fibre Channel

IP: Interfaz de Programación

DB: Data Base

FB_DB: Function Block Data Base

TIC: Tecnologías de la Información y Comunicación

API: Application Programming Interface

DOM: Document Object Model

SAX: Simple API for XML

3 MEMORIA

3.1 Objetivos y alcance del trabajo

3.1.1 Objetivos

Este trabajo tiene como objetivos generales los que se relacionan a continuación:

1. Manipular un proyecto de automatización fuera del entorno de Siemens, utilizando las herramientas que proporciona TIA Portal.

2. Definir un Metamodelo, que se ajuste a las características de una empresa real

3. Validar el Metamodelo con un ejemplo de aplicación

4. Establecer la estructura para realizar un generador de código

5. Desarrollar un generador de código

6. Validar el generador de código mediante un caso de estudio.

3.1.2 Alcance

El alcance del proyecto es la definición de la máquina modular, en base a componentes mecatrónicos reutilizables, que permite automatizar la definición de nuevas máquinas, asegurando la corrección de la definición y generando el proyecto de automatización (hardware y software) de la máquina completa. Se desarrolla para Controladores Lógicos Programables desde la herramienta TIA Portal de Siemens, utilizando el API TIA Portal Openness, que facilita la manipulación automática de proyectos completos.

En alcance del trabajo incluye el desarrollo de una arquitectura que permite la generación automática del proyecto de control para máquinas modulares, utilizando XML, para plataforma TIA Portal, evitando los inconvenientes que presenta hacerlo de forma manual y obteniéndose un mayor grado de reutilización de los desarrollos generados. La arquitectura desarrollada en el presente trabajo contempla el nivel máquina, pudiendo ampliarse añadiendo más niveles, como podría ser la línea completa de fabricación. El alcance también incluye la implementación de las diferentes aplicaciones necesarias para el funcionamiento del generador automático de proyectos de automatización para plataforma Siemens.

3.2 Beneficios que aporta el trabajo

En este apartado se relacionan los beneficios obtenidos una vez completado el trabajo, así como los potenciales derivados de la implantación de la solución final en las empresas que diseñan, fabrican y emplean máquinas modulares

3.2.1 Beneficios económicos

Los potenciales beneficios económicos que obtendrían las empresas relacionados con los resultados de su actividad están ligados a:

- La oportunidad de ampliar sus líneas de negocio a la servitización, extendiendo la oferta de productos a la de servicios.
- El aumento en las ventas de las nuevas máquinas, al poder ofertarlas a precios competitivos.

En ambos casos es esencial la implantación de la metodología Plug-and-Produce, que permite integrar o retirar módulos de una línea de fabricación de una forma rápida y sencilla

En cuanto a la oferta de nuevos servicios, la modularidad de las máquinas les permitirá incluir tanto la actualización de las tecnologías, así como el rediseño de las líneas de producción, adaptándolas en un corto periodo de tiempo a las necesidades del cliente en lo relativo a funcionalidades y capacidad de producción. Un ejemplo de servitización sería la contratación del servicio de test de fugas en lugar de la compra de la máquina.

En relación con la oferta de producto, la modularidad permite acortar el tiempo necesario para la puesta en el mercado de una nueva máquina o para comercializar módulos innovadores. De esta forma se reducirá el periodo de retorno de la inversión inicial en el diseño y desarrollo, a través de los ingresos por ventas, por lo que el precio final de la máquina no se incrementará significativamente a pesar de incorporar mejoras sustanciales.

En cuanto al periodo de retorno de la inversión en diseño y desarrollo, se muestra a continuación una gráfica en la que puede verse la evolución de los conceptos de costes e ingresos, a lo largo del ciclo de vida de un producto nuevo. En una primera etapa, coincidente con la de introducción en el mercado del producto, el flujo de efectivo es negativo ya que los ingresos no cubren los costes iniciales de desarrollo y de producción. A medida que aumentan las ventas se recupera la inversión inicial y comienzan a generarse beneficios. Los ingresos por ventas se mantienen durante la fase de crecimiento y madurez, hasta que comienzan a declinar las ventas y el producto debe ser sustituido.

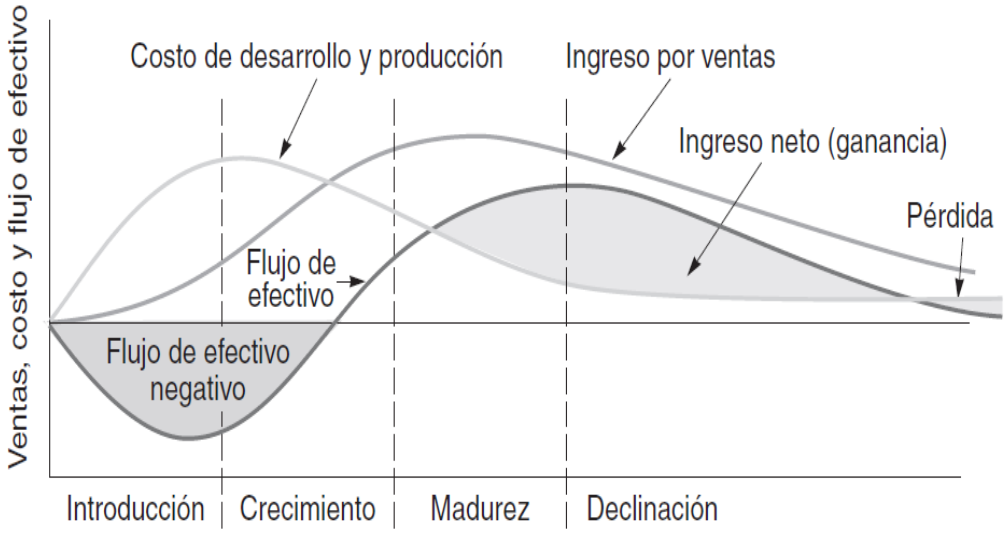


Ilustración 1 Gráfica tradicional del ciclo de vida producto

En el caso de las máquinas modulares objeto del trabajo, la evolución de los conceptos inversión vs beneficio difiere de la anterior, ya que antes de alcanzar el periodo de madurez y declinación, pueden introducirse en el mercado productos con módulos innovados. Las modificaciones requieren menores inversiones, por lo que se amortizan en un periodo de tiempo más corto, a través de los beneficios generados.

Finalmente cabe destacar el caso particular de las empresas que fabrican máquinas especiales, diseñadas bajo las especificaciones del cliente y que se producen unitariamente, ya que la evolución de los conceptos de costes e ingresos a lo largo del tiempo difiere del caso anterior. En la gráfica que se muestra a continuación se representa la evolución de la inversión vs beneficio correspondiente a este tipo de máquina.

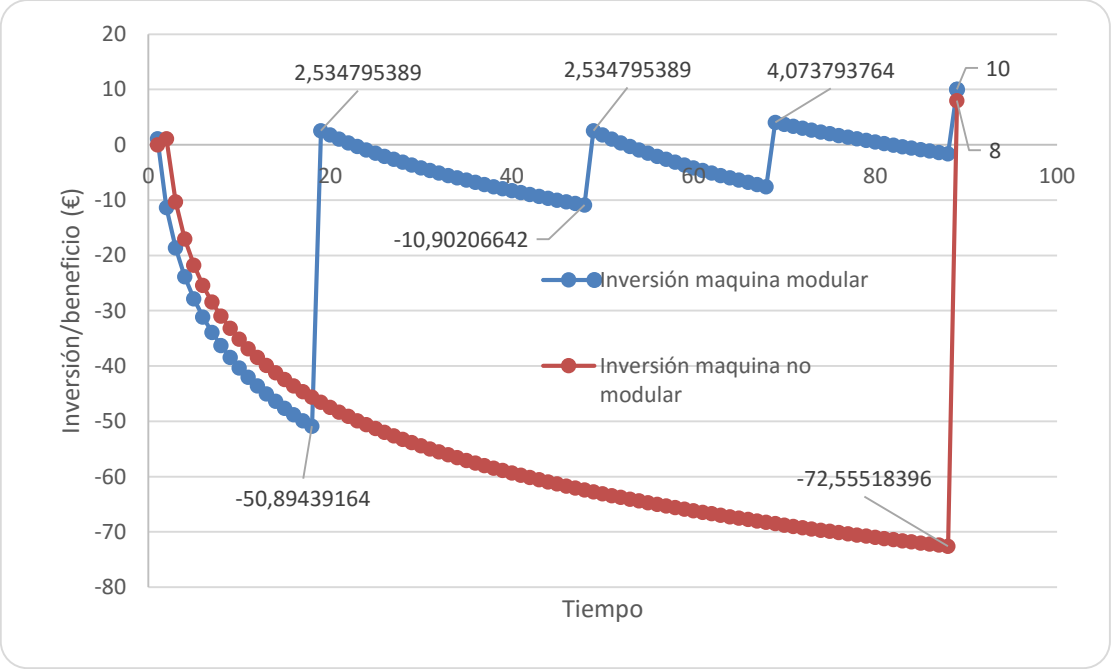


Ilustración 2 Gráfica ciclo de vida de maquina especial

Como puede apreciarse en la curva correspondiente a la inversión inicial de una máquina no modular de fabricación en serie, la inversión es recuperable solo cuando se entrega el producto final, haciendo que el riesgo sea muy alto y que cualquier percance pueda suponer la pérdida total de la inversión.

Sin embargo la máquina modular aunque podría requerir una inversión más elevada en una primera fase, con la venta del primer paquete (máquina más primer módulo) se recuperaría gran parte de la inversión, reduciendo el riesgo de no recuperar la inversión.

Otros factores que contribuyen al aumento de la competitividad de la empresa y de los resultados del negocio son:

- La reducción del tiempo de puesta en el mercado de las siguientes máquinas y módulos desarrollados con posterioridad
- El cambio de uso de la máquina no requiere la construcción de una máquina nueva, sino solo la actualización de los módulos, por lo que la solución tendrá un coste inferior
- La fidelización del cliente, que tiene a su disposición módulos para ampliar las funciones o mejorar el rendimiento de los procesos
- La mejora de la imagen de la empresa como una organización innovadora

En cuanto a los beneficios económicos potenciales que se materializarían durante la etapa de uso de las máquinas serían los relacionados con:

- La viabilidad de hacer cambios de características, añadiendo o mejorando las funciones
- La capacidad de implantar las nuevas versiones de un determinado módulo, extendiendo la actualización al resto de los módulos que comparten el servicio común
- La disminución del tiempo de puesta en marcha
- La facultad de crear sistemas más complejos de forma más sencilla, al dividir el problema en partes más pequeñas
- La minimización de las incidencias como efecto de la mayor automatización

3.2.2 Beneficios científicos

En relación con los beneficios científicos, al tratarse de un proyecto de I+D+i, el resultado del trabajo ha contribuido en la generación de un mayor conocimiento relacionado con un Metamodelo para una máquina modular a partir del cual es posible generar de forma automática el proyecto TIA Portal, utilizando tecnologías XML.

El conocimiento generado se va a aplicar en futuros trabajos del Departamento de Ingeniería de Sistemas y Automática de la UPV-EHU y ha sido difundido a través de un artículo científico que fue presentado en las XXXVIII Jornadas de Automática, celebradas en Gijón, en septiembre de 2017, con el título “Generación automática del proyecto de automatización TIA Portal para máquinas modulares”.

3.2.3 Beneficios técnicos

Se citan a continuación los beneficios técnicos aportados por el presente trabajo:

- La arquitectura desarrollada permite generar proyectos de automatización de máquinas modulares a partir de proyectos tipo de forma automática
- La reducción de los inconvenientes de la generación manual de los proyectos
- El aumento del grado de reutilización de los desarrollos realizados
- La viabilidad de añadir proyectos tipo básicos que se pueden ir desarrollando para futuras máquinas, al tratarse de una arquitectura abierta
- La posibilidad de ampliar la arquitectura de nivel máquina a otros niveles, como puede ser una línea completa del proceso de fabricación.

3.3 Contexto

El presente trabajo se ha realizado en el Departamento de Ingeniería de Sistemas y Automática de la UPV-EHU, cuya actividad investigadora se desarrolla en varias líneas de investigación alineadas con los retos de la denominada Industry 4.0, siendo uno de ellos el *Plug-and-Produce for Adaptable Factories*.

En este contexto el trabajo se enmarca, dentro de los proyectos de investigación aplicada, en el desarrollo de modularidad empleando tecnologías Siemens, en concreto haciendo uso de la herramienta TIA Portal Openness.

En la actualidad en el Departamento existen dos líneas de trabajo enfocadas a conseguir modularidad:

- Dentro del entorno de tecnologías de Siemens
- Haciendo uso de las tecnologías de Siemens, pero consiguiendo la modularidad fuera del entorno

Este trabajo se centra en la segunda línea y forma parte de un proyecto de I+D+i aplicada que se está desarrollando de forma conjunta con una empresa que diseña y fabrica máquinas especiales.

El entorno de aplicación es una máquina modular en la que es posible insertar nuevos módulos para ampliar las funcionalidades, como se muestra en la siguiente ilustración

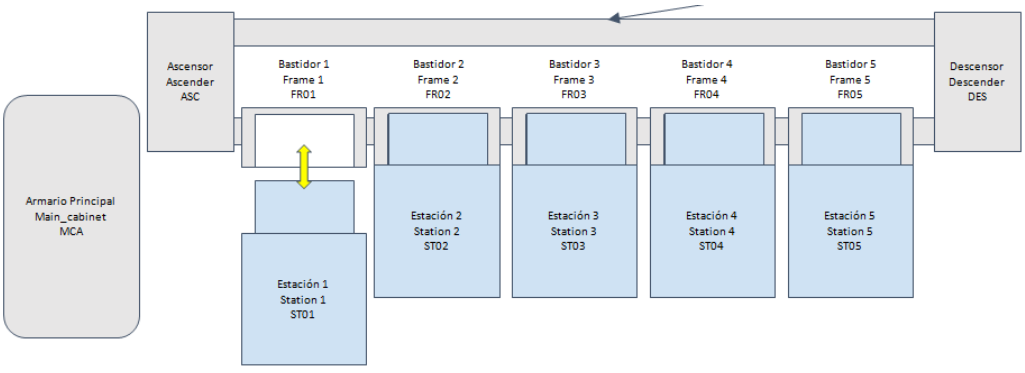


Ilustración 3 Maquina modular lineal, mediante inserción de carritos en bastidores

El resultado esperado del trabajo es lograr el desarrollo de módulos automatizados que permitan la flexibilización de la fabricación mediante la implantación del concepto *Plug-and-Produce*, así como la mejora de la eficiencia de los procesos de fabricación.

Un primer ejemplo consistiría en el aumento de la capacidad de producción de una máquina modular, mediante la incorporación de varios módulos iguales.

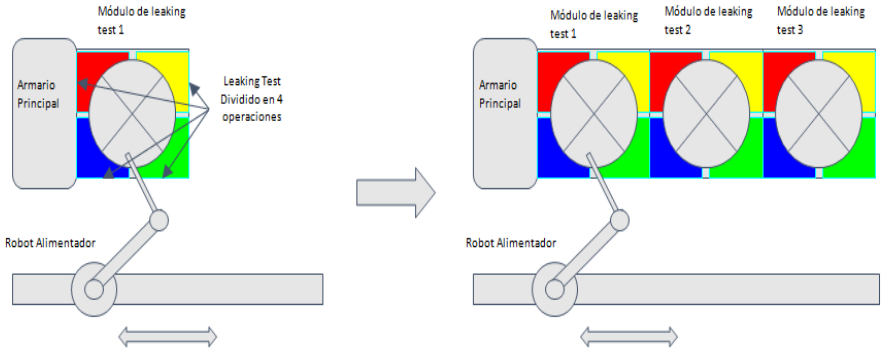


Ilustración 4 Maquina modular, duplicando módulos para conseguir mayor cadencia de producción

Un segundo ejemplo podría ser la ampliación de las funciones de la máquina, integrando módulos de diferente naturaleza.

3.4 Estado del Arte

El contexto actual de mercados globalizados, que exige productos de mayor calidad y con precios competitivos, está obligando a las empresas a aumentar el nivel de automatización de sus procesos de fabricación para dar respuesta a estas demandas. Por otra parte, los procesos son cada vez más complejos y deben ser lo suficientemente flexibles para adaptarse a las necesidades del mercado, por lo que los tiempos de cadencia de producción son cortos y las series de producción cada vez más pequeñas. [1]. A todo ello se une la necesidad de acortar el tiempo necesario para modificar la línea de producción con el fin de minimizar los tiempos de parada de la línea y reducir costes de producción. En resumen, los procesos de fabricación deben ser flexibles, adaptables a la fabricación de lotes de productos mucho más cortos que los actuales y más personalizados, que den respuesta a las necesidades y expectativas de los clientes finales. Todas estas adaptaciones y cambios en los procesos de fabricación deben realizarse de forma ágil, rápida y fiable, para que no incidan negativamente en la eficiencia de los procesos. Para hacer frente a estos retos, en Europa se ha optado por la estrategia Industry 4.0, también conocida como 4ª revolución industrial [2], que se fundamenta en la implantación de nuevas tecnologías. Para desarrollar dicha estrategia se han definido distintas las líneas de actuación, como la digitalización en las distintas etapas s del ciclo de vida del producto y la obtención de conocimiento sobre el proceso y el producto a partir de la captación, análisis y procesado de la información recogida en las distintas etapas del ciclo de vida. La aplicación conjunta de ambas da como resultado sistemas de fabricación más flexibles y eficientes que los actuales.

El diseño modular de sistemas de fabricación [3] es un ejemplo ya implantado en varios trabajos, disponiéndose de bibliografía en la que se presentan distintos casos prácticos.. Así, [4] se centra en aspectos de la granularidad de los módulos para conseguir altos grados de reutilización con mínimo esfuerzo. En [5] se propone la definición y diseño de un sistema de automatización capaz de asegurar su disponibilidad mediante técnicas y tecnologías de modelado y tecnología multi-agente. [6] estudia la problemática del control de versiones de software. En [7] se analiza aspectos de cómo generar el programa de control de sistemas modulares, en un entorno multidisciplinar de tres formas diferentes: basada en librería central, en base a parámetros y basada en plantillas. En estos trabajos, la modularidad se utiliza para facilitar el diseño del sistema de control. Pero en prácticamente todos ellos, bien a través del modelado, bien a través de la codificación, existen procesos manuales que son propensos a error. Sólo en [8] se propone una generación automática de código modular. Ahora bien, los módulos físicos se definen de forma abstracta y es el usuario el que debe modelar la máquina, definiendo todos sus componentes.

La aportación de este trabajo consiste en que, a través de la definición de la máquina modular en base a componentes mecatrónicos reutilizables, es posible automatizar la definición de nuevas máquinas, asegurando la corrección de la definición y generando

el proyecto de automatización (hardware y software) de la máquina completa. De esta forma se eliminan tareas repetitivas (como cambio de nombres de variables o asignación de direcciones de entrada/salida) y propensas a error. Concretamente se realiza para Controladores Lógicos Programables desde la herramienta TIA Portal de Siemens utilizando el API TIA Portal Openness que facilita la manipulación automática de proyectos completos.

3.5 Análisis de alternativas

3.5.1 Discusión general del problema a resolver

La modularización tiene dos beneficios principales: la posibilidad de reutilizar los diferentes módulos, y la de descomponer un sistema complejo en subsistemas más simples. Este trabajo se centra en dar respuesta a la cuestión de la reutilización, dejando libertad a la empresa para que posteriormente decida las subdivisiones de los módulos a realizar, con el fin de que el sistema sea más simple. Por tanto, se manipulará el módulo como conjunto, quedando las subdivisiones excluidas del alcance del presente trabajo.

Uno de los problemas que surgen al utilizar sistemas modulares, más allá del propio sistema modular, es la adopción de una metodología de trabajo específica. El modo de trabajar cambia y las soluciones de automatización dejan de ser locales para convertirse en globales, es decir, no se realiza una solución para un problema concreto sino para una tipología de problema. Ésto requiere aplicar una visión global cuando se desarrollan los módulos y una mayor inversión en la realización de cada una de las soluciones si se compara con la forma de trabajo tradicional. Por ello los beneficios de emplear un sistema modular no son percibidos a corto plazo, requiriendo la reutilización de los módulos para amortizar la inversión inicial.

3.5.1.1Modularización

El primer paso en la modularización de un sistema es la subdivisión del sistema en módulos, cuya composición de como resultado el propio sistema. La característica fundamental de un sistema modular es la definición de los propios módulos que consiste en delimitarlos mediante restricciones que consiguen que cada módulo tenga un comportamiento determinado, independientemente de su contenido.

La definición de un sistema modular, hace que el sistema sea menos flexible, por lo que es crítico conseguir que los límites coincidan con los del propio sistema, ya que cualquier sistema que quede fuera de las divisiones propuestas no es válido. Por eso cuando se definen los módulos se tienen que tener en cuenta todos los sistemas posibles dentro del área de conocimiento a modulizar y su uso, tanto los actuales como los que puedan incorporarse en el futuro.

La característica principal que define la usabilidad de los módulos es la granularidad, que es es el tamaño de los diferentes módulos en los que se divide un sistema. Si el

tamaño es muy grande, el número de combinaciones que se puede realizar es muy reducido y se corre el riesgo de no cubrir todo el sistema. Si el tamaño es muy pequeño, el número de módulos aumenta, lo que supone una mayor inversión y un mayor el trabajo de parametrización de los módulos y de composición del sistema. En ambos casos podría llegar a no compensar el cambio del modo de trabajo tradicional.

En resumen, para que la modularización sustituya al sistema no modular los beneficios que se obtengan tienen que ser superiores a los del método tradicional. Por ello la elección de la granularidad es crítica y los módulos tienen que ser lo suficientemente pequeños para abarcar todas las posibilidades y lo suficientemente grandes para que su uso sea prolongado.

Otra característica importante de un sistema modular es la jerarquía, ya que cada módulo está compuesto a su vez de módulos más pequeños por lo que es importante definir la jerarquía que existe entre ellos, así como su comportamiento con los modulos del mismo y de distinto nivel.

También se debe tener en cuenta la trazabilidad, ya que si el sistema no está perfectamente documentado y estructurado, existe el riesgo de desarrollar un módulo nuevo por no haberlos podido localizar uno ya desarrollado.

Por último, dado que los módulos no son permeables hay que definir la forma de interacción con el exterior mediante interfaces ya sean con módulos de nivel superior, inferior o de igual nivel jerárquico. Las interfaces hay que realizarlas de forma que cubran todas las necesidades de un tamaño tal que puedan ser utilizables.

3.5.1.2Modularización de software Siemens

Aunque tradicionalmente Siemens se ha caracterizado por ser un fabricante reacio a la apertura del acceso a sus herramientas, en los últimos años ha habido un cambio de tendencia, aunque sigue siendo un entorno muy cerrado. Este trabajo tiene en cuenta las particularidades del mismo, como pueden ser el sistema de directorios de TIA Portal.

Siemens proporciona herramientas que ayudan a resolver algunos de los problemas que surgen en la modularización de las máquinas, como la posibilidad de crear sistemas modulares, la reutilización de código o la capacidad de utilizar librerías, donde se pueden guardar partes de un proyecto para su posterior utilización.

Sin embargo a veces no son lo suficientemente potentes para cumplir con las especificaciones previamente marcadas, como en el casos de un proyecto de automatización en la que muchos elementos que tienen que ser únicos. En el supuesto de ser necesario un cambio de dirección se debe entrar en el configurador de TIA Portal y realizar los cambios de forma manual.

Se listan a continuación los elementos que deben ser únicos:

- Nombres de dispositivos
- Direcciones IP
- Direcciones asociados a cada dispositivo
- Nombres tablas de variables
- Nombres de las variables
- Direcciones de las variables
- Nombres de DBs
- Parámetros de las instanciaciones de los FBs
- Llamadas de FB_DBs
- Nombres de los FC
- Llamadas de FC
- Nombres de DBs
- Llamadas de DBs
- Timers
- Contadores
- Llamadas de OBs

En la siguiente ilustración podemos ver las diferentes direcciones de solo uno de los elementos hardware del proyecto de automatización, si se quieren cambiar las direcciones hay que entrar en el configurador de TIA Portal y cambiarlas desde cada tarjeta.

PLC OP110 M1	0	1		
Interfaz PROFINET_1	0	1 X1		
	0	1 X2		
DI 16x24VDC ST_1	0	2	0...1	
DI 16x24VDC ST_2	0	3	2...3	
DI 16x24VDC ST_3	0	4	4...5	
DI 16x24VDC ST_4	0	5	6...7	
DI 16x24VDC ST_5	0	6	8...9	
AI 4xU/I 2-wire ST_1	0	7	300...307	
DQ 16x24VDC/0.5A ST_1	0	8		0...1
DQ 16x24VDC/0.5A ST_2	0	9		2...3
DQ 16x24VDC/0.5A ST_3	0	10		4...5
F-DI 8x24VDC HF_1	0	11	20...25	20...23
F-DI 8x24VDC HF_2	0	12	30...35	30...33
F-DI 8x24VDC HF_3	0	13	40...45	40...43
F-DQ 4x24VDC/2A PM HF	0	14	50...54	50...54
Módulo servidor_1	0	15		

Ilustración 5 Direcciones de entrada y salida de un dispositivo dentro de un proyecto TIA Portal

Siemens hace una comprobación del proyecto que tiene instalado con los esclavos que posee y en el caso de tener una discrepancia aparecerá un error, por lo que si desconectamos un módulo tendremos que realizar un proyecto de automatización nuevo para evitar este error. El problema se acrecenta debido a que los proyectos son compilados, y estan tanto el software como la configuración por lo que hay que compilar desde TIA Portal si hay alguna discrepancia dentro de la configuración HW.

3.5.2 Alternativas propuestas

Se describen a continuación las distintas alternativas posibles para estructurar los diferentes bloques dentro de un programa de TIA Portal, correspondientes al paso de parámetros entre bloques funcionales y llamadas de funciones.

3.5.2.1 Paso de parámetros entre bloques funcionales

Dentro de la organización interna de los bloques funcionales de un programa de automatización, podemos hacer uso de las cabeceras que nos proporcionan los FB para hacer paso de parámetros entre bloques funcionales. La idea principal es encapsular la lógica de los programas interactuando entre ellos mediante el paso de parámetros.

Existen dos alternativas para hacerlo, mediante un método directo o un método anidado.

El primer método consiste en actuar sobre los valores actuales de todos los FBs, para lo que el generador de código tendrá que ser capaz de acceder a todos los FBs del proyecto.

Se muestran a continuación dos ejemplos imágenes de cómo asignar los valores a los parámetros en la herramienta TIA Portal. En la primera, aparece la asignación de los FBs y en la segunda la de un FC que llama a un FB, que a su vez es llamado por un OB.

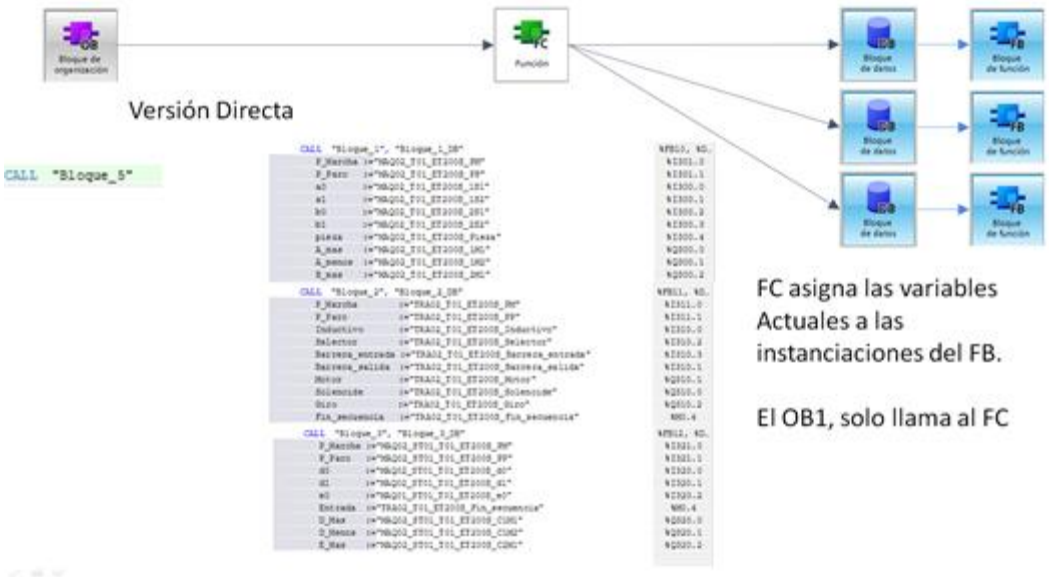


Ilustración 6 Método de paso de parámetros de un bloque funcional de forma directa

XML Directo

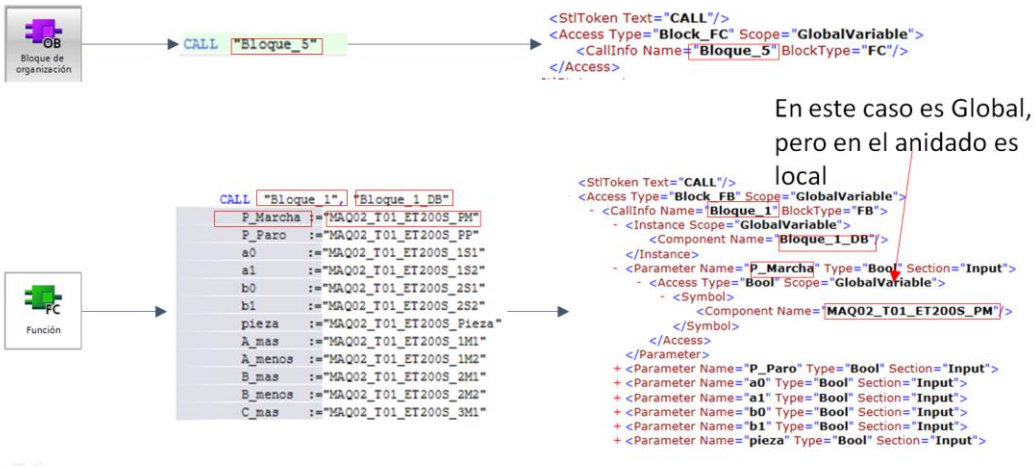


Ilustración 7 XML resultante del método de paso de parámetros de un bloque funcional de forma directa

El segundo método consiste en que la asignación de los parámetros de cada FB del proyecto se realiza desde el OB, para ello en cada llamada se van incorporando la asignación de los parámetros.

Se muestran a continuación dos ejemplos imágenes de cómo asignar los valores a los parámetros en la herramienta TIA Portal. En la primera, aparece la asignación de los FBs y en la segunda la de un FC que llama a un FB, que a su vez es llamado por un OB.

A diferencia del anterior,el generador de código soloo tiene que actuar sobre un único bloque, sin embargo previamente se han tenido que diseñar los bloques para que dispongan de esta utilidad.

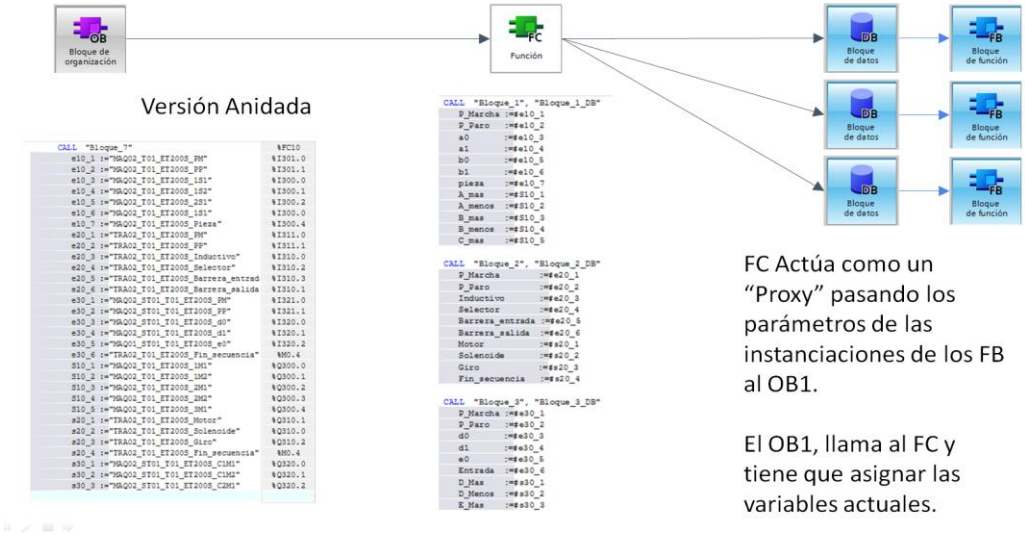


Ilustración 8 Método de paso de parámetros de un bloque funcional de forma

Ilustración 9 XML resultante del método de paso de parámetros de un bloque

Se recogen a continuación en una tabla, las ventajas y desventajas de los dos métodos.

Tabla 1 Tabla comparativa entre los diferentes métodos de paso de parámetros

	Ventajas	Desventajas
Método directo	Cada FB solo es responsable de sus parámetros	Para asignar las variables actuales hay que penetrar en las diferentes capas del programa
Método anidado	Los cambios se concentran únicamente en la capa superior	Cada FB es responsable de transmitir la información de los atributos del FB inferior

En relación con el paso de parámetros entre bloques funcionales, se ha optado por el método directo, porque que no presenta ninguna restricción a la hora de diseñar los módulos de una máquina, ya que los FBs son solo responsables de sus parámetros.

3.5.2.2 Llamadas a bloques funcionales

Dentro de un proyecto de automatización de una máquina modular, cada módulo tiene su propia lógica de control que pueden hacer uso de un mismo recurso, como puede ser el caso de un contador. Cada vez que se necesita no es necesario realizar toda la codificación, pudiéndose emplear un FB con la lógica e instanciarlo cuando es necesario.

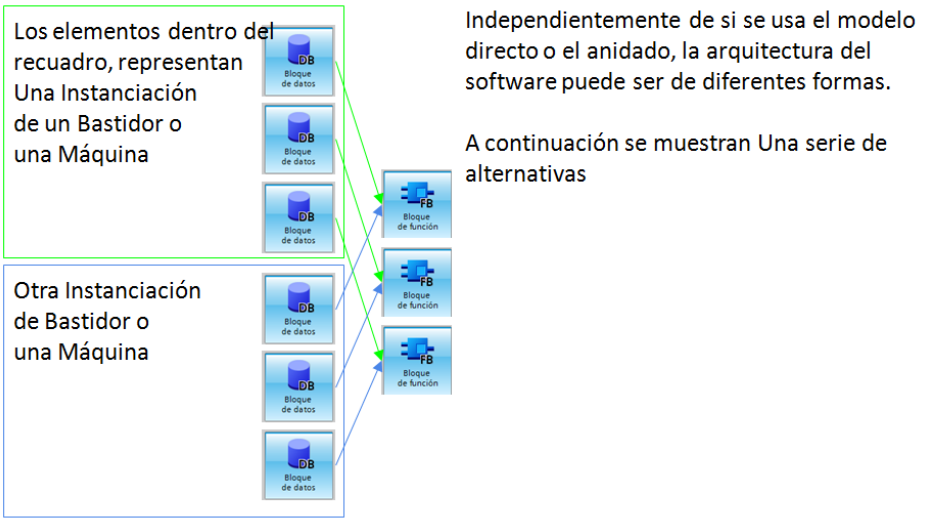


Ilustración 10 Instanciaciones de diferentes bloques funcionales

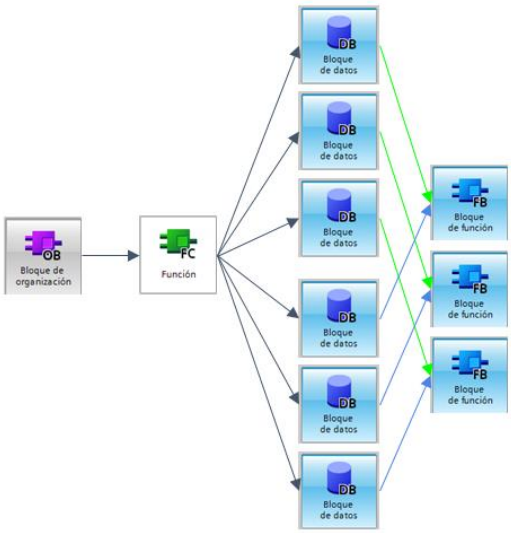
Se han analizado tres alternativas de llamadas a estas instanciaciones:

- Llamada desde el OB
- Llamada desde un único FC
- Llamada desde un FC por módulo

La primera alternativa consiste en que el bloque principal del sistema OB1 llame a cada una de las instanciaciones, como se muestra en la siguiente ilustración.

Ilustración 11 Llamada a las instanciaciones sin intermediarios desde el bloque de organización

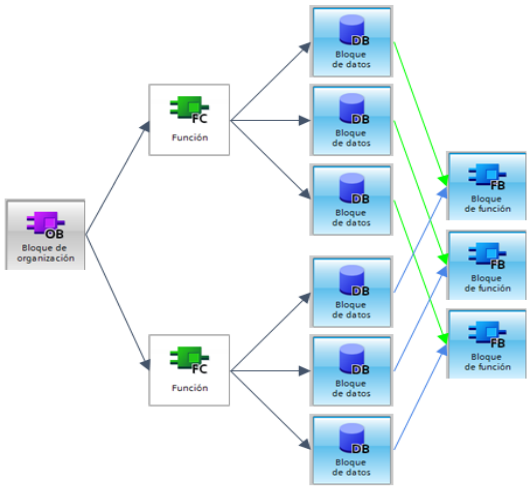
La segunda alternativa consiste en hacerlo de forma indirecta y que el OB1 llame a un único FC que se encarga de llamer a todas las instanciaciones de los FBs



El Ob1 llama a un único FC, Este puede instanciar los FB o pasar Todos sus parámetros en forma de parámetros del FC.

Ilustración 12 Llamada a las instanciaciones con un FC y este a su vez desde el bloque de organización

La última alternativa que el OB1 llame a tantos FCs como módulos tiene la máquina modular y son estos FCs son los que se encargan de llamar a su vez a cada una de las instanciaciones dentro de su modulo.



Cada instanciación de un bastidor o máquina pasan a través de un FC.

Al igual que en los casos anteriores el FC puede asignar las variables o pasarlas como parámetros y ser el OB quien los asigna.

Ilustración 13 Llamada a las instanciaciones desde diferentes FC y estos a su vez desde el bloque de organización

Se recogen a continuación en una tabla, las ventajas y desventajas de las tres alternativas .

	Ventajas	Desventajas
Llamada desde el OB	No hay intermediarios entre las llamadas y el OB	Si hay código que no pertenece a ningún módulo no existe distinción
Llamada desde un único FC	Se diferencian las llamadas inherentes del sistema y las de los módulos de la máquina	No se pueden identificar los módulosa través de sus llamadas
Llamada desde un FC por modulo	A nivel de organización es más clara que cada módulo llame a sus propias instancias	Al existir separación entre los módulos, dificultar asignar el FB a un módulo

En cuanto a las llamadas a los bloques funcionales, se ha elegido la segunda alternativa, que consiste en la llamada desde un único FC, por tratarse de la opción más equilibrada y que tiene la ventaja de subordinar la llamada de los bloques funcionales.

3.5.3 Solución adoptada

3.5.3.1 Modelado

En este apartado se presenta el modelado propuesto para la maquina modular, teniendo en cuenta la problemática que presenta la modularización en el capítulo Discusión general del problema a resolver. El modelo se basa en dos vistas, en la primera se definen los módulos y la relación que existe entre ellos y en la segunda (vista de implementación), se define el hardware y el software. Dentro del hardware podemos encontrar los dispositivos de control, la configuración de sus conexiones, etc. y ejemplos del software de control pueden ser bases de datos, funciones, variables, etc.

3.5.3.2Diseño de Vista funcional del Metamodelo

En la vista funcional se presentan los diferentes elementos que pueden componer una máquina modular. En la Ilustración 14 se aprecia que el dispositivo puede tener un conjunto de diferentes señales (representa los sensores, actuadores, intercambio de información). A su vez un módulo puede tener diferentes dispositivos y también un módulo puede tener otros módulos.

El Metamodelo consigue una estructura jerárquica gracias a la recursividad que existe entre los módulos. Esta jerarquía permite establecer diferentes grados de granularidad resolviendo de esta forma uno de las problemáticas de la modularización.

Se definen cuatro reglas de composición.

1. De cada una de las bifurcaciones el modulo del nivel más bajo tiene que tener al menos un dispositivo.
2. Cada elemento, solo puede pertenecer a un único elemento.
3. Un módulo solo puede contener módulos de un nivel inferior
4. Solo existe un elemento con un PLC y este tiene que ser la raíz del resto de módulos

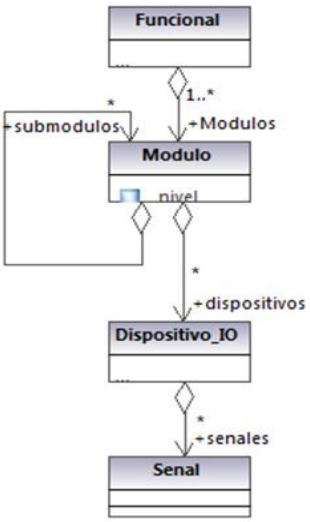


Ilustración 14 Metamodelo de la Vista Funcional

3.5.3.3Diseño del Modelado funcional de la máquina

En este apartado se presenta una vista funcional mostrada anteriormente para una empresa que se dedica al montaje de líneas o partes de líneas de producción donde podemos encontrar diferentes máquinas. Estas máquinas están compuestas por un conjunto de bastidores que sirven de soporte para diferentes estaciones en las que se pueden realizar una o más de las operaciones de la línea. Cada máquina está compuestas a su vez por unas botoneras que permiten el mando de las máquinas, un armario donde se encuentran las protecciones de la máquina como son los magneto térmicos y el conjunto de señales que no pertenecen a los grupos anteriores. Opcionalmente las máquinas pueden tener un transporte como medio de distribución entre las diferentes estaciones de la maquina o un HMI para complementar el control de la botonera y supervisión.

Se ha optado por esta granularidad, dado que aunque la empresa hace máquinas a medida, existen elementos comunes a diferentes maquinas como pueden ser la botonera o el armario lo que permite una gran reutilización

El Metamodelo de la maquina descrita se presenta en la Ilustración 15.

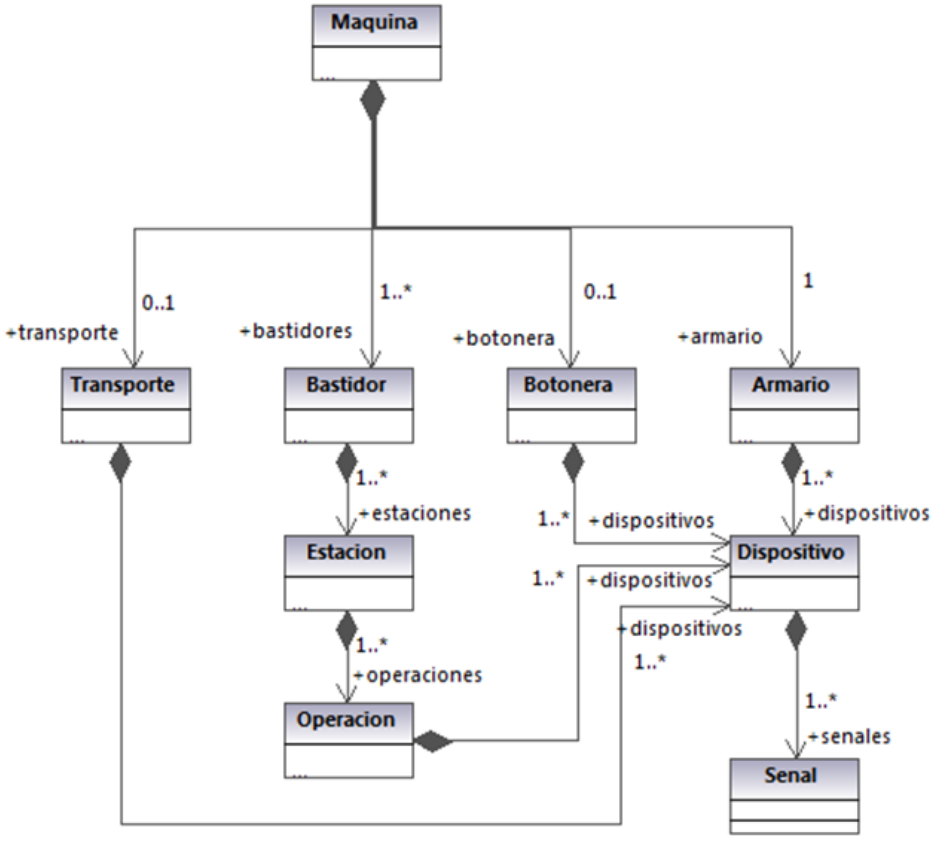


Ilustración 15 Metamodelo Funcional de un tipo de máquina

El Metamodelo resultante tiene 4 niveles, apreciándose la particularidad de los módulos bastidor y estación que no tienen dispositivos. Al no ser los últimos de su rama, el Metamodelo de máquina cumple con las reglas de composición. Podemos sacar la conclusión que tanto el bastidor como la estación siempre tienen que tener los módulos de su nivel inferior. Esto ocurre porque esta división tiene una labor documental y estructural para la definición de los tipos de máquinas. Permitiendo diferenciar dos máquinas con las mismas operaciones pero que son diferentes a nivel estructural.

3.5.3.4Diseño de la vista de implementación del Metamodelo

La vista de implementación del Metamodelo representada en la Ilustración 16. Se define siguiendo la estructura adoptada por Siemens para el software de proyectos de automatización dentro de TIA Portal como se puede observar en la Ilustración 19.

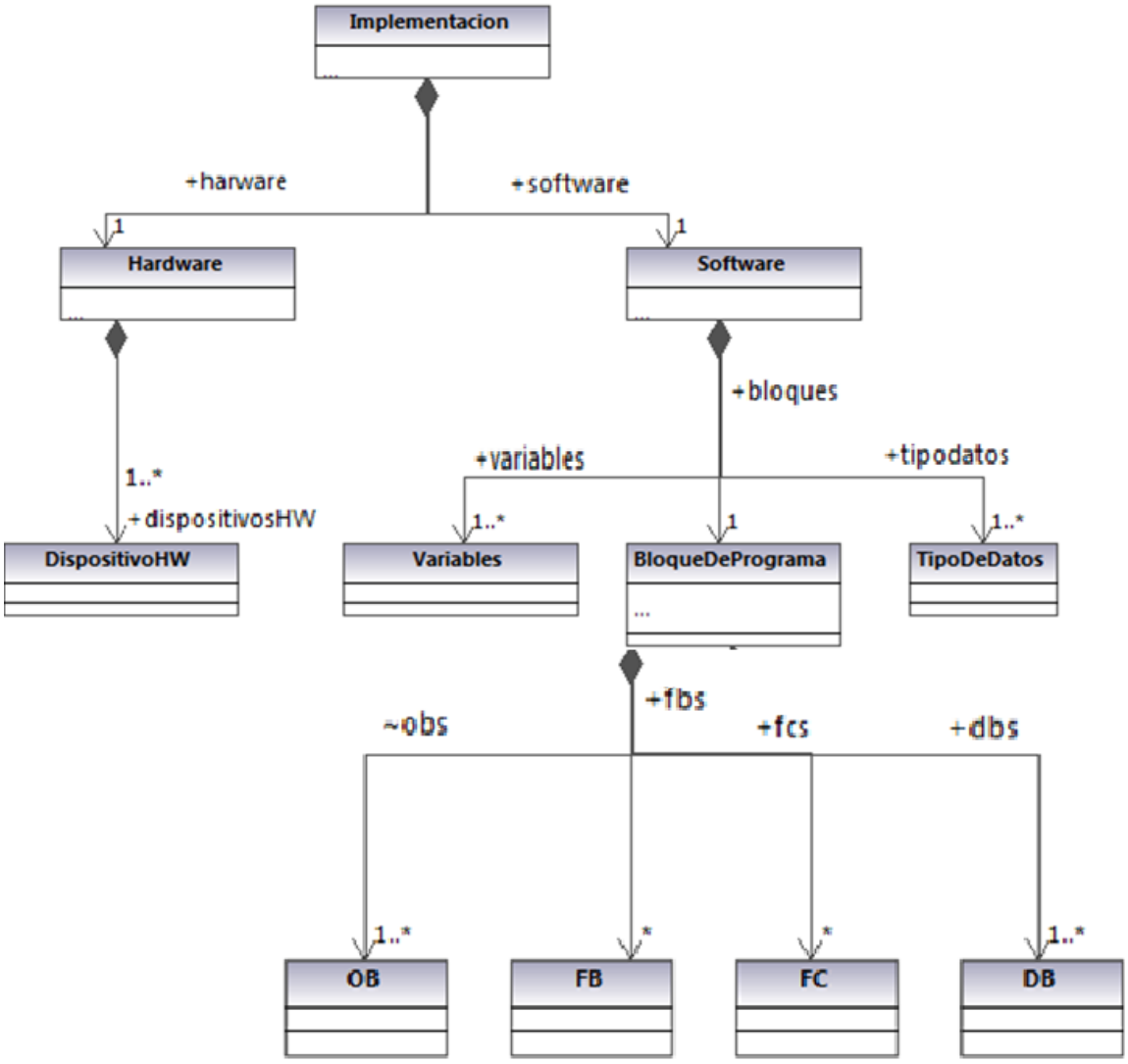


Ilustración 16 Metamodelo de la implementación

Se ha caracterizado un dispositivo de una red Profinet. Esta caracterización consta de un nombre, el nombre Profinet y la dirección IP (necesarios para configurar la red) así como la dirección inicial de señales de E/S. Estas propiedades de dispositivos hardware se representan en la **Error! Reference source not found.**.

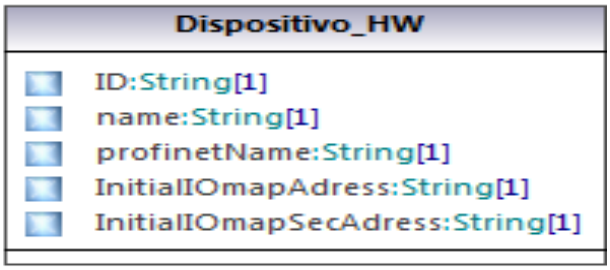


Ilustración 17 Dispositivo HW

La máquina modular laforman las vistas del Metamodelo y las relaciones entre ellas. La relación existente se da entre los dispositivos hardware de la vista de implementación y el dispositivo IO de la vista funcional. Concretamente quedan ligados en las tablas de variables donde se relacionan las direcciones de entrada y salida de los dispositivos hardware con las variables utilizadas por los bloques funcionales

3.5.3.5 Diseño de la aplicación de generación automática de un proyecto de automatización.

La Ilustración 18 Arquitectura general representa la arquitectura de alto nivel de la aplicación de generación.

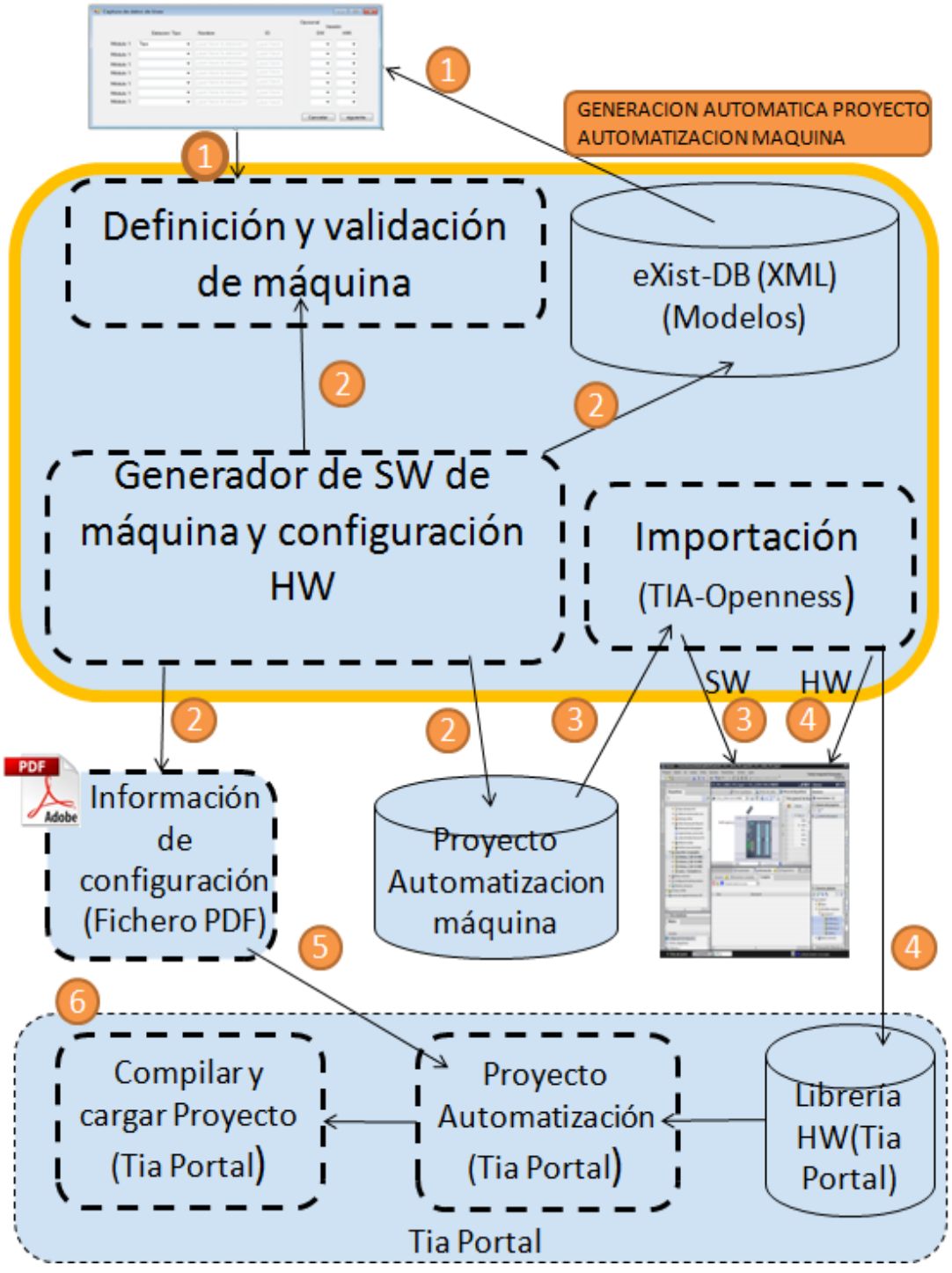


Ilustración 18 Arquitectura general

La aplicación de generación de un proyecto de automatización está dividida en una serie de fases: se comienza definiendo una nueva máquina que está compuesta por un conjunto de módulos con diferentes funcionalidades. Por cada módulo que compone la máquina (y que puede contener una jerarquía de módulos) se dispone de un proyecto TIA Portal que corresponde a su modelo de implementación. La aplicación debe garantizar que se respeten que etiquetas de código, variables y direcciones Profinet duplicadas sean unicas y generar el código final.

Se ha definido una base de datos orientada a modelos que contiene todos los proyectos de automatización tipo correspondiente a módulos funcionales, así como las nuevas máquinas definidas.

La base de datos debe de ser tipo NoSQl, y XML nativa lo que la hace adecuada para el manejo y almacenamiento de modelos en formato XML, que es el formato en el que se almacena la información de los modelos.

Una de las características de estas bases de datos es que no se necesita especificar un XML schema de la información que se almacena, lo cual da mucha flexibilidad para introducir diferentes tipos de documentos, y para futuras ampliaciones. Esto no significa que no se puedan hacer validación de los documentos XML que se introduzcan.

La elección de una base de datos XML nativa se debe a que la herramienta TIA Portal ofrece TIA Portal Openness, un API que permite acceder a la estructura de la herramienta TIA Portal, de forma remota y modificar el contenido de la misma. De esta forma se pueden generar proyectos de automatización utilizando objetos almacenados en librerías u objetos externos importados en XML.

También, se puede acceder a los datos de proyectos para su posterior procesamiento, extraer datos estadísticos, realizar copias de seguridad o actualizar el contenido de los proyectos.

Por otro lado, un proyecto de automatización TIA Portal contiene toda la información relacionada con todos los controladores del proyecto. Por cada controlador se genera una estructura de directorios, tal y como se ilustra en la Ilustración 19. En: la carpeta de "bloques de programa", en la cual se organizan todos los bloques de programa del PLC (OB, FB, FC, DB), la carpeta "variables PLC", donde están declaradas las variables globales del programa del PLC y la carpeta de "tipos de dato de PLC", en la que se pueden definir tipos de dato de usuario específicos para la aplicación. Estas carpetas se generan automáticamente vacías en TIA Portal cuando se inserta un PLC y se va introduciendo el contenido a medida que se va desarrollando el proyecto.

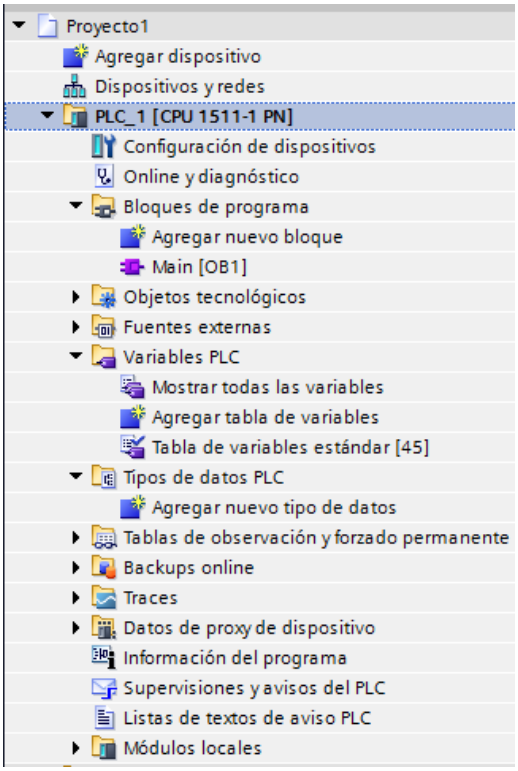


Ilustración 19 Árbol de directorios de PLC en TIA Portal

A continuación se describen los pasos que ejecuta la aplicación de generación:

Captura de datos de definición de la máquina

Se define una aplicación que permite, que el operador defina las características de la máquina. Que disponga de un interfaz de usuario que accede a la base de datos de la que se extraen los proyectos tipo definidos, que pueden utilizarse en la composición de la máquina. Esto corresponde al paso 1 de la Ilustración 18 Arquitectura general.

Base de datos orientada a modelos

Una vez definida la máquina a partir de módulos funcionales, la aplicación accede a la base de datos orientada a modelos que contiene los modelos de implementación que siguen la estructura TIA Portal correspondientes a los módulos funcionales tipo, paso 2 de la Ilustración 18 Arquitectura general.

La Ilustración 20 representa las colecciones de la base datos para el caso de la máquina tipo definida en la sección 2. Contiene los modelos de implementación (bastidor, transporte, botonera, armario y HMI) de los que se van a componer las máquinas. Cada modelo representa la información relativa a un proyecto de automatización con toda la estructura hardware y software, siguiendo el meta-modelo. Los proyectos de automatización de las nuevas máquinas definidas, se almacenan como proyectos máquinas tipo para su posterior reutilización.

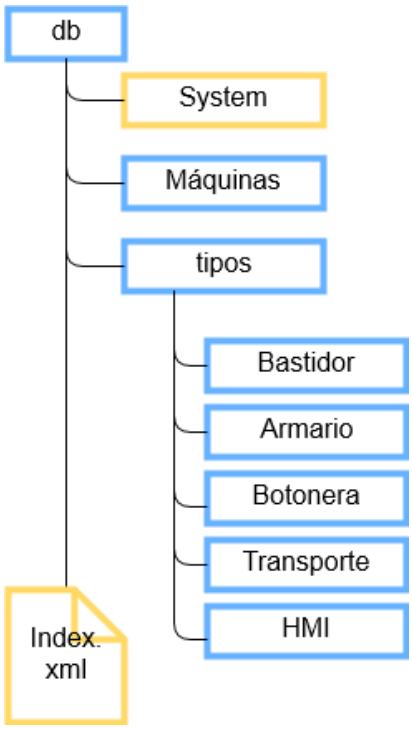


Ilustración 20 Estructura de la base de datos de modelos de implementación

El proyecto Máquina se genera a partir de los proyectos tipo de los módulos que la componen.

Esta parte de la aplicación se encarga de adaptar el código y los datos de cada proyecto tipo, para asegurar que las instanciaciones en el código generado sean correctas, paso 2 de la Ilustración 18.

Esta operación se realiza en las siguientes fases:

- Fase 1. Se dispone del archivo de definición de la máquina en formato XML, generado en el paso 1. La aplicación extrae de él la información necesaria para seleccionar de la base de datos los proyectos tipo con los que se va a construir la máquina.
- Fase 2. Adaptación de las variables globales (Tipos de datos, DB globales, Tablas de variables) de aquéllos módulos tipo con más de una instancia. A modo de ejemplo, la Ilustración 21 representa las modificaciones que habría que realizar en una tabla de variables en cada instancia del mismo módulo.

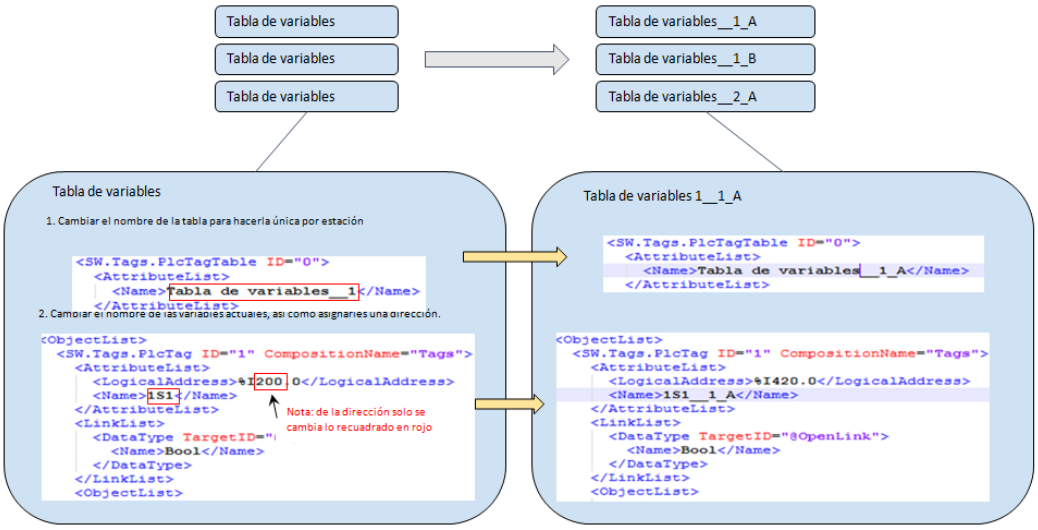


Ilustración 21 Modificación de la tabla de variables

Fase 3. Modificación de los nombres de los parámetros actuales que se utilizan en las instanciaciones de los módulos de programa, haciendo uso de las variables declaradas en las tablas de variables como se puede observar en la Ilustración 22 y la Ilustración 23.

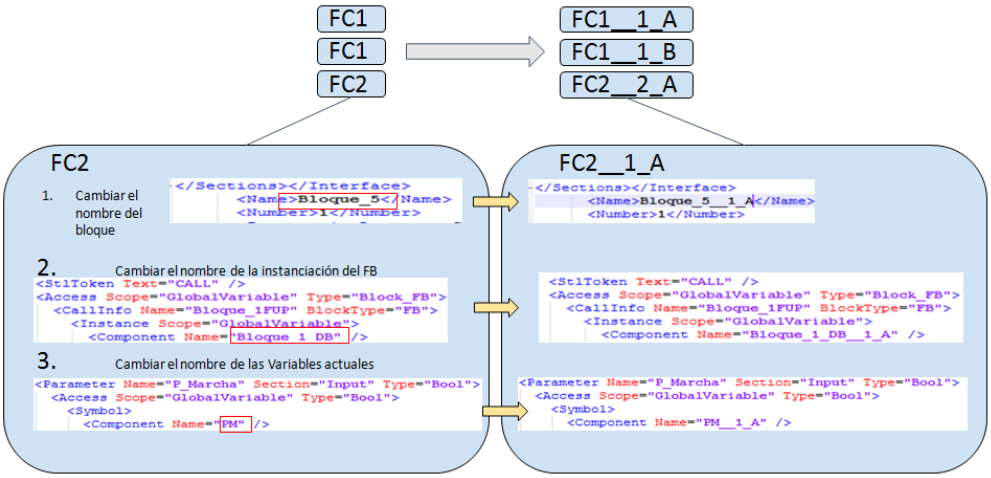


Ilustración 22 Modificación de las FC

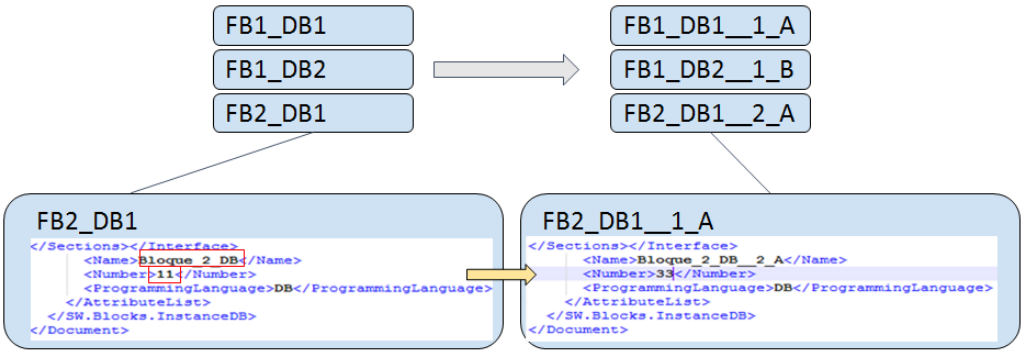


Ilustración 23 Modificación de las instanciaciones de los FB

Fase 4. Generación del modelo de implementación del proyecto de la máquina integrando en los OBs del mismo tipo de ejecución, el contenido de los OBs de cada módulo tipo. En el modelo final solo existe un OB de cada tipo de ejecución que integrará el código de todos los proyectos tipo. En la Ilustración 24 se indica la forma de proceder para la generación de los OBs.

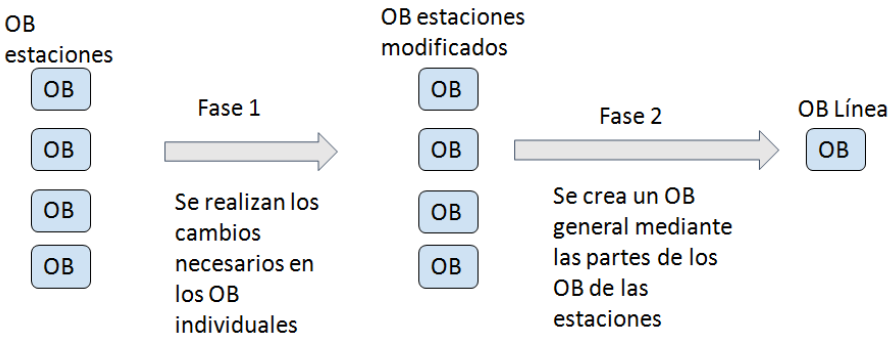


Ilustración 24 Pasos de la generación del OB1

Fase 5. Generación de un documento PDF con la documentación necesaria para la configuración del hardware y del sistema de comunicaciones de la máquina.

Generación del proyecto en TIA Portal

El modelo de implementación de la máquina generado siguiendo el procedimiento descrito en el apartado 3.3 se importa a la herramienta TIA Portal utilizando las funciones del API TIA Portal Openness, paso 3 de la Ilustración 18.

La importación se realiza en dos fases:

- Fase 1. Utiliza la función de grupos para crear bloques, variables y tipos en una estructura de carpetas. La estructura de carpetas del proyecto generado en TIA Portal sigue la estructura del modelo funcional.
- Fase 2. Importa el software del proyecto generado descrito en el apartado 3.3 a las diferentes carpetas del proyecto.

El hardware relacionado con los proyectos tipo se encuentra almacenado en la librería global de TIA Portal. Para completar el proyecto de automatización que se está generando, es necesario importar en el mismo el hardware de cada componente mecatrónico, paso 4 de la Ilustración 18 Arquitectura general

TIA Portal Openness dispone de funciones que permite importar hardware almacenado en librerías TIA Portal al proyecto de automatización generado.

El operador configurará el hardware y las comunicaciones de forma manual en el entorno TIA Portal. La información necesaria para realizar esta configuración la genera

la aplicación, a partir del proyecto generado, en un fichero PDF. Corresponde al paso 5 de la Ilustración 18.

Una vez completados los pasos anteriormente indicados se tendrá el proyecto completo generado en TIA Portal. Quedaría por último la compilación del mismo y la carga en el PLC, paso 6 de la Ilustración 18

3.6 Análisis de riesgos

En este apartado se recoge de forma resumida el resultado de la identificación y análisis de los riesgos potenciales que pueden afectar a la consecución de los resultados del trabajo y a su aplicación. En la evaluación se ha tenido en cuenta la probabilidad de que se materialicen y el impacto que producirían en los resultados finales.

Se listan a continuación los riesgos más relevantes y su tratamiento:

1. Fallo de la retrocompatibilidad que afecte a las distintas herramientas (TIA Portal o cualquiera de sus módulos, TIA Portal Openness, Visual Studio, XML Spy, Windows, …)
   Tratamiento: Aceptar

2. Cambio de especificaciones de modularidad por parte de la empresa fabricante (que los módulos diseñados para las máquinas de fugas no sirvan para otras)
   Tratamiento: Adaptar el programa

3. Falta de preparación técnica de las personas que manejan el nuevo módulo
   Tratamiento: Impartir un curso de capacitación

4. Falta de preparación técnica de las personas que elaboran los proyectos tipo
   Tratamiento: Impartir un curso de capacitación
5. Falta de experiencia por parte de la empresa en trabajo multidisciplinar de desarrollo máquinas modulares
   Tratamiento: Reorganizar el trabajo y establecer procedimientos

6. Dificultad de realizar proyectos modulares
   Tratamiento: Impartir Cursos de formación y establecer procedimientos de trabajo

7. El tiempo empleado con la solución modular sea superior al de partida Tratamiento: Aceptar

8. Uso indebido de la aplicación desarrollada
   Tratamiento: Emplear técnicas poka-yoke en la aplicación

9. Cierre de la plataforma Openness a usuarios no acreditados por Siemens
   Tratamiento: Solicitar acreditación a Siemens

10. Tecnología nueva y en constante desarrollo
    Tratamiento: Aplicar Vigilancia tecnológica a TIA Portal Openness

Las acciones propuestas están orientadas a evitar, transferir, reducir o aceptar los riesgos identificados.

4 METODOLOGÍA

En esta sección se mencionan en primer lugar los medios empleados en el trabajo, describiéndose a continuación la metodología que se ha empleado para el desarrollo del software y finalmente se presentan los productos finales del diseño (entregables), que se han ido generando a lo largo del mismo.

4.1 Medios empleados

Durante el desarrollo del trabajo se han empleado, además de equipos hardware y varias herramientas software. Dentro del hardware se encuentran:

- PLC s7 1500 (Siemens)
- Esclavo Profinet ET 200 (Siemens)
- Módulo neumático CPX (Festo)

Entre las herramientas software están:

- TIA Portal
- TIA Portal Openness
- Visual Studio 2015
- XML Spy
- VM WARE

La aplicación más relevante ha sido TIA Portal Openness, cuyas funcionalidades permiten:

- Manipular objetos de un proyecto (p.e.: carpetas, hardware, bloques de programa, variables, tipos de variables, etc.), tanto del PLC, como del HMI
- Utilizar librerías, tanto globales como del proyecto
- Exportar e importar información relativa al software de proyectos.
- Ejecutar comandos relacionados con tareas del TIA Portal (p.e.: compilación de un proyecto generado previamente y su volcado en el PLC)

4.2 Desarrollo del software

Para el desarrollo del software se ha elegido una metodología ágil y de integración continua, como puede ser SCRUM, ya que es la que más se ajusta a las características del presente trabajo, complejo por ser de innovación, con requisitos cambiantes y resultados que deben generarse en cortos espacios de tiempo, no superiores a las tres semanas.

La metodología requiere la generación de entregables pequeños, que sea individualmente un desarrollo completo y funcional, que formará parte de la aplicación final. Cada entregable puede ser testeado individualmente, lo que facilita una rápida detección y corrección de posibles errores antes de completarse el desarrollo de la aplicación final.

4.3 Entregables

A continuación se presentan los productos finales del diseño (entregables), que se han ido generando a lo largo del mismo, fundamentalmente desarrollos software, como aplicaciones informáticas, bases de datos o entornos gráficos.

Su desarrollo ha sido necesario para materializar en productos software (entregables) la solución adoptada que se ha presentado en el apartado 3 del trabajo. En esta sección se hace una breve descripción de cada uno, explicando su función, y se muestran algunos ejemplos de su uso, indicándose la fase y el paso de la solución adoptada que materializan.

4.3.1 Modelado

El conjunto de reglas, relaciones y arquitectura mostrados en la vista funcional del Metamodelo, vista de implementación del Metamodelo y la particularización de la vista funcional de máquina, se han recogido en tres clases de C#.

4.3.2 Proyectos tipo

Los módulos que componen una máquina están compuestos de proyectos tipo, que son proyectos de automatización de TIA Portal. Los proyectos tipo deben cumplir un conjunto de reglas, para que el generador de código pueda acceder a las informaciones, que se listan a continuación:

- Uso de una única CPU
- Todos los dispositivos de periferia descentralizada en la carpeta de no agrupados
- Método de paso de parámetros, mediante el método directo
- Llamad de funciones con un único FC
- Bloques de programas y variables comunes en una carpeta de compartidos
- Comunicaciones mediante Profinet

Para que los proyectos se puedan importar, deben seguir una estructura determinada, por lo que se ha definido un Schema de proyecto tipo que sigue la estructura de proyectos dentro de TIA Portal, como puede verse en la Ilustración 19.

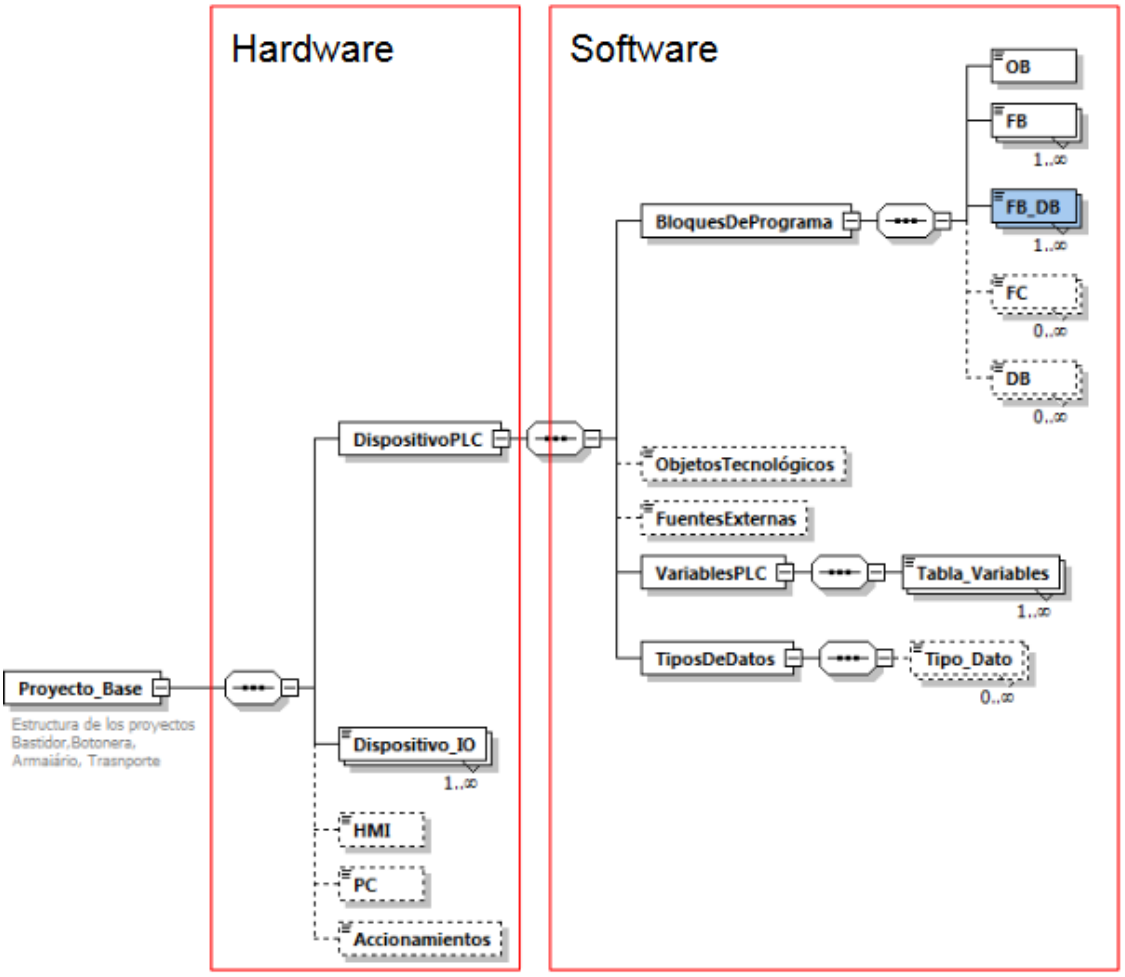


Ilustración 25 Schema de proyecto tipo

4.3.3 Manual de uso de librerías globales de TIA Portal para el apartado hardware de la aplicación

Se ha elaborado el procedimiento para que dentro de la herramienta TIA Portal se almacene el hardware de los diferentes proyectos dentro de las librerías globales, dado que TIA Portal Openness no permite ni la creación de dispositivos hardware, ni su exportación. Por ello para poder reutilizar los diferentes elementos de hardware se ha decidido almacenarlos dentro de las librerías que proporciona TIA Portal. Se adjuntan a continuación dos ejemplos del Manual.

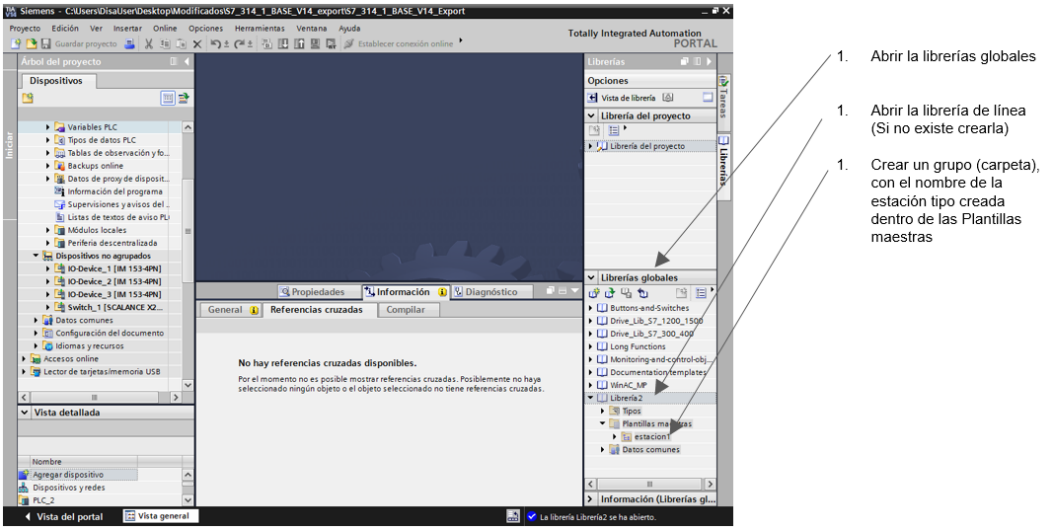


Ilustración 26 Ejemplo de guardado del manual de uso de librerías globales TIA Portal

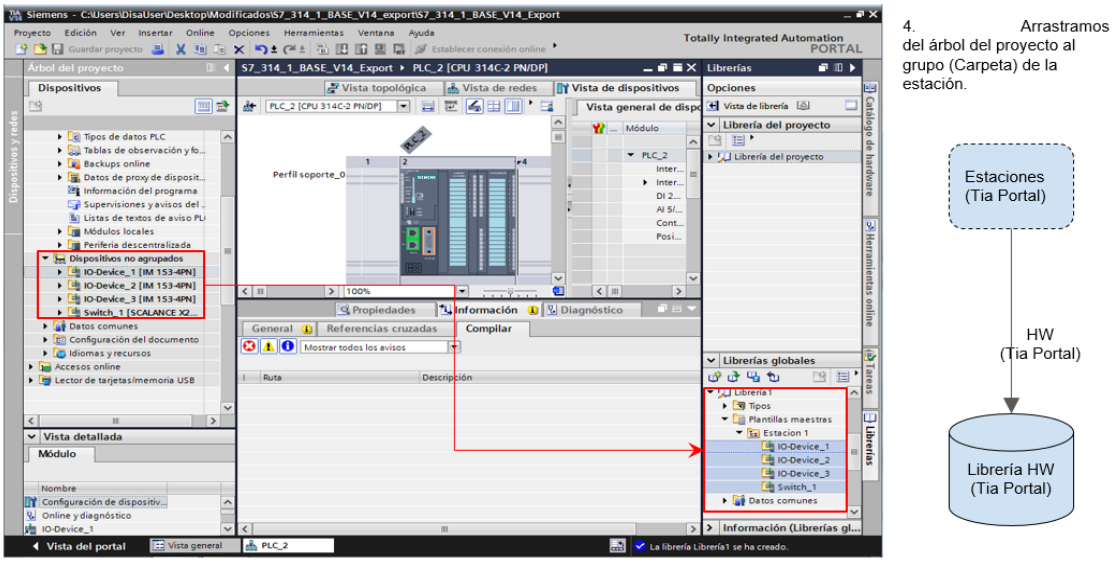


Ilustración 27 Ejemplo distribución del hardware tras el guardado del manual de uso de librerías globales TIA Portal

4.3.4 Aplicación personalizada de interactuación con TIA Portal

Esta aplicación es la utilizada por el generador de código en los pasos 3, 4 de la arquitectura general de la solución adoptada y también podrá ser utilizada por otras aplicaciones que requieran de la integración con TIA Portal.

Todas su funciones se han desarrollado utilizando instrucciones **USING**, porque garantizan la liberación de los recursos utilizados dentro de ellas, incluso si surgen imprevistos o excepciones del programa. Esto es especialmente útil cuando se trabaja con productos limitados, como el caso de TIA Portal, ya que no pueden utilizarse todas sus funciones mientras esté en funcionamiento la aplicación que interactúa con TIA Portal.

Como medida de adicional de seguridad, dentro de la instrucciones **USING**, se han utilizado instrucciones **EXCLUSIVE ACCESS**, que evitan que otro usuario pueda manipular TIA Portal mientras esté en funcionamiento la aplicación que interactúa con TIA Portal.

Dentro de las dos instrucciones anteriores, se ha utilizado la instrucción **TRANSACTION**, que solo guarda los cambios realizados dentro de TIA Portal tras la confirmación de que se han realizado con éxito y en caso de no confirmarse se deshacen automáticamente todos los cambios. Con este último control, si surgen imprevistos o excepciones del programa, en mitad de la interaccion con TIA Portal, no causarían ni gún efecto negativo.

La aplicación personalizada que desarrollado a lo largo de este trabajo dispone, entre otras de las siguientes funcionalidades:

- Interacción con TIA Portal
  - Abrir, cerrar
  - Conectarse, desconectarse a una instancia de TIA Portal
- Interacción con proyectos de TIA Portal
  - Abrir, cerrar
  - Crear, borrar
  - Exportar, importar (XML)
  - Conversión de proyectos a la última versión
  - Compilación
- Interacción con librerías de TIA Portal
  - Abrir, cerrar
  - Crear, borrar
  - Exportar, importar de TIA Portal
  - Copiar entre librería global y local de proyecto
- Interacción con interfaz gráfica de TIA Portal

- Cambio entre las diferentes pantallas que puede ver el usuario dentro del programa de TIA Portal (p.e.: entrar en la interfaz gráfica de dispositivos y redes

4.3.5 Aplicación para modificar los proyectos de automatización en formatos XML

Esta aplicación es la utilizada por el generador de código en el paso 2 de la arquitectura general de la solución adoptada, siendo XML es la tecnología seleccionada para realizar la generación de código.

Se ha desarrollado un programa que actúa como librería y que además dispone de interfaz gráfica para poder utilizarlo de forma independiente, con el fin de agilizar el desarrollo de aplicaciones.

La aplicación modifica los siguientes ítems dentro de un proyecto de automatización en formatos XML:

- Nombres de dispositivos
- Nombres tablas de variables
- Nombres de las variables
- Direcciones de las variables
- Nombres de DBs
- Parámetros de las instanciaciones de los FBs
- Llamadas de FB_DBs
- Nombres de los FC
- Llamadas de FC
- Nombres de los DB
- Llamadas de DBs
- Llamadas del OBs

Cabe destacar que las moficaciones se han creado para tres lenguajes de programación, como se muestra en las siguientes ilustraciones.

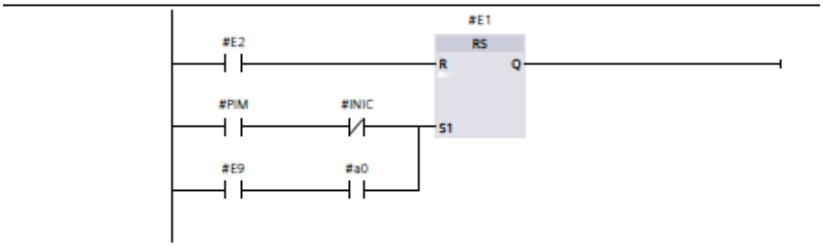


Ilustración 28 Bloque de programa número 10 en diagrama de contactos

```
</Sections></Interface>
      <Name>Bloque_1Graph</Name>
      <Number>10</Number>
      <ProgrammingLanguage>LAD</ProgrammingLanguage>
    </AttributeList>
    <ObjectList>
      <SW.Blocks.CompileUnit ID="1" CompositionName="Compil
        <AttributeList>
          <NetworkSource><FlgNet xmlns="http://www.siemens.
  <Parts>
    <Access Scope="LocalVariable" Type="Bool" UId="21">
      <Symbol>
        <Component Name="a0" />
      </Symbol>
    </Access>
    <Access Scope="LocalVariable" Type="Bool" UId="22">
      <Symbol>
        <Component Name="b0" />
      </Symbol>
    </Access>
    <Access Scope="LocalVariable" Type="Bool" UId="23">
      <Symbol>
        <Component Name="a1" />
      </Symbol>
```

Ilustración 29 XML de bloque de programa numero 10 en diagrama de contactos

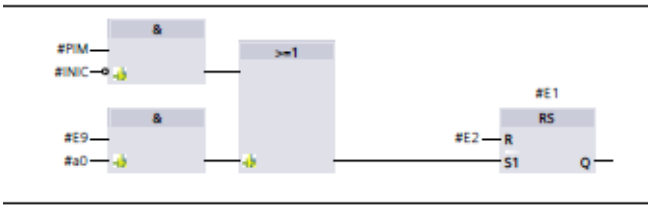


Ilustración 30 Bloque de programa número 10 en bloques funcionales

```
    <Name>Bloque_1FUP</Name>
    <Number>10</Number>
    <ProgrammingLanguage>FBD</ProgrammingLanguage>
  </AttributeList>
  <ObjectList>
    <SW.Blocks.CompileUnit ID="1" CompositionName="CompileUn
      <AttributeList>
        <NetworkSource><FlgNet xmlns="http://www.siemens.com
<Parts>
  <Access Scope="LocalVariable" Type="Bool" UId="21">
    <Symbol>
      <Component Name="a0" />
    </Symbol>
  </Access>
  <Access Scope="LocalVariable" Type="Bool" UId="22">
    <Symbol>
      <Component Name="b0" />
    </Symbol>
  </Access>
  <Access Scope="LocalVariable" Type="Bool" UId="23">
    <Symbol>
      <Component Name="a1" />
```

Ilustración 31 XML de bloque de programa número 10 en bloques funcionales

```
0001        A     #E2
0002        R     #E1
0003        A(
0004        A     #PIM
0005        AN    #INIC
0006        O
0007        A     #E9
0008        A     #a0
0009        )
0010        S     #E1
0011        NOP 0
```

Ilustración 32 Bloque de programa número 10 en lista de instrucciones

```
</Sections></Interface>
      <Name>Bloque_1STL</Name>
      <Number>10</Number>
      <ProgrammingLanguage>STL</ProgrammingLanguage
    </AttributeList>
    <ObjectList>
      <SW.Blocks.CompileUnit ID="1" CompositionName
        <AttributeList>
          <NetworkSource><StatementList xmlns="http
  <StlStatement>
    <StlToken Text="A" />
    <Access Scope="LocalVariable" Type="Bool">
      <Symbol>
        <Component Name="a0" />
      </Symbol>
    </Access>
  </StlStatement>
  <StlStatement>
    <StlToken Text="A" />
    <Access Scope="LocalVariable" Type="Bool">
      <Symbol>
        <Component Name="b0" />
      </Symbol>
    </Access>
  </StlStatement>
  <StlStatement>
```

Ilustración 33 XML de bloque de programa número 10 en lista de instrucciones

El programa desarrollado realiza los cambios en los puntos objetivos a lo largo de todo el documento XML y además dispone de una interfaz gráfica en la que se visualizan el XML antes y después permitiendo el ajuste fino de la programación del generador de código, facilitando el trabajo al desarrollador.

Cabe señalar que para el programa se ha elegido la metodología DOM, frente a SAX, ya que la primera permite la accesibilidad a los diferentes nodos por nombre y por atributos de forma no secuencial.

Se muestran a continuación algunas características del programa desarrollado.

- Búsqueda integrada de archivos XML

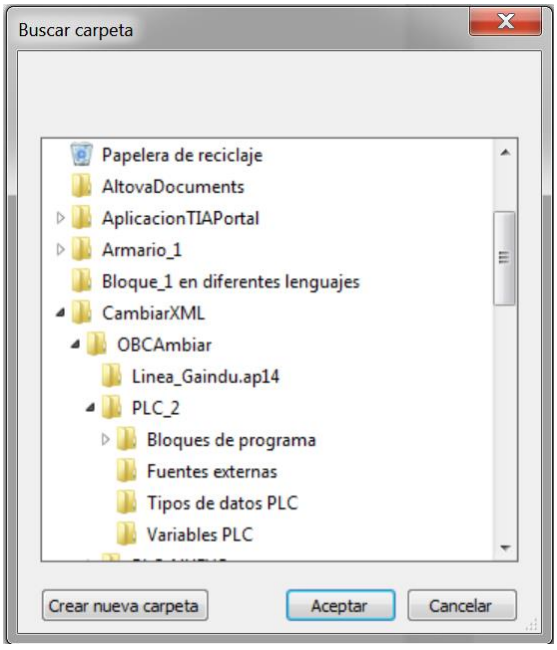


Ilustración 34 Búsqueda integrada de archivos XML de la aplicación modificadora de proyectos

- Dentro de los bloques OB: Cambio de nombre OB, Cambio de nombre de llamada de FC, instanciación de FC y llamada de DB y cambio de nombre de los parámetros que contienen y prefijo de variables actuales. **Implementación de la fase 4 del paso 2 de la arquitectura general de la solución propuesta**, como puede verse en la Ilustración 24

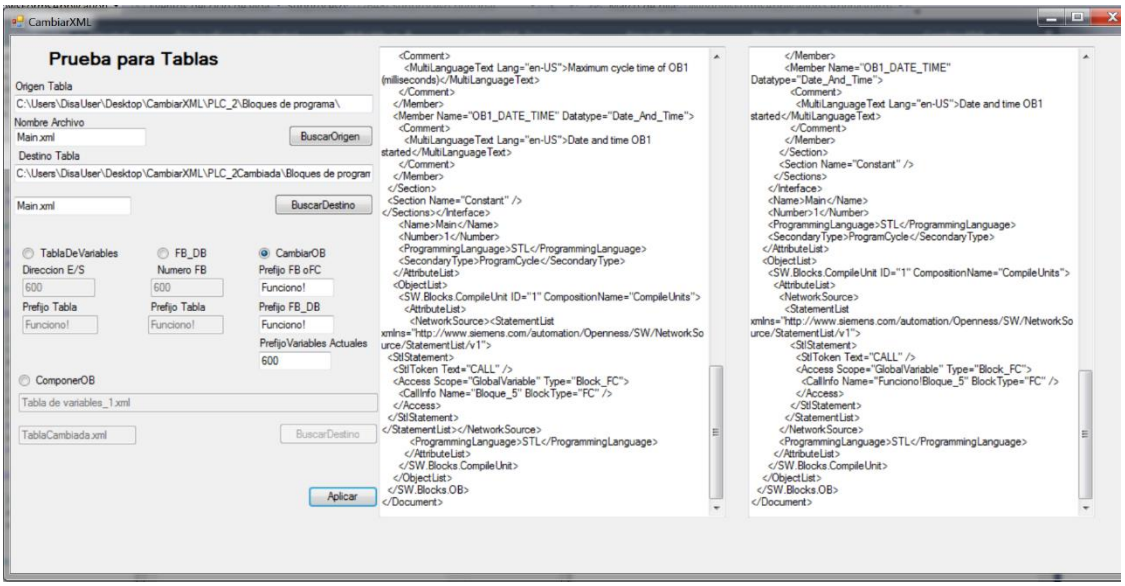


Ilustración 35 Ejemplo de modificación de OB, dentro de la aplicación transformación de proyectos

- Dentro de la tabla de variables: Cambio de nombre Tabla de variables, cambio del nombre y de la dirección de las variables. **Implementación de la fase 2 del paso 2 de la arquitectura general de la solución propuesta**, como puede verse en la Ilustración 21

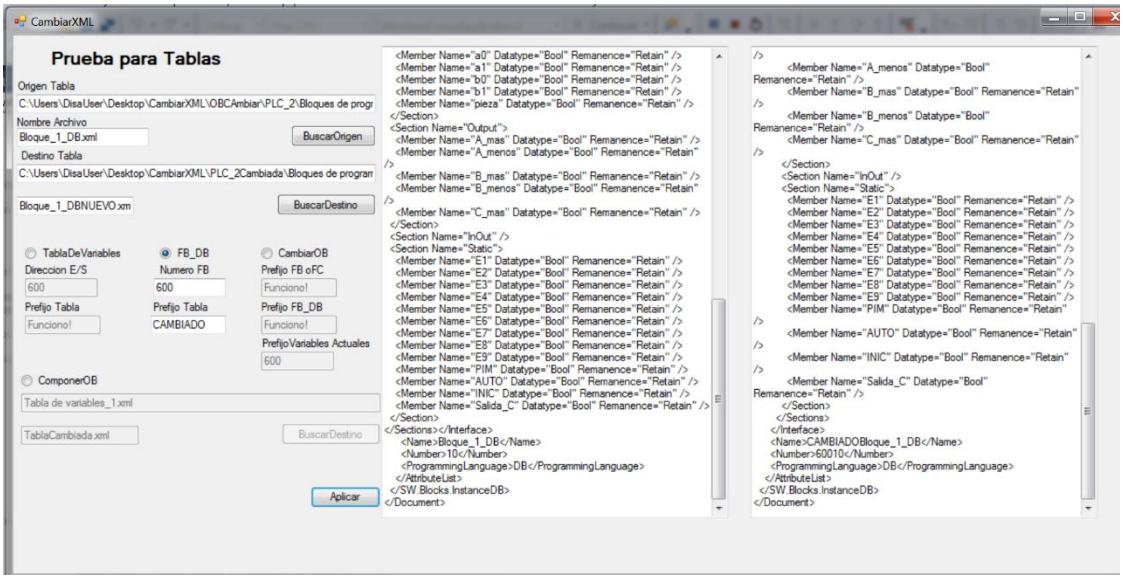


Ilustración 36 Ejemplo de modificación de variables, dentro de la aplicación transformación de proyectos

- Con los bloques OB de los diferentes proyectos tipo se crea un único bloque OB copiando las llamadas a los bloques funcionales de su interior. **Implementación de la fase 4 del paso 2 de la arquitectura general de la solución propuesta**, como puede verse en Ilustración 24

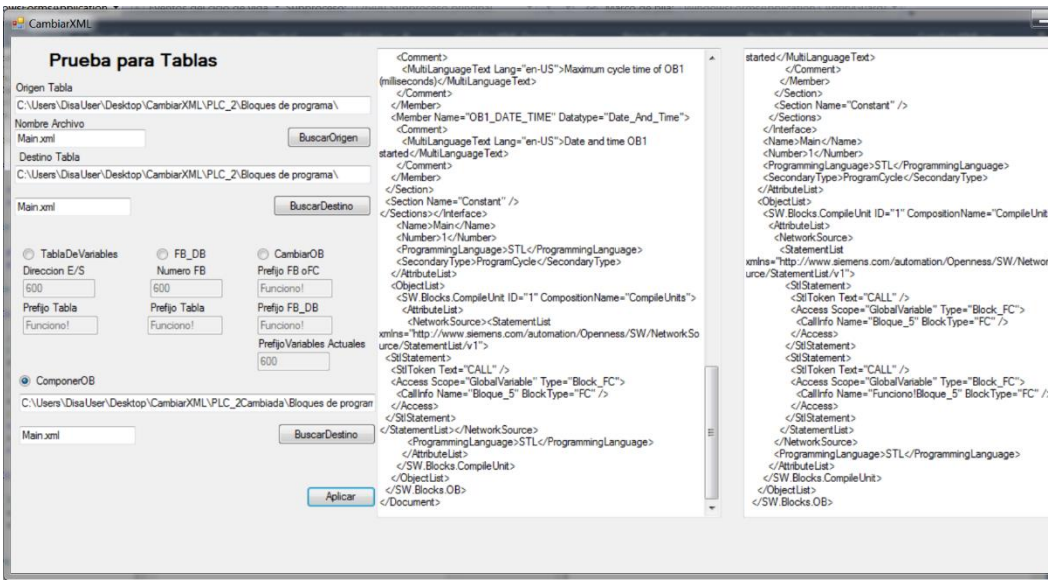


Ilustración 37 Ejemplo de generación de OB, dentro de la aplicación transformación de proyectos

- Dentro de las instaciaciones de los bloques FB: Cambio de nombre FB, cambio del nombre y de la dirección de las variables. **Implementación de la fase 3 del paso 2 de la arquitectura general de la solución propuesta**, como puede verse en Ilustración 23

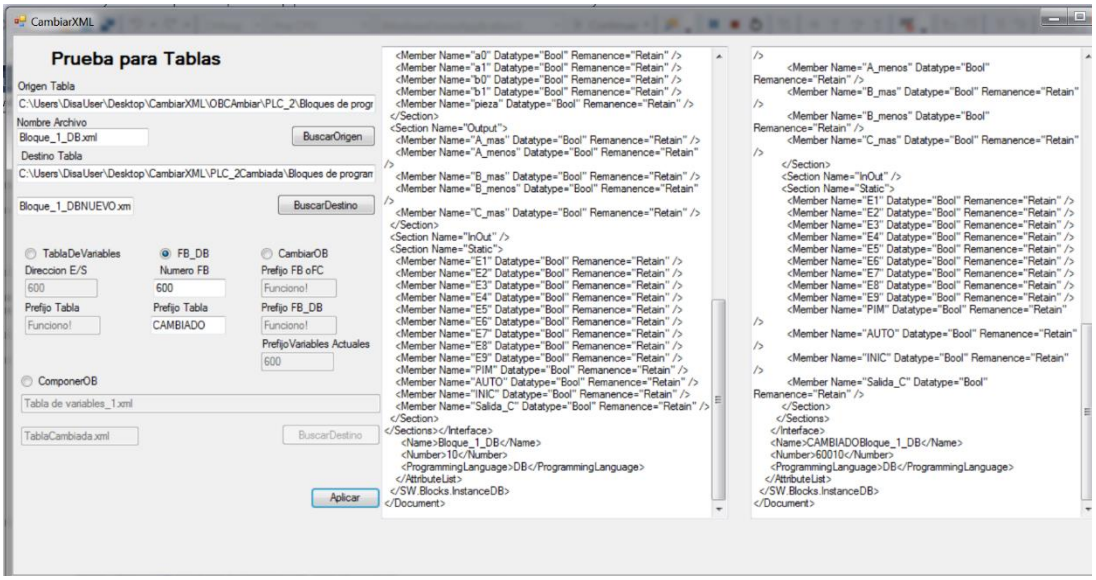


Ilustración 38 Ejemplo de modificación de una instancia de un bloque funcional, dentro de la aplicación transformación de proyectos

Se muestra a continuación un ejemplo de los resultados obtenidos (los cambios aparecen resaltados en cuadros) tras la realización de los cambios de la tabla de variables de un proyecto de automatización, empleando la característica del programa desarrollado que aparece en la Ilustración 36.

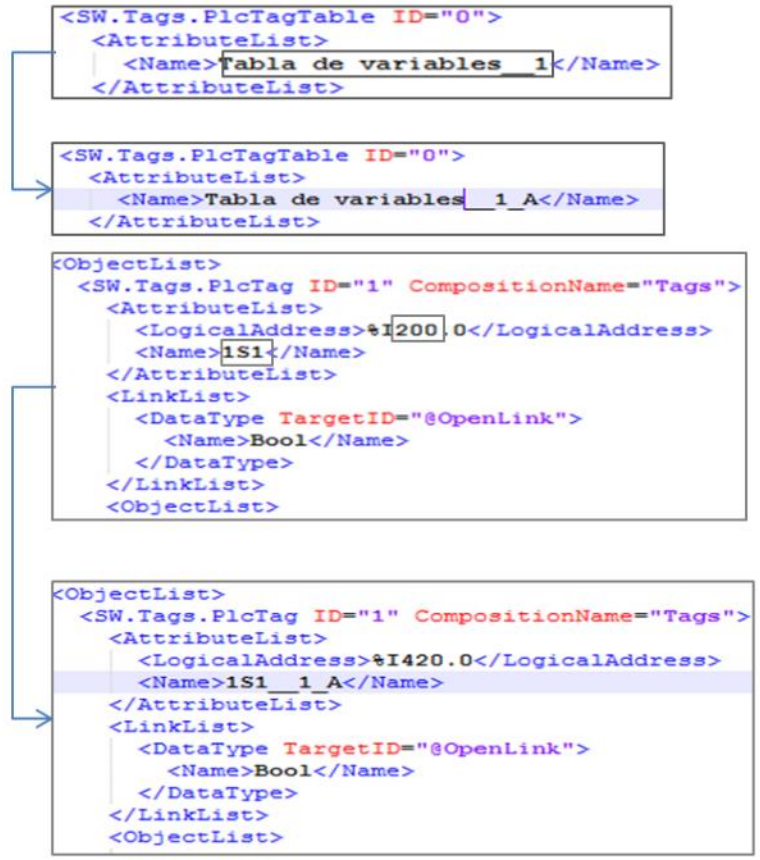


Ilustración 39 Adaptación de la tabla de variables

Se muestra a continuación un ejemplo de los resultados obtenidos tras la realización de los cambios de la tabla de variables de un proyecto de automatización, empleando la característica del programa desarrollado que aparece en la Ilustración 37.

Por último, se muestra un ejemplo de los resultados obtenidos tras la realización de los cambios del OB, en cuanto a la adaptación de sus OBs origen y la generación del OB final, empleando las características del programa desarrollado que aparece en la Ilustración 35 e Ilustración 37.

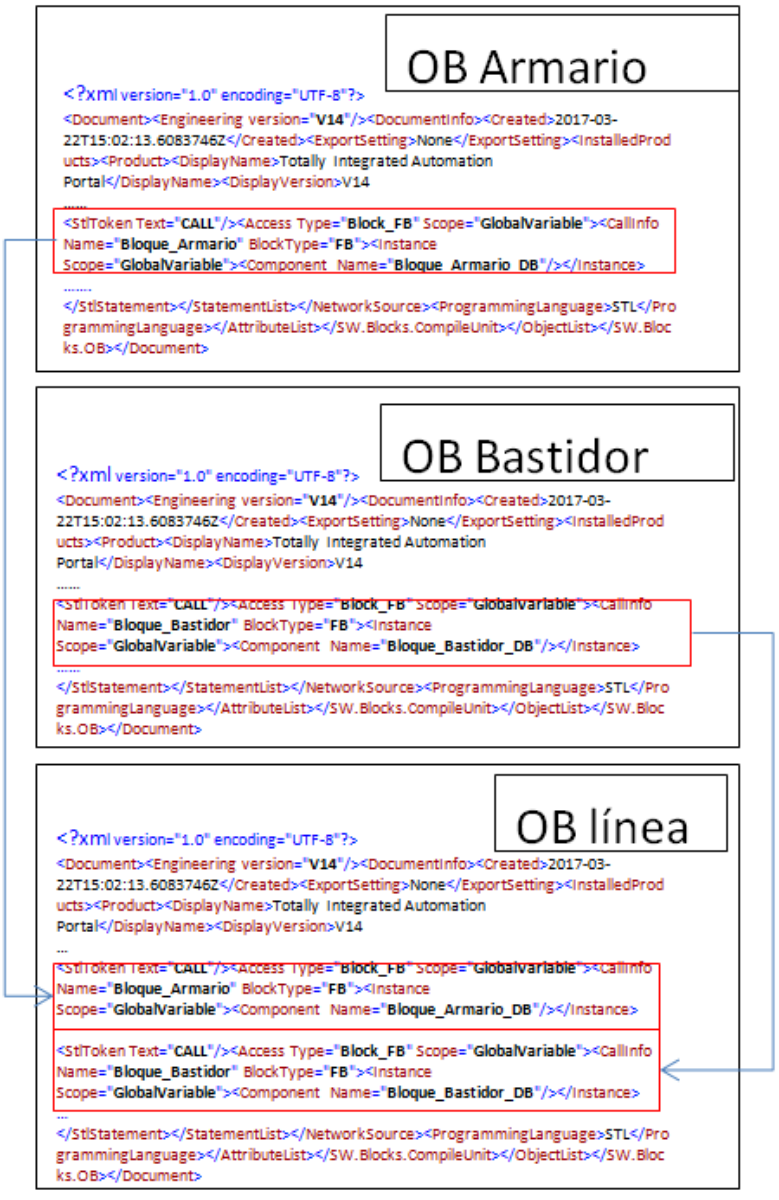


Ilustración 40 Generación del OB1 del proyecto máquina

4.3.6 Base de datos eXist_DB.

La base de datos elegida eXist, es del tipo NoSQl lo que la hace adecuada para el manejo y almacenamiento de modelos en formato XML, que es el formato en el que se almacena la información de los modelos.

Una de las características de esta base de datos es que no se necesita especificar un XML schema de la información que se almacena, lo que le confiere flexibilidad para introducir diferentes tipos de documentos y para futuras ampliaciones. Esto no significa que no se puedan hacer validación de los documentos XML que se introduzcan.

eXist-DB es independiente de la plataforma, dado que está basada en java, al igual que su API. Esta API permite el uso de lenguajes query como Xpaht o Xquery. También permite la administración de usuarios, permisos, indizado…

La administración de los datos se consigue por medio de colecciones de documentos jerárquica, por los que se pueden realizar las acciones sobre conjuntos de documentos. Para actualizar o borrar documentos se puede hacer uso del lenguaje XUpdate, que nos permite actualizar el documento entero o la elección de los nodos a actualizar. También provee de mecanismos de copias de seguridad y restauración (Backup/restare). El motor de la base de datos tiene funciones básicas de seguridad, como pude ser el control de acceso mediante contraseña de los usuarios a los grupos que pertenezcan.

La comunicación con la base de datos se realiza mediante REST lo que permite una comunicación sin estado que ayuda, en caso de pérdida de conexión, a reenviar en un solo envío una solicitud de la información necesaria.

Además de la configuración de la base de datos, se ha creado una aplicación que permite subir los documentos XML, en carpetas concretas. Para ello y con el fin de evitar fallos de comunicación hace el intento de conexión 10 veces. En caso de no alcanzar la conexión aparecerá un error.

Ilustración 41 Logotipo librería eXist

4.3.7 Aplicación de conversión XML y volcado de proyectos tipo en TIA Portal

Se ha desarrollado una aplicación que sirve para la conversión de los proyectos tipo dentro de TIA Portal a archivos XML, haciendo uso de las característica de exportado de la aplicación personalizada de interacción con TIA Portal y posterior volcado de la información en la base de datos de eXist_DB.

El funcionamiento se realiza siguiendo las siguiendo los siguientes pasos:

- Conexión a la base de datos para recoger la información sobre los tipos de módulos que puede contener
- El usuario selecciona el tipo de módulo que quiere volcar en la base de datos y el proyecto tipo desarrollado en TIA Portal
- La aplicación transforma el proyecto en TIA Portal en múltiples archivos XML y los vuelca en la base dd datos siguiendo la estructura que aparece en el Schema de proyecto tipo que aparece en la Ilustración 25

Se ha poblado la librería con diferentes proyectos tipo para validar el diseño del generador de código de la solución adoptada, mostrándose a continuación el interfaz gráfico que se ha desarrollado con tal fin.

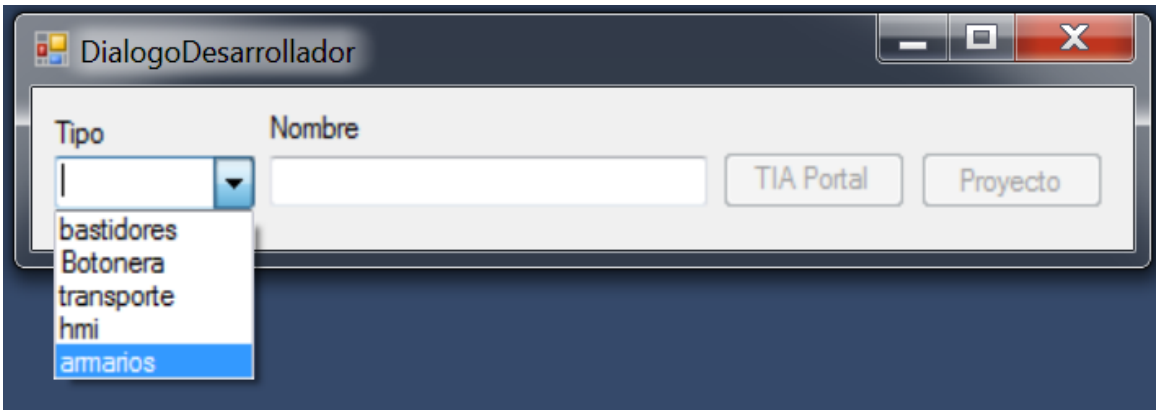


Ilustración 42 Interfaz gráfica de la aplicación de conversión de proyecto TIA Portal a XML y volcado a ExistDB

4.3.8 Aplicación de captura de datos de la máquina.

Se ha desarrollado una aplicación con una interfaz de usuario que permite especificar la información necesaria para la creación de una máquina modular, usando el programa de comunicaciones con la base de datos, creado previamente.

La aplicación accede a la base de datos de la que se extraen los proyectos tipo definidos, permitiendo al usuarios saber los que pueden utilizarse en la composición de la máquina, clasificados por componentes mecatrónicos de primer nivel: Bastidor, Transporte, Botonera y Armario. Esto corresponde al paso 1 de la Ilustración 18 Arquitectura general.

Se muestra a continuación la interfaz gráfica del usuario de esta aplicación.

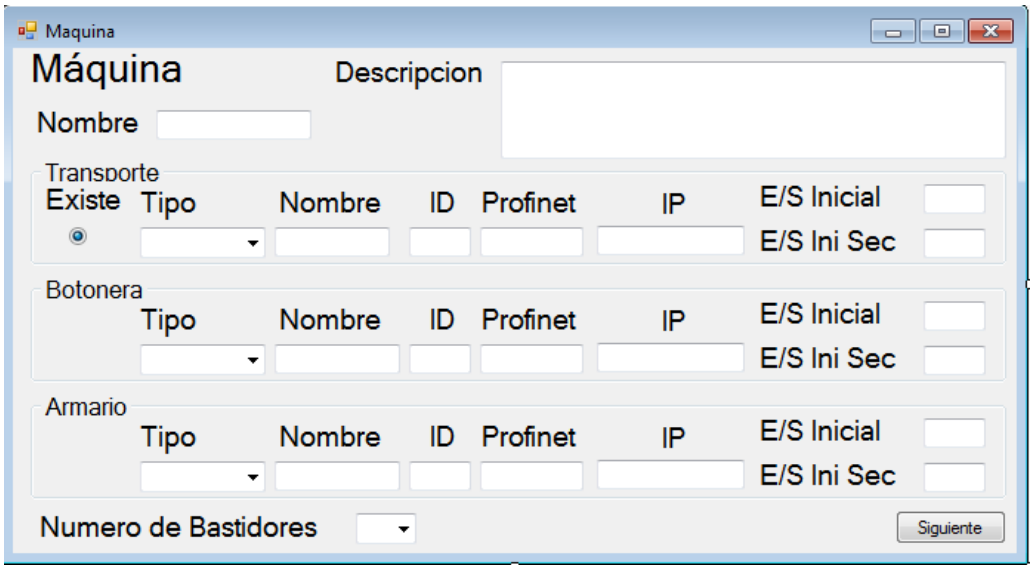


Ilustración 43 Interfaz de definición de la máquina

- Al leer aml y general XML, hace que la aplicación se a independiente de cómo se haya generado el primer XML.

Con el fin de garantizar que la configuración de la máquina sea correcta, el cuadro de dialogo no permite guardar los datos introducidos, en el caso de que no se cumplan que sean diferentes las:

- IDs
- IPs de todos los dispositivos de la máquina
- Direcciones de E/S
- Direcciones seguras
- Los nombres Profinet

Con los datos recogidos y siguiendo unos criterios previamente establecidos, se equilibra automáticamente el número de entradas y salidas igualándose al mayor. Así mismo, el direccionamiento de E/S y de las direcciones IP dentro de los dispositivos de un módulo se realiza automáticamente de forma secuencial.

Toda la información recogida se guarda como una instancia en una de las tres clases de C#, previamente desarrollada en el entregable de modelado, denominada vista de implementación. La instancia que contiene toda la información necesaria para que el generador de código pueda crear una máquina modular se guarda en formato XML dentro de la base de datos.

Se usa la aplicación descrita en... para subir los archivos a eXist.

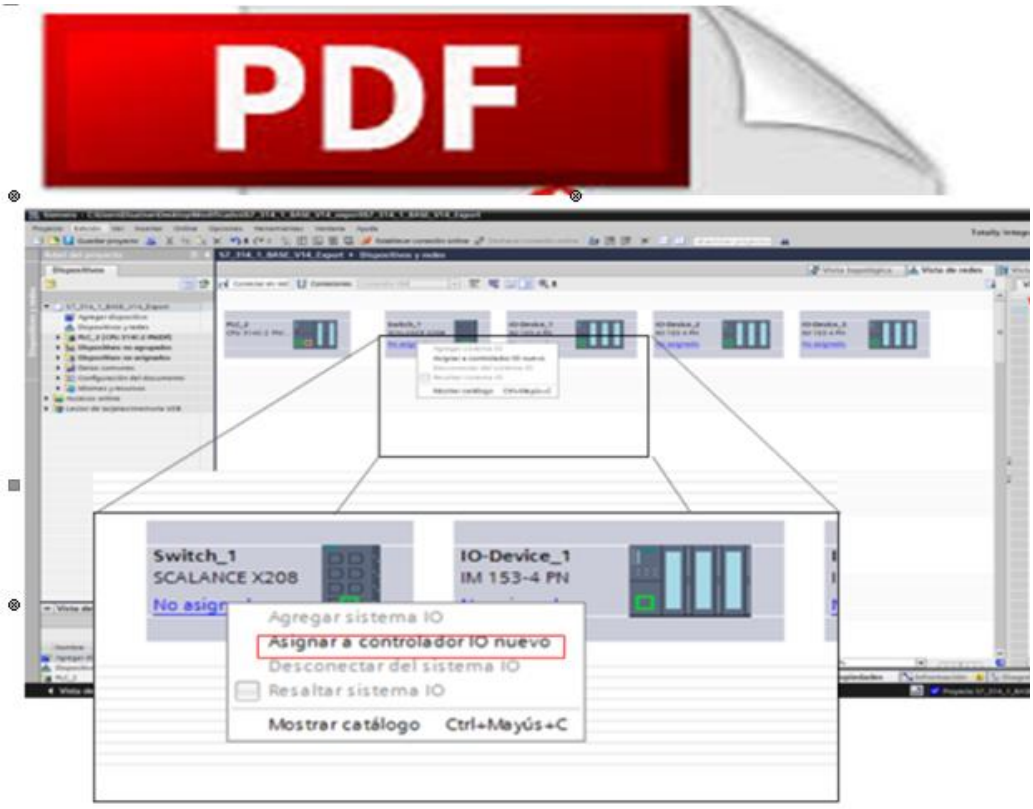
(En una segunda versión se lee del mal para saber cuántos dispositivos hay etc.,)

4.3.9 Aplicación de generación automática de manuales de configuración (hardware y comunicaciones)

Esta aplicación es la utilizada por el generador de código en la fase 5 del paso 2 de la arquitectura general de la solución adoptada, que sirve para generar un manual con las instrucciones necesarias para que el operario introduzca la información que no se puede generar de forma automática.

Para ello la aplicación lee la información contenida en el archivo XLM de configuración de la máquina modular y el generador de manuales crea un manual especifico, que incluye los pasos a seguir para configuración del hardware y de las comunicaciones de la máquina selecionada.

Se muestra a continuación una página del documento creado en esta fase.



- Botón derecho en el esclavo.
- pinchar en Asignar controlador IO nuevo
- Seleccionar el maestro (Único elemento de la lista)

Ilustración 44 Ejemplo de manual de configuración hardware para asignar controlador IO a esclavo de periferia distribuida en Profinet

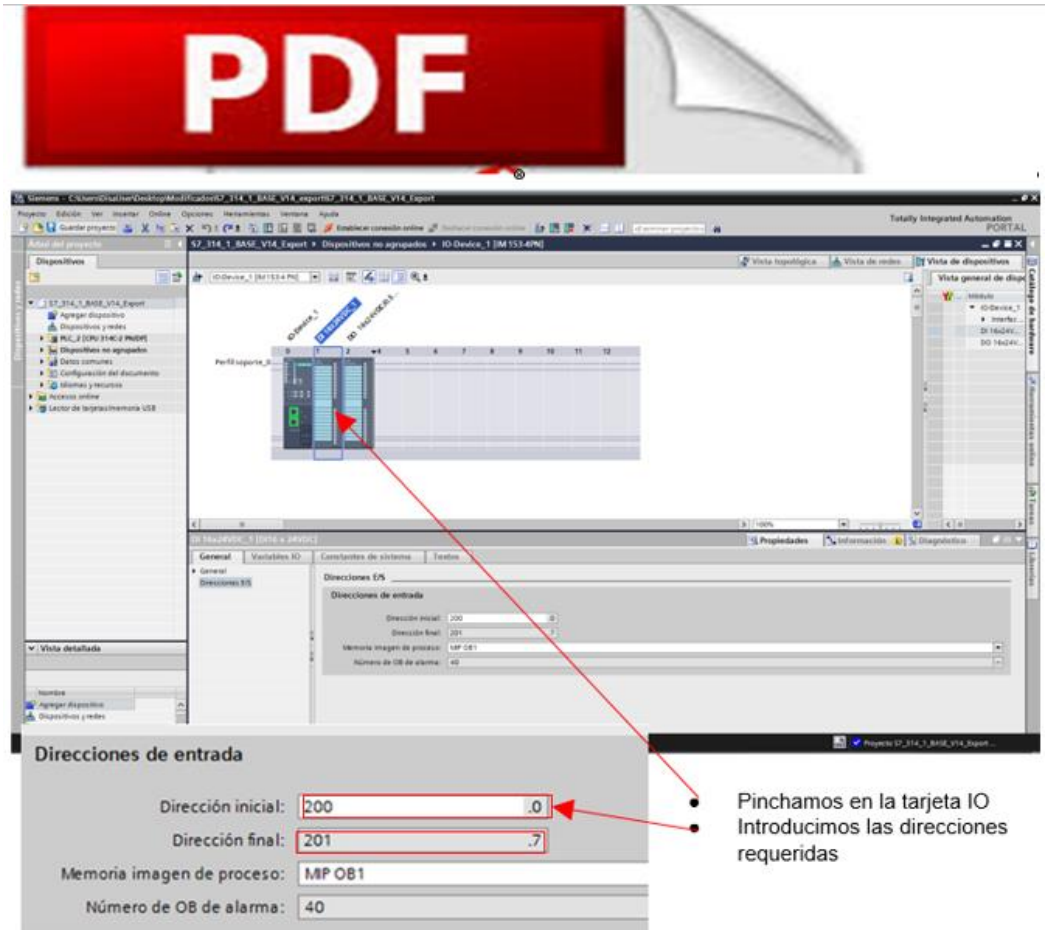


Ilustración 45 Ejemplo de manual de configuración hardware para asignar direcciones de E/S al PLC

5 CASO DE ESTUDIO

El caso de estudio se ha desarrollado para la generación del proyecto de automatización de una máquina real de testeo de fugas, utilizada en la fabricación de bloques y culatas de motores de combustión en el sector del automóvil, diseñada de forma no modular.

Aunque este tipo de máquinas pueden tener diferente formas constructivas, en muchos casos la máquina está formada por un plato giratorio, con varias posiciones (3 o 4), en las que se realizan las operaciones sobre las piezas. Estas máquinas pueden estar juntas formando una línea de procesando de piezas, como se muestra en la siguiente ilustración

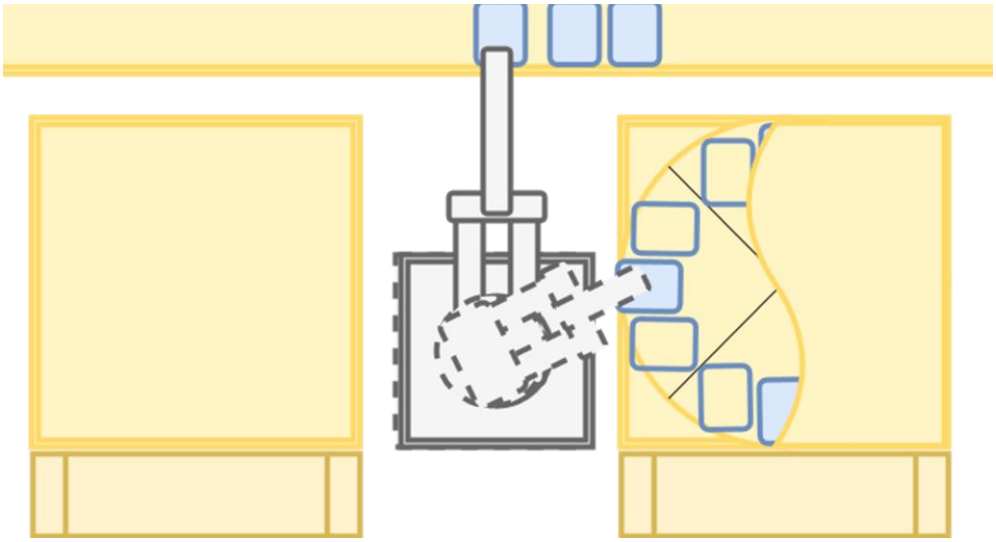


Ilustración 46 Vista en planta de la disposición de máquina de fugas

La máquina consta de los elementos componentes mecatrónicos siguientes.

- Plato divisor (Bastidor): Es un plato giratorio de 3 posiciones desfasadas 120º, accionado por un servo-accionamiento eléctrico en posición, que dispone de señales E/S analógicas y digitales y tiene un FB asociado de control. En una posición se realiza la carga y descarga de las piezas, en la segunda posición se realiza el test del circuito de agua y en la tercera posición se realiza el test del circuito de aceite.
- Cilindro hidráulico (Operación): El cilindro hidráulico eleva la pieza hasta la posición donde se encuentra un bloque mecánico, sobre el que se encuentran los sensores y actuadores del “sistema de inyección de aire y registro de valores”. Tiene un dispositivo IO con un FB para su control, por cada pieza y por cada estación de la máquina.
- Sistema de inyección de aire y registro (Operación): Este sistema consta de sensores de medida y actuadores y dispone de un sistema de control propio que

realiza el control del testeo sobre la pieza. Dispone de un FB por cada tipo de pieza y por estación. La comunicación con el PLC es en Profinet

- Transporte: la maquina utilizada como caso de estudio no tiene transporte.
- Sistema de alimentación: La alimentación de la máquina se hace de forma manual. Las señales están conectadas a un dispositivo I/O y se procesan mediante un FB.
- Interfaz de operador (HMI): La máquina dispone de un interfaz de operador HMI desde donde se puede dar órdenes al sistema de control.
- Botonera (Botonera): La máquina dispone de un conjunto de pulsadores y dispositivos de puesta en marcha y paros de la máquina. Las señales implicadas se controlan mediante un FB de control de señalizaciones y se capturan y transmiten a través de un dispositivo I/O.

5.1 Funcionamiento de la máquina

La máquina de test de fugas forma parte de una línea de montaje donde se encarga de realizar el test a una familia de piezas. La máquina se alimenta mediante un robot de ABB con su propio PLC mediante un acoplador PN-PN, que a su vez sirve para realizar la trazabilidad de la máquina con PC de trazabilidad de la línea.

5.2 Pasos de ejecución

Se muestran a continuación los pasos de ejecución de la máquina:

1. Carga de piezas a la estación 1 y giro del plato divisor con las 4 estaciones 90º
2. Elevación mediante pistones hidráulicos de las piezas contenidas en las estaciones 2, 3 y 4, presionándolas contra un útil diseñado para garantizar la estanqueidad de la superficie. Este útil cuenta con una serie de cilindros accionados por un sistema neumático que sellan los orificios y cavidades internas del motor para su posterior prueba
3. Diferentes testeos de estanqueidad y continuidad de orificios, según la estación y pieza de estudio

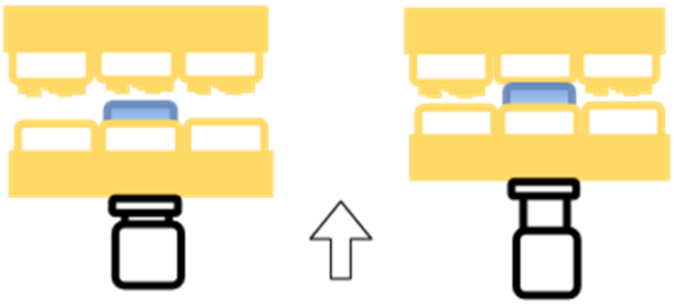


Ilustración 47 Movimiento de sello mecánico

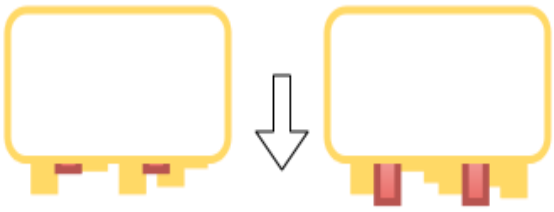


Ilustración 48 Detalle sello de mecánico mediante cilindros de los orificios de una culata para su test de fugas

5.3 Modelado de la máquina:

Al tratarse de una adaptación de una máquina no modular a un diseño modulra, el primer paso consiste en identificar y relacionar los componentes de la máquina real con el modelo definido para este tipo de máquinas. En el modelo, la máquina se divide en las partes funcionales que aparecen en la ilustración:

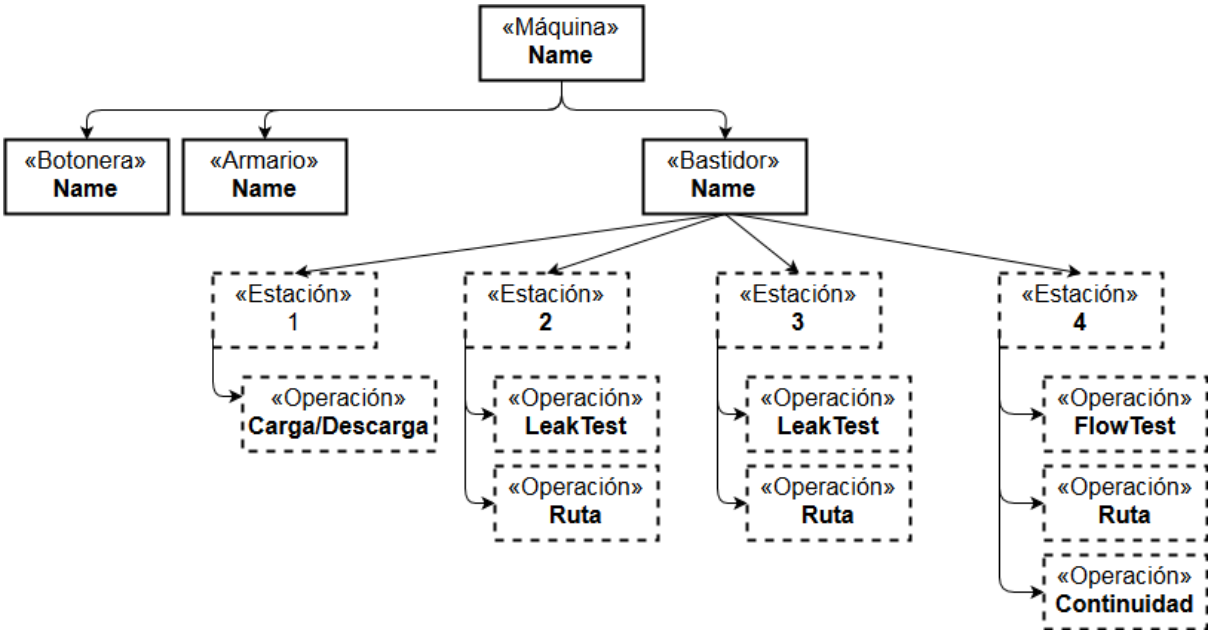


Ilustración 49 Modelos de máquina de test de fugas

Análisis del hardware de la Máquina:

La máquina inicial, a nivel de hardware, está está formada por los dispositivos que se muestran a continuación:

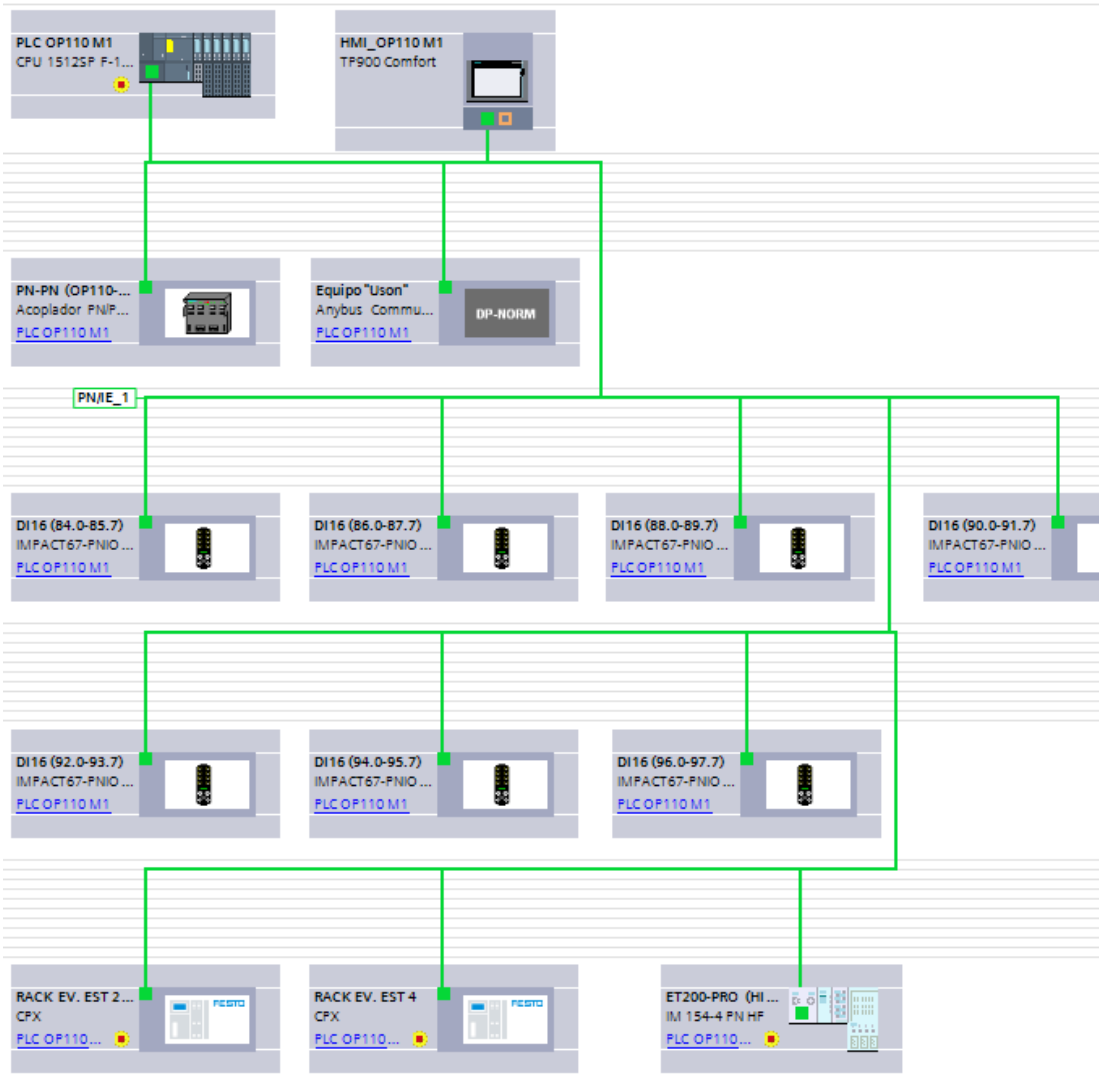


Ilustración 50 Dispositivos y redes maquina de fugas

Se ha realizado un análisis de la relación entre los dispositivos del sistema de control (PLC, dispositivos IO, etc.) y los módulos o elementos del modelo (botonera, armario, etc.). El resultado del análisis se muestra en la siguiente ilustración.

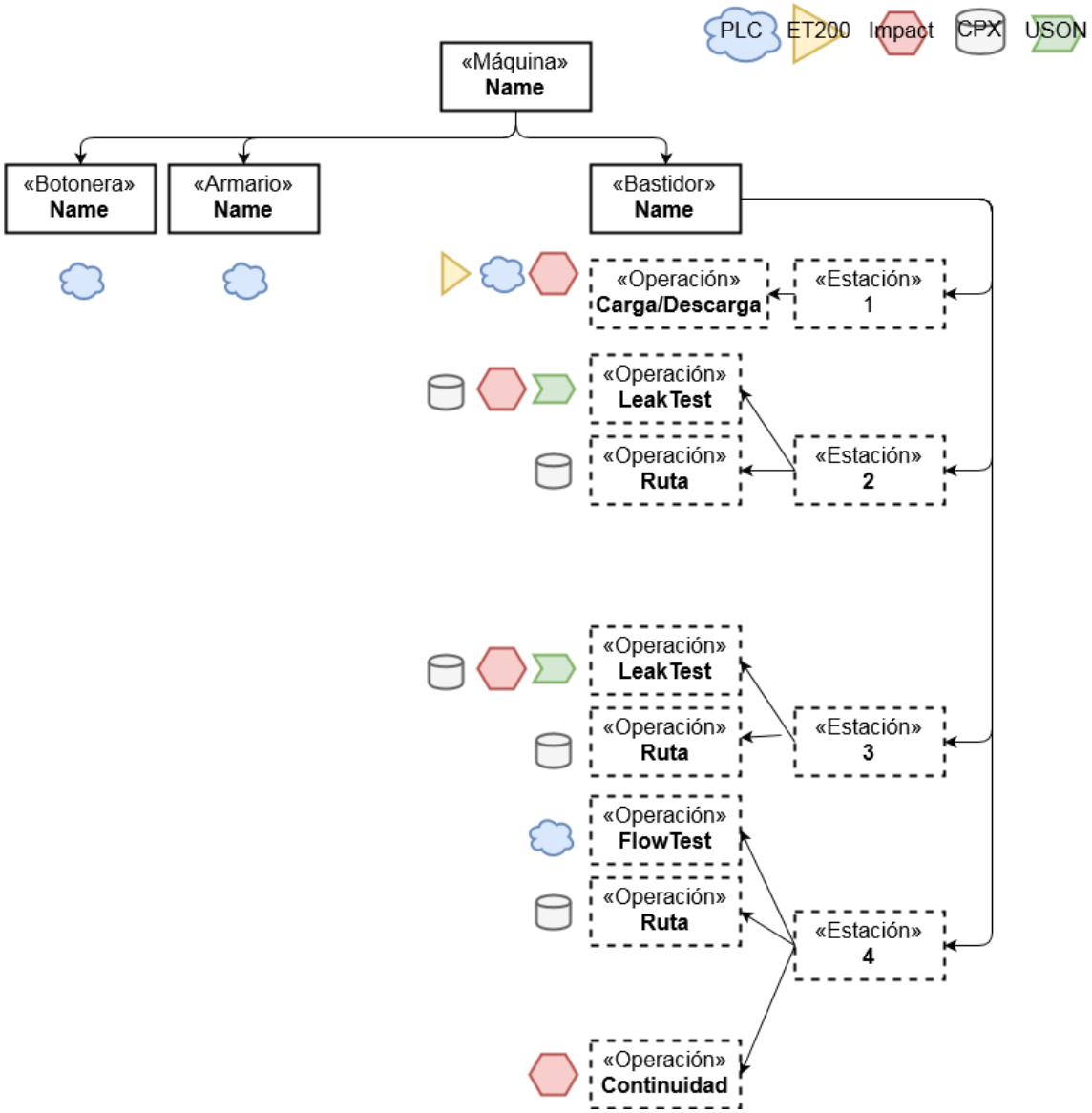


Ilustración 52 Relación entre dispositivos del sistema de control y los elementos del modelo

A continuación se ha realizado un análisis de dónde están señales de E/S relacionadas con los componentes del modelo de la máquina. Así por ejemplo, se observa que todas las señales del elemento “Botonera” están en tarjetas de E/S del PLC.

Posteriormente se hace un análisis de las relaciones de dependencia que existen entre los dispositivos hardware del proyecto y las señales de E/S de los componentes del modelo de la máquina. Para ello, se han analizado las direcciones parametrizadas en los dispositivos IO, y su relación con la E/S de los componentes del modelo y se ha identificado cada dispositivo hardware con el último campo de la dirección IP del mismo, encuadrados dentro de los iconos que representan los diferentes dispositivos de la máquina.

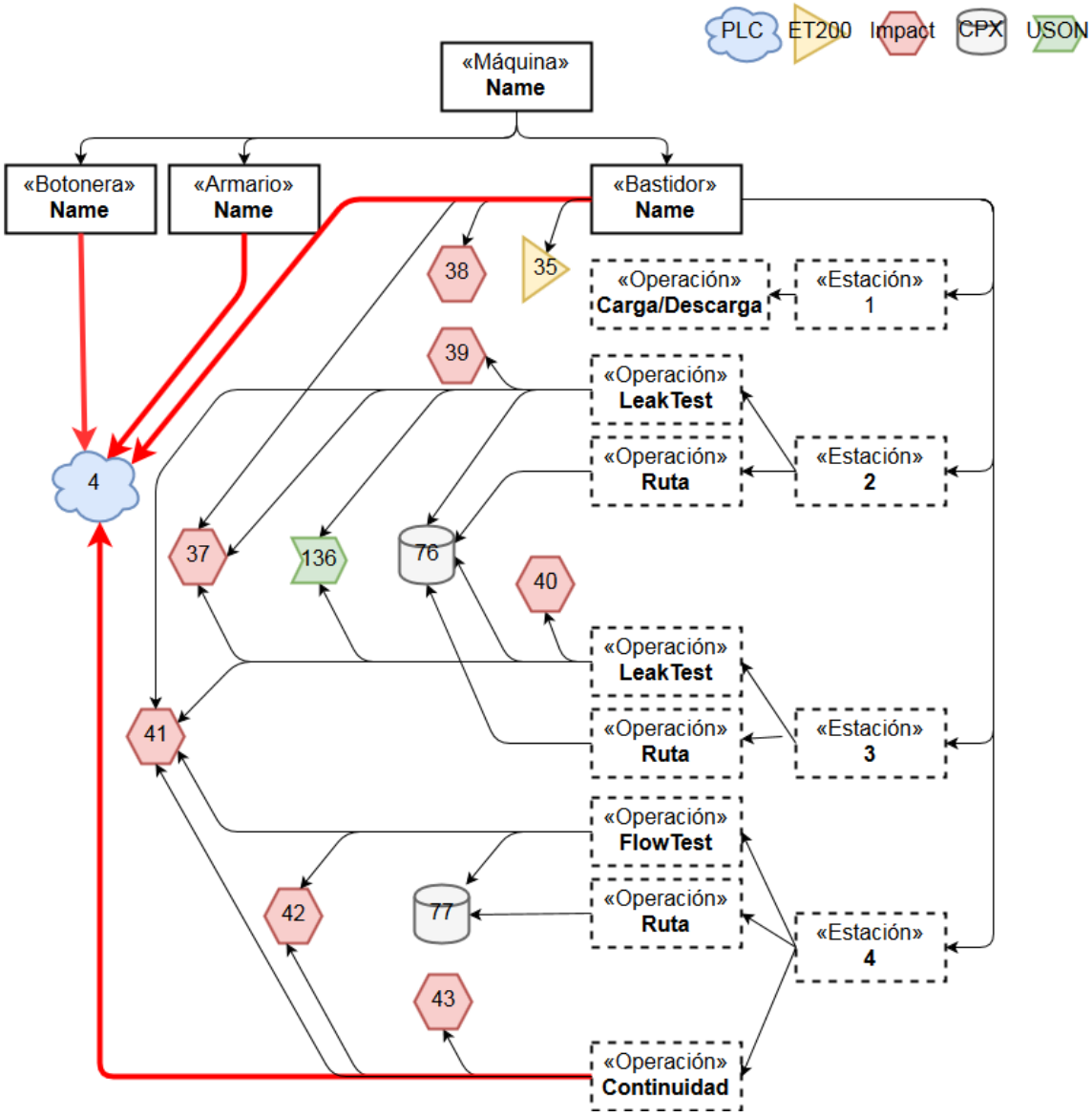


Ilustración 53 Relación de E/S de los módulos tipo con los dispositivos IO

Con el objetivo de conseguir una distribución del hardware más modular, se propone que cada módulo funcional tenga su propio hardware independiente del resto para que el software se pueda desarrollar asociado a dicho módulo. Para ello se proponen los los siguientes cambios:

- Añadir un dispositivo I/O de E/S para “Botonera” y otro dispositivo I/O para el “Armario”. De esta forma se gana en modularidad independizando las E/S de ambos componentes del las E/S del PLC haciendo que los proyectos TIA Portal de ambos componentes sean independientes.
- Cambiar las direcciones de las variables, para reducir las relaciones de dependencia (representadas como flechas hacia un mismo dispositivo en la ilustración), siguiendo los pasos que se citan a continuación.

- Paso de las dos señales del “Cierre inferior de la estación 2” del dispositivo IO con IP finalizada en 37 al dispositivo IO con IP finalizada en 39.
- Paso de las dos señales del “Cierre inferior de la estación 3” del dispositivo IO con IP finalizada en 37 al dispositivo IO con IP finalizada en 40.
- Paso de las señales del “Cierre frontal de la estación 3” del dispositivo IO con IP finalizada en 41 al dispositivo IO con IP finalizada en 40.
- Paso de las señales del “Conjunto cierre de la estación 2” del dispositivo IO con IP finalizada en 41 al dispositivo IO con IP finalizada en 39.
- Paso de todas las señales de “Test continuidad eléctrica” del dispositivo IO con IP finalizada en 41 a los dispositivos IO con IP finalizadas en 42 y 43.

En el resultado final solo existe relación de dependencia en aquellos equipos que no se pueden sustituir por dispositivos E/S, como son la máquina que realiza el test (USON) y el PLC, como puede verse en la siguiente ilustración.

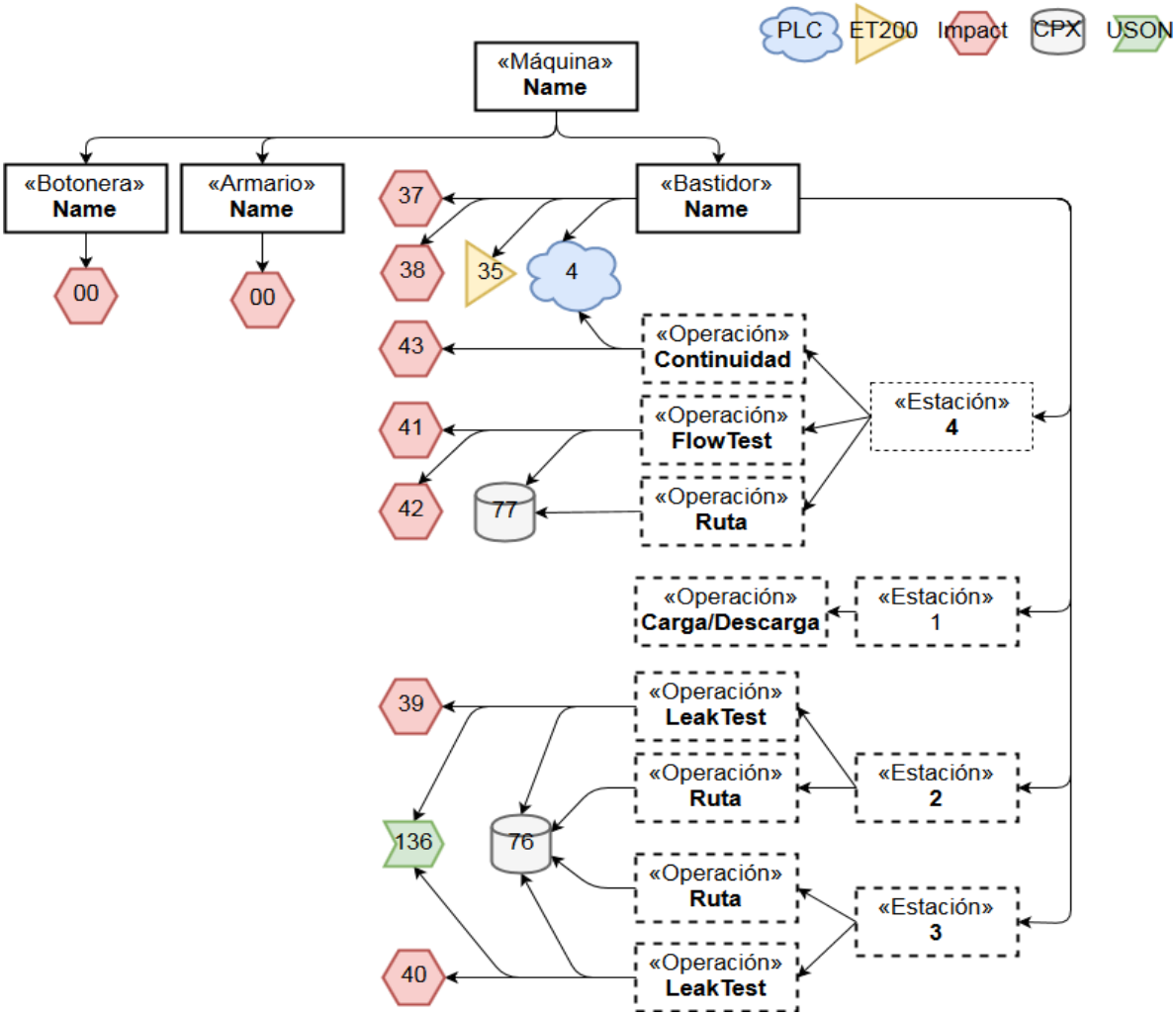


Ilustración 54 . Propuesta de distribución de las señales de E/S

Cabe reseñar que el proyecto de automatización en TIA Portal está realizado de forma no modular al no disponerse del conocimiento necesario para diseñar una máquina de fugas (labor de la empresa) , por lo tanto se han creado una batería de proyectos tipo simulando los bloques de programa de la máquina modular, sin la lógica interna (Know-how no visible al generador de cdigo “caja negra”) pero que contiene las varaibles y parámetros de dichas funciones.

Resultado obtenido tras la aplicación del generador de código es la siguiente,

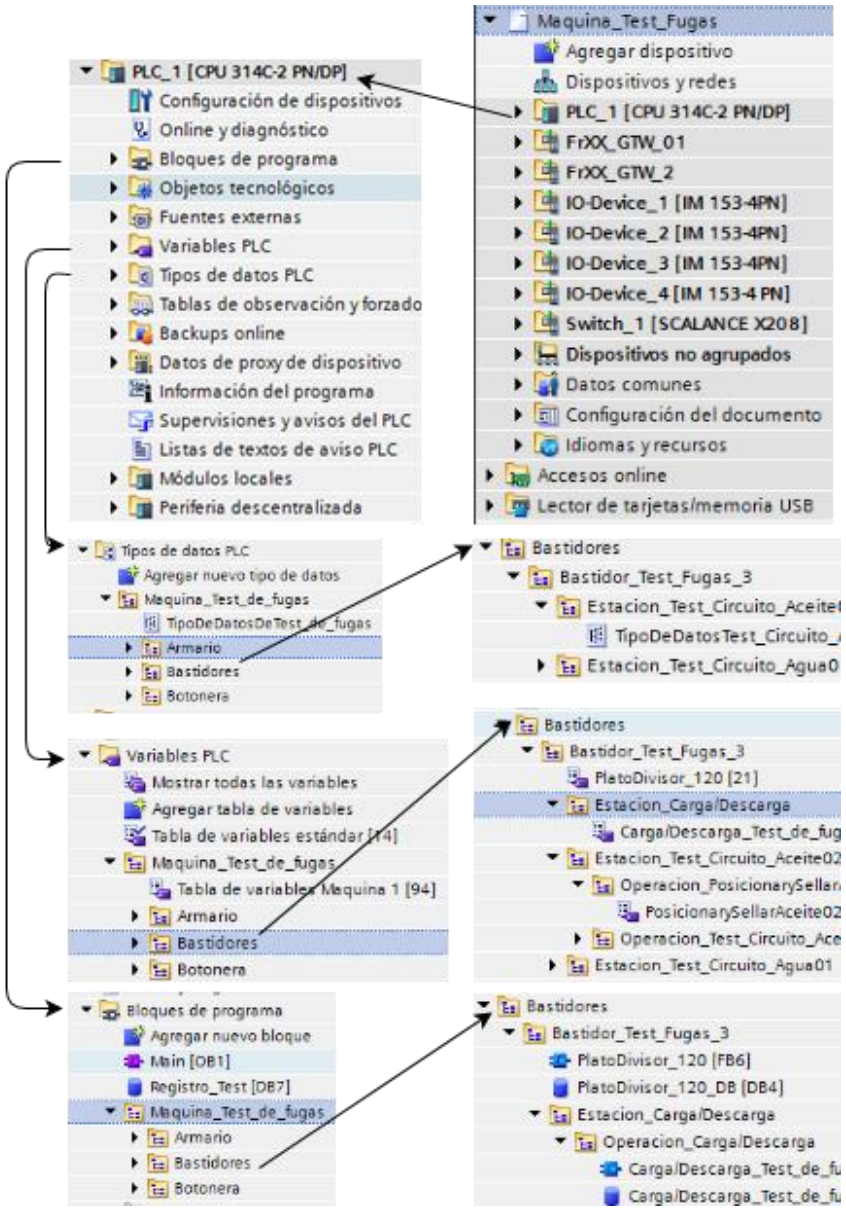


Ilustración 55 Resultado generación de código en TIA Portal

6 ASPECTOS ECONÓMICOS

En este capítulo se incluye un descargo de los gastos realizados durante el desarrollo del trabajo y se justifica la no aplicabilidad de la realización del análisis de la rentabilidad.

6.1 Presupuesto

Al tratarse de un proyecto de I+D+i no conlleva una ejecución inmediata del mismo por lo que la partida más alta corresponde a las horas de dedicación internas.

Tabla 2 Resumen de coste de horas de dedicación internas

Horas Internas imputadas			
Trabajador	**Coste Horario**	**Horas**	**Coste**
Ingeniero Junior	20	870	17400
Ingeniero Senior	50	650	32500
Subtotal Horas Internas			**49.900 €**

Tabla 3 Resumen de gastos del trabajo

Gastos	
Material Oficina	100 €
Desplazamientos	100 €
Subtotal Gastos	**200 €**

Tabla 4 Resumen de gastos amortizaciones de infraestructura

Amortizaciones				
Inversión/Activo Fijo	**Coste de adquisición**	**Vida útil**	**Tiempo de utilización en el trabajo**	**Amortización**
Ordenador	1.500 €	4 años	700 horas	165,53 €
Licencias TIA Portal	2.800 €	1 años	250 horas	473,50 €
Licencia Visual Studio	574 €	1 años	500 horas	156,25 €
Licencia XML Spy	391 €	1 años	100 horas	24,43 €
Licencia Microsoft Office 360	190 €	1 años	300 horas	35,62 €
PLC s7 1500	3.327 €	10 años	100 horas	20,80 €
Esclavo Profinet ET200	236 €	10 años	60 horas	0,90 €
Módulo neumático CPX	430 €	10 años	60 horas	1,61 €
Subtotal Amortizaciones				**878,64 €**

Tabla 5 Resumen de descargo de gastos del trabajo

Resumen	
Horas internas	49.900 €
Amortizaciones	878,64 €
Gastos	200 €
Subtotal	50.979 €
Imprevistos (7%)	3.569 €
TOTAL	**54.547 €**

6.2 Análisis de rentabilidad

Al tratarse de un proyecto de I+D+i, que en la actualidad se encuentra en una fase temprana de estudio no es posible realizar el análisis de rentabilidad del producto final que incorpora la nueva tecnología.

7 DESCRIPCIÓN DE TAREAS. DIAGRAMA GANTT

Tabla 6 Tarea1 Recopilación de información

Tarea: 1	Recopilación de información
Objetivo	Recoger datasheet, manuales, sis del proyecto a realizar
Duración	10 días
Subtareas	1. Revisión documentación previa del Departamento 2. Descarga de manuales y datasheet 3. Revisar publicaciones científicas

Tabla 7 Tarea 2 Análisis del proyecto a realizar.

Tarea: 2	Análisis del proyecto a realizar.
Objetivo	Crear un plan de acción.
Duración	20 días
Subtareas	1. Problemática 2. Herramientas necesarias

Tabla 8 Tarea 3 Desarrollo de un framework de trabajo

Tarea: 3	Desarrollo de un framework de trabajo
Objetivo	Crear un entorno de trabajo, que facilite el desarrollo de la aplicación a desarrollar
Duración	7 días
Subtareas	1. Crear máquina virtual 2. TIA Portal, openness etc. 3. Redes de comunicaciones 4. Notepad ++ 5. Visual Studio 6. Exist DB

Tabla 9 Tarea 4 Modelado

Tarea: 4	Modelado
Objetivo	Crear un modelo para la utilización por parte del generador de código automático
Duración	35 días
Subtareas	1. Análisis del entorno de aplicación de la máquina modular y definición de las características necesarias 2. Creación de un Metamodelo que cumpla las características marcadas 3. Aplicación del Metamodelo a una maquina concreta

Tabla 10 Tarea 5 Definición de los módulos

Tarea: 5	definición de los módulos
Objetivo	Definir una metodología para conseguir los módulos y Permitir el correcto funcionamiento del generador de código
Duración	20 días
Subtareas	1. Identificar la problemática de las herramientas de siemens 2. Análisis de alternativas para alcanzar las características marcadas 3. Elaboración de la solución marcada 4. Crear un conjunto de proyectos tipos

Tabla 11 Tarea 6 Desarrollo de la aplicación de interacción con TIA Portal

Tarea: 6	Desarrollo de la aplicación de interacción con TIA Portal
Objetivo	Crear una aplicación personalizada a las necesidades de interactuación del proyecto con la herramienta TIA Portal
Duración	17 días
Subtareas	1. Análisis de la herramienta TIA Portal Openness 2. Estudio de los demostradores de Siemens 3. Acceso a Tia Portal y proyectos 4. Exportación e importación del software de los proyectos TIA Portal 5. Manipulación de las librerías de TIA Portal, para su uso en el apartado hardware 6. Navegación dentro de las carpetas de un proyecto de automatización

Tabla 12 Tarea 7 Estructura de la aplicación final

Tarea: 7	Estructura de la aplicación final
Objetivo	Definir el ecosistema necesario para la generación automática de TIA Portal
Duración	9 días
Subtareas	1. Definición de las etapas 2. Definición de la relación entre las etapas

Tabla 13 Tarea 8 Librerías hardware

Tarea: 8	Librerías hardware
Objetivo	Manipular la información de configuración de Hardware.
Duración	1 días
Subtareas	1. Manual de hardware 2. Archivo configuración hardware

Tabla 14 Tarea 9 Aplicación de manipulación de los módulos en formato de archivos XML

Tarea: 9	Aplicación de manipulación de los módulos en formato de archivos XML
Objetivo	Conseguir una herramienta para la manipulación de los proyectos tipo
Duración	15 días
Subtareas	1. Cambio tipos de datos 2. Cambios en las variables 3. Cambio de bloques de programa en STL 4. Ampliar el cambio de bloques de programa a todos los lenguajes de programación 5. Generador del OB del proyecto con los Obs de los módulos. 6. Entorno grafico para realizar el cambio de XML de forma automática de todos los cambios mencionados anteriormente.

Tabla 15 Tarea 10 Base de datos eXist modo manual etc. (subir base de datos)

Tarea: 10	base de datos eXist modo manual etc.(subir base de datos)
Objetivo	Conseguir un sistema centralizado de almacenamiento que permita trazabilidad y seguridad de acceso
Duración	20 días
Subtareas	1. Instalación y configuración de la base de daros 2. Estructura de la base de datos para los XML 3. API Sistema de comunicación mediante c# 4. Aplicación con interfaz gráfica para el desarrollador pueda subir los módulos, directamente de los proyectos tipo de TIA Portal

Tabla 16 Tarea 11 Aplicación de captura de datos y configuración de máquinas

Tarea:11	Aplicación de captura de datos y configuración de máquinas
Objetivo	Aplicación con interfaz gráfica para que el usuario pueda crear máquinas a partir de la composición de los módulos disponibles en la base de datos.
Duración	5 días

Tabla 17 Tarea12 Caso de estudio

Tarea:12	Caso de estudio
Objetivo	Validar cada una de las aplicaciones con un caso simulado, pasando de máquina a módulos y de nuevo a máquina.
Duración	25 días
Subtareas	• Análisis de un proyecto no modular para su modularización • Creación de los proyectos tipo mediante la adaptación del proyecto no modular. • Crear los módulos HW de cada proyecto tipo • Subir a la base de datos los proyectos tipos en XML • Definir y parametrizar la máquina • Manipular los XML

7.1 Diagrama Gantt

	Nombre	Duración
1	⊞Recopilación de información	5d
5	⊞Análisis del proyecto	20d?
8	⊞desarrollo framework de trabajo	7d?
15	⊞modelado	35d?
19	⊞definición de los módulos	20.5d?
24	⊞aplicación de interacción con TIA Portal	17d?
29	⊞Estructura de la aplicación final	9d?
32	⊞librarias hardware	1d?
35	⊞módulos en formato XML	15d?
42	⊞base de datos eXist	19.5d?
47	Aplicación de Adquisición de datos	4.5d?
48	⊞Caso de estudio	25.5d?

Ilustración 56 Diagrama Gantt con las tareas del trabajo

	Nombre	Duración
1	**Recopilación de información**	**5d**
2	revisión documentación previa lavoratorio	5d
3	recopilación datasheet, manuales,	5d
4	publicaciones cientificas	5d
5	**Análisis del proyecto**	**20d?**
6	problemática	20d?
7	herramientas necesarias	5d?
8	**desarrollo framework de trabajo**	**7d?**
9	maquina virtual	2d?
10	Tia portal, openness	5d?
11	Redes de comunicaciones	2.5d?
12	Notepad	3d?
13	Visual Studio	5d
14	Exist DB	5d?
15	**modelado**	**35d?**
16	Análisis del entorno de aplicación de la maquina modular y defir	15d?
17	Creación de un Metamodelo que cumpla las características mar	30d?
18	Aplicación del Metamodelo a una maquina concreta	5d?
19	**definición de los módulos**	**20.5d?**
20	Identificar la problemática de las herramientas de Siemens	10d?
21	Análisis de alternativas para alcanzar las características marcad	5d
22	Elaboración de la solución marcada	20d?
23	Crear un conjunto de proyectos tipos	0.5d?
24	**aplicación de interacción con TIA Portal**	**17d?**
25	Análisis de la herramienta TIA Portal Openness	5d?
26	Estudio de los demostradores de siemens	5d?
27	Acceso a Tia portal y proyectos	5d?
28	Exportación e importación del software	2d?
29	**Estructura de la aplicación final**	**9d?**
30	Definición de las etapas	9d?
31	Definición de la relación entre las etapas	9d?
32	**librarias hardware**	**1d?**
33	Manual de hw	1d?
34	Archivo conf hw	1d?
35	**módulos en formato XML**	**15d?**
36	Cambio tipos de datos	10d?
37	Cambios en las variables	5d?
38	Cambio de bloques de programa en STL	5d?
39	bloques de programa a todos los lenguajes	5d
40	Generador del OB del proyecto	5d?
41	Entorno grafico cambio de XML de	10d?
42	**base de datos eXist**	**19.5d?**
43	Instalación y configuración de la base de daros	4.5d?
44	Estructura de la base de datos para los XML	6d?
45	Api Sistema de comunicación	15d?
46	Aplicación con interfaz gráfica	5.5d?
47	Aplicación de Adquisición de datos	4.5d?
48	**Caso de estudio**	**25.5d?**
49	Análisis de un proyecto no modular	13.5d?
50	Creación de los proyectos tipo	4.5d?
51	Crear los módulos HW	1.5d?
52	Subir a la base de datos los proyectos tipos en XML	5d?
53	Definir Y parametrizar la maquina	2.5d?
54	Manipular los XML	3d?

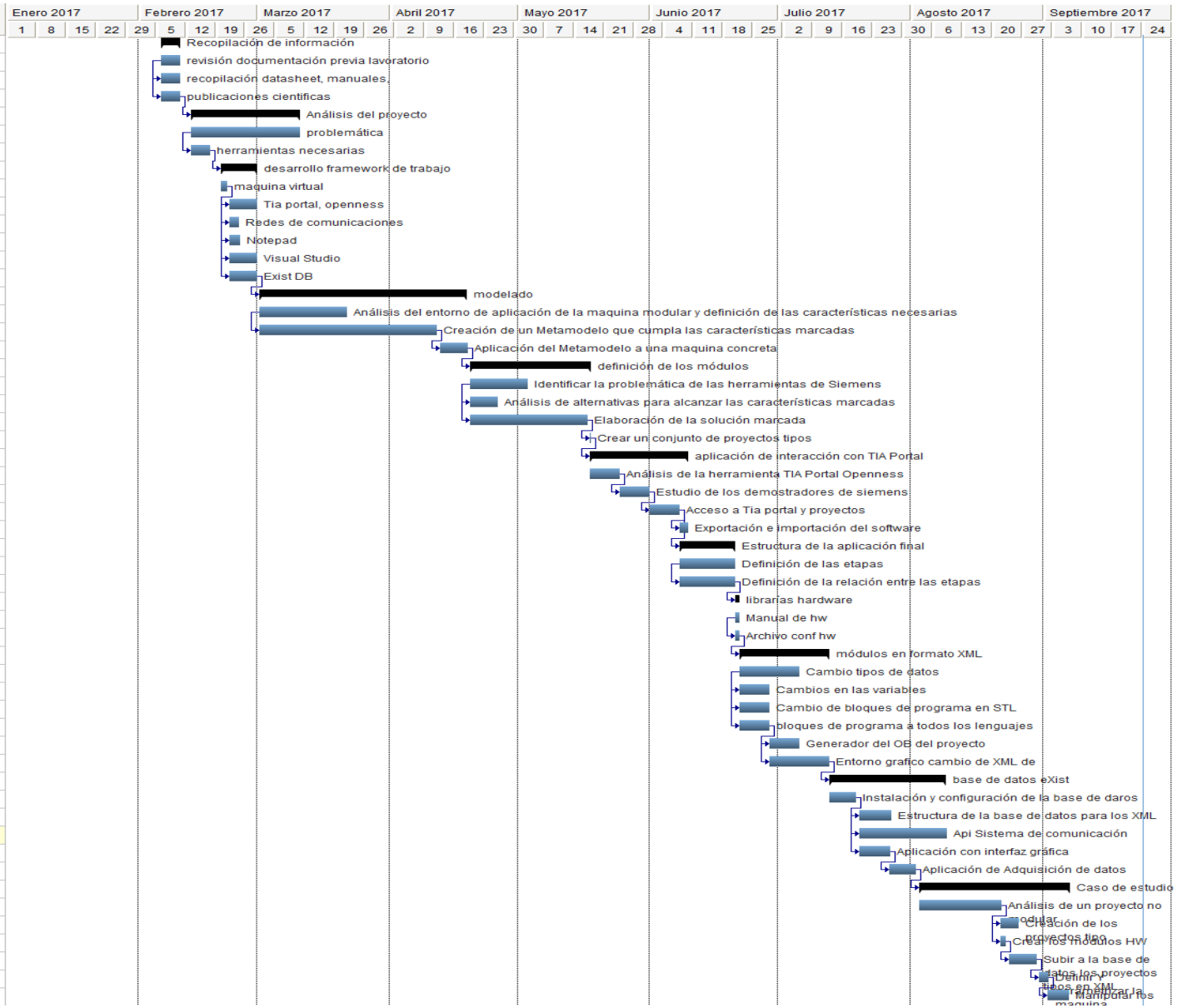


Ilustración 57 Diagrama Gantt con tareas y Subtareas del trabajo

8 CONCLUSIONES Y TRABAJOS FUTUROS

En este apartado se exponen las conclusiones más relevantes extraídas del presente trabajo, presentándose también los trabajos futuros que le darán continuidad en el Departamento de Ingeniería de Sistemas y Automática.

8.1 Conclusiones

Se indican a continuación las principales conclusiones obtenidas tras la finalización del trabajo:

1. La aplicación desarrollada siguiendo la arquitectura propuesta, permite generar de forma automática proyectos de automatización para máquinas modulares a través de módulos desarrollados como proyectos tipo.
2. La nueva metodología que se ha desarrollado elimina los inconvenientes que surgen durante el desarrollo de máquinas de forma manual, como es el caso del cambio de direccionamiento, lográndose un alto grado de reutilización del trabajo realizado previamente.
3. La característica de ser un entorno abierto, con una utilidad incremental, permite que cada vez que se desarrolla una nueva máquina se pueda reutilizar la información sobre los módulos existentes dentro de la base de datos y ampliarla incorporando la correspondiente a los nuevos módulos diseñados.
4. El uso de tecnologías de modelado (Metamodelo) y métodos de desarrollo de programación, como es el caso de los diagramas de clases UML, permite añadir una capa de abstracción y el uso de técnicas probadas y altamente utilizadas en tecnologías de la información (TIC), facilitando de este modo la labor de diseño.
5. El empleo del estándar de lenguajes de programación IEC 61131-3 permite que la metodología de programación de los proyectos tipo, así como los objetivos marcados donde actúa el generador de código sean comunes en todas las plataformas.
6. El patrón de diseño modular exige que todos los módulos que componen la máquina hayan sido previamente diseñados e implementados en proyectos tipo TIA Portal.
7. La apuesta de Siemens por abrir su plataforma TIA Portal mediante la API TIA Portal Openness, supone el lanzamiento de nuevas versiones en periodos muy cortos en el tiempo, con cambios severos de las funciones, que pueden provocar efectos no deseados por obsolescencia y la no retrocompatibilidad con proyectos en versiones anteriores así como las posibles incidencias no conocidas al tratarse de un software nuevo.

8. El diseño de los módulos de las máquinas debe realizarse de forma coordinada con el resto de áreas de ingeniería, como por ejemplo la mecánica o la hidráulica, para que la modularización sea efectiva.

8.2 Trabajos futuros

El trabajo ha supuesto un primer paso para la modularización de los elementos lógicos de los proyectos de ingeniería, al haberse desarrollado un Metamodelo y la arquitectura general del generador de código de proyectos de automatización, enmarcado en el nivel máquina. Al tratarse de una arquitectura abierta es posible la ampliación a otros alcances como el inmediatamente superior, correspondiente al nivel de línea de fabricación o el inferior, como podría ser el caso del desarrollo de bastidores.

Cabe reseñar que, dentro del Departamento de Ingeniería de Sistemas y Automática, se ha decidido dar continuidad al proyecto de modularización, ampliando el alcance del trabajo al nivel de línea de fabricación. Para ello se utilizará la nueva versión TIA Portal Openness V14 SP1, de reciente lanzamiento por parte de Siemens, que incluye nuevas utilidades que se emplearan en los futuros trabajos.

Entre las principales novedades que incluye se encuentra la introducción de AML como soporte de exportación, tanto de la configuración del hardware como de la configuración de las comunicaciones, lo que facilita la exportación de la totalidad de los proyectos fuera de la herramienta TIA Portal en dos soportes diferentes pero complementarios. Esto permite desechar la utilización de las librerías de TIA Portal, otorgando mayor independencia del software propietario de Siemens.

Futuras líneas de trabajo podrían incluir el desarrollo de proyectos multiplataforma, traduciendo los modelos para adecuarlos a la tecnología demandada. Un ejemplo podría ser el desarrollo de los proyectos tipo en tecnologías Siemens, su exportación a formato XML y la traducción a un XML universal, que podría ser utilizado en un entorno distinto a Siemens, permitiendo la reutilización multiplataforma.

9 BIBLIOGRAFÍA

[1] B. Prasad, “Analysis of pricing strategies for new product introduction,” *Pricing Strateg. Pract.*, vol. 5, no. 4, pp. 132–141, 1997.

[2] Plattform Industrie 4.0, “What is Industrie 4.0?,” pp. 4–5, 2017.

[3] K. Ulrich, “Fundamentals of product modularity,” in *Management of Design*, Springer, 1994, pp. 219–231.

[4] C. Maga, N. Jazdi, and P. Göhner, “Reusable models in industrial automation: experiences in defining appropriate levels of granularity,” *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 9145–9150, 2011.

[5] R. Priego, A. Armentia, E. Estévez, D. Orive, N. Iriondo, and M. Marcos, “Implementación de mde para la generación de sistemas de automatización reconfigurables,” *XXXVI Jornadas de Automática*, pp. 166–173, 2015.

[6] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, “Evolution of software in automated production systems: Challenges and research directions,” *J. Syst. Softw.*, vol. 110, pp. 54–84, 2015.

[7] S. Feldmann and B. Vogel-Heuser, “Interdisciplinary product lines to support the engineering in the machine manufacturing domain,” *Int. J. Prod. Res.*, vol. 55, no. 13, pp. 3701–3714, Jul. 2017.

[8] E. Estévez, M. Marcos, and D. Orive, “Automatic generation of PLC automation projects from component-based models,” *Int. J. Adv. Manuf. Technol.*, vol. 35, no. 5–6, pp. 527–540, 2007.

[9] E. Siegel and A. Retter, *eXist A NoSQL Document Database and Application Platform*. O’Reilly Media, 2014.

1 Schemas de TIA Portal Openness

1. “Generación automática del proyecto de automatización TIA Portal para máquinas modulares”- Darío Orive, Aintzane Armentia, Eneko Fernández, Marga Marcos. XXXVIII Jornadas de Automática. Septiembre 2017

2 Schemas de TIA Portal Openness

1. IEC/UNE-EN 61131-3:2013. Autómatas programables. Parte 3: Lenguajes de programación

3 Schemas de TIA Portal Openness

4 Schemas de TIA Portal Openness

1.1.1 SW.Common.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="AttributeList" type="AttributList_T"/>
	<xs:complexType name="AttributList_T">
		<xs:choice minOccurs="0" maxOccurs="unbounded">
			<xs:element ref="BooleanAttribute"/>
			<xs:element ref="IntegerAttribute"/>
			<xs:element ref="RealAttribute"/>
			<xs:element ref="StringAttribute"/>
		</xs:choice>
	</xs:complexType>
	<xs:element name="BooleanAttribute" type="BooleanAttribute_T">
		<xs:annotation>
			<xs:documentation>A member attribute with a type restriction of boolean.</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:complexType name="BooleanAttribute_T">
		<xs:simpleContent>
			<xs:extension base="xs:boolean">
				<xs:attribute name="Name" type="xs:string" use="required"/>
```

<xs:attribute name="Informative" type="xs:boolean" default="false">
<xs:annotation>
<xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:attribute name="SystemDefined" type="xs:boolean" default="true">
<xs:annotation>
<xs:documentation>An attribute of attribute, denotes if it is defined by a user or the system itself. In V14, if exists it is always true.</xs:documentation>
</xs:annotation>
</xs:attribute>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
<xs:element name="Comment">
<xs:annotation>
<xs:documentation>Not allowed in STL</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:complexContent>
<xs:extension base="Comment_T"/>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:group name="Comment_G">
<xs:annotation>
<xs:documentation>LAD/FBD: Only for Parts</xs:documentation>

```
            </xs:annotation>
            <xs:sequence>
                <xs:choice maxOccurs="unbounded">
                    <xs:element ref="Comment"/>
                    <xs:element ref="LineComment"/>
                </xs:choice>
            </xs:sequence>
        </xs:group>
        <xs:complexType name="Comment_T">
            <xs:sequence minOccurs="0">
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>For NumBLs. NumBLs is the count of the blank spaces before the actual text in the Comment. This is informative.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="MultiLanguageText"/>
                </xs:sequence>
            </xs:sequence>
            <xs:attribute name="Inserted" type="xs:boolean" default="false">
                <xs:annotation>
                    <xs:documentation>Denotes if the comment is at the end of the line (using //) or inside the line (using /* */)</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="Informative" type="xs:boolean" default="false">
                <xs:annotation>
```

<xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

<xs:simpleType name="GUID_TP">

<xs:restriction base="xs:string">

<xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}"/>

</xs:restriction>

</xs:simpleType>

<xs:element name="IntegerAttribute" type="IntegerAttribute_T">

<xs:annotation>

<xs:documentation>A member attribute with a type restriction of integer.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:complexType name="IntegerAttribute_T">

<xs:annotation>

<xs:documentation>Not for LAD/FBD.</xs:documentation>

</xs:annotation>

<xs:simpleContent>

<xs:extension base="xs:integer">

<xs:attribute name="Name" type="xs:string" use="required"/>

<xs:attribute name="Informative" type="xs:boolean" default="false">

<xs:annotation>

<xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>

</xs:annotation>

```
                    </xs:attribute>
                    <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
                        <xs:annotation>
                            <xs:documentation>An attribute of attribute, denotes if it is defined by a user or the system itself. In V14, if exists it is always true.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:simpleType name="Lang_TP">
            <xs:restriction base="xs:string">
                <xs:pattern value="[a-zA-Z]{2}-[a-zA-Z]{2}"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="LineComment" type="LineComment_T">
            <xs:annotation>
                <xs:documentation>Not for LAD/FBD </xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:complexType name="LineComment_T">
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>For NumBLs. NumBLs is the count of the blank spaces before the actual text in the LineComment. This is informative.</xs:documentation>
                    </xs:annotation>
                </xs:element>
```

```
                    <xs:element ref="Text">
                        <xs:annotation>
                            <xs:documentation>the value of the comment</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="Inserted" type="xs:boolean" default="false">
                    <xs:annotation>
                        <xs:documentation>Denotes if the comment is at the end of the line (using //) or inside the line (using /* */)</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:complexType>
            <xs:element name="MultiLanguageText" type="MultiLanguageText_T"/>
            <xs:complexType name="MultiLanguageText_T">
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="Lang" type="Lang_TP" use="required"/>
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
            <xs:element name="RealAttribute" type="RealAttribute_T">
                <xs:annotation>
                    <xs:documentation>A member attribute with a type restriction of real.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:complexType name="RealAttribute_T">
```

```
            <xs:simpleContent>
                <xs:extension base="xs:double">
                    <xs:attribute name="Name" type="xs:string" use="required"/>
                    <xs:attribute name="Informative" type="xs:boolean" default="false">
                        <xs:annotation>
                            <xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                    <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
                        <xs:annotation>
                            <xs:documentation>An attribute of attribute, denotes if it is defined by a user or the system itself. In V14, if exists it is always true.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
        <xs:simpleType name="SectionName_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="None"/>
                <xs:enumeration value="Input"/>
                <xs:enumeration value="Return"/>
                <xs:enumeration value="Output"/>
                <xs:enumeration value="InOut"/>
                <xs:enumeration value="Static"/>
                <xs:enumeration value="Temp"/>
```

```
                <xs:enumeration value="Constant"/>
                <xs:enumeration value="Base"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SimaticName_TP">
            <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:simpleType name="SimaticType_TE">
            <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:element name="StringAttribute" type="StringAttribute_T">
            <xs:annotation>
                <xs:documentation>A member attribute with a type restriction of string.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:complexType name="StringAttribute_T">
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="Name" type="xs:string" use="required"/>
                    <xs:attribute name="Informative" type="xs:boolean" default="false">
                        <xs:annotation>
                            <xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>
                        </xs:annotation>
                    </xs:attribute>
                    <xs:attribute name="SystemDefined" type="xs:boolean" default="true">
                        <xs:annotation>
```

<xs:documentation>An attribute of attribute, denotes if it is defined by a user or the system itself. In V14, if exists it is always true.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:element name="DateAttribute" type="DateAttribute_T"/>

<xs:complexType name="DateAttribute_T">

<xs:simpleContent>

<xs:extension base="xs:dateTime">

<xs:attribute name="Name" type="xs:string" use="required"/>

<xs:attribute name="Informative" type="xs:boolean" default="false">

<xs:annotation>

<xs:documentation>Exported only with ReadOnly option, ignored during import.</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="SystemDefined" type="xs:boolean" default="true">

<xs:annotation>

<xs:documentation>An attribute of attribute, denotes if it is defined by a user or the system itself. In V14, if exists it is always true.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

```
<xs:element name="Text" type="Text_T"/>
<xs:complexType name="Text_T">
	<xs:simpleContent>
		<xs:extension base="xs:string"/>
	</xs:simpleContent>
</xs:complexType>
<xs:element name="Token" type="Token_T"/>
<xs:complexType name="Token_T">
	<xs:sequence minOccurs="0">
		<xs:element ref="IntegerAttribute" minOccurs="0">
			<xs:annotation>
				<xs:documentation>For NumBLs. NumBLs is the count of the blank spaces at the start.This is informative.</xs:documentation>
			</xs:annotation>
		</xs:element>
	</xs:sequence>
	<xs:attribute name="UId" type="xs:int">
		<xs:annotation>
			<xs:documentation>Not allowed in STL</xs:documentation>
		</xs:annotation>
	</xs:attribute>
	<xs:attribute name="Text" type="Token_TE" use="required"/>
</xs:complexType>
<xs:simpleType name="Token_TE">
	<xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="VersionString_TP">
	<xs:restriction base="xs:string">
```

```
                <xs:pattern value="[0-9]+(.[0-9]+){0,3}"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="ViewInfo" type="ViewInfo_T"/>
        <xs:complexType name="ViewInfo_T">
            <xs:attribute name="Start" type="xs:boolean" use="optional"/>
            <xs:attribute name="XPos" type="xs:int" use="optional"/>
            <xs:attribute name="YPos" type="xs:int" use="optional"/>
            <xs:attribute name="Width" type="xs:int" use="optional"/>
            <xs:attribute name="Height" type="xs:int" use="optional"/>
        </xs:complexType>
</xs:schema>
```

1.1.2 SW.InterfaceSections.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:include schemaLocation="SW.Common.xsd"/>
        <xs:simpleType name="Accessibility_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Public"/>
                <xs:enumeration value="Internal"/>
                <xs:enumeration value="Protected"/>
                <xs:enumeration value="Private"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="IndexPath_TP">
            <xs:restriction base="xs:string">
```

```
                <xs:pattern value="-?\d+(,-?\d+)*(;(-?\d+(,-?\d+)*))?"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="Member" type="Member_T"/>
        <xs:complexType name="Member_T">
            <xs:sequence>
                <xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element ref="Member"/>
                    <xs:element ref="Sections"/>
                    <xs:element ref="StartValue"/>
                    <xs:element ref="Comment"/>
                </xs:choice>
            </xs:sequence>
            <xs:attribute name="Name" type="xs:string" use="required"/>
            <xs:attribute name="Datatype" type="SimaticType_TE" use="required"/>
            <xs:attribute name="Version" type="xs:string" use="optional">
                <xs:annotation>
                    <xs:documentation>The version of the library type to use. Previous to this, the version was written inside the Datatype attribute itself, like "dtl:v1.0". Now, this is written in two separate attributes, to mitigate problems with weird names ("dtl:v1.0" could be a UDT name!).</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="Remanence" type="Remanence_TE" default="NonRetain"/>
            <xs:attribute name="Accessibility" type="Accessibility_TE" default="Public"/>
            <xs:attribute name="Informative" type="xs:boolean" default="false"/>
        </xs:complexType>
        <xs:simpleType name="MemberAttributes_TE">
```

```
            <xs:restriction base="xs:string">
                <xs:enumeration value="CodeReadOnly">
                    <xs:annotation>
                        <xs:documentation>Write acces only inside function</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="At">
                    <xs:annotation>
                        <xs:documentation>string: Member shares offset with another member in this structure</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="SetPoint">
                    <xs:annotation>
                        <xs:documentation>boolean: Member can be synchronized with work memory</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="UserVisible">
                    <xs:annotation>
                        <xs:documentation>boolean: Editor does not show the member</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="UserReadOnly">
                    <xs:annotation>
                        <xs:documentation>boolean: User cannot change member name</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
```

<xs:enumeration value="UserDeletable">

<xs:annotation>

<xs:documentation>boolean: Editor does not allow to delete the member</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="HmiAccessible">

<xs:annotation>

<xs:documentation>boolean: No HMI access, no structure item</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="HmiVisible">

<xs:annotation>

<xs:documentation>boolean: Filter to reduce the number of members shown in the first place</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="Offset">

<xs:annotation>

<xs:documentation>integer: </xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="PaddedSize">

<xs:annotation>

<xs:documentation>integer: </xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="HiddenAssignment">

<xs:annotation>

<xs:documentation>boolean: Hide assignement at call if matches with PredefinedAssignment</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="PredefinedAssignment">

<xs:annotation>

<xs:documentation>string: Input for the paramter used when call is placed</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="ReadOnlyAssignment">

<xs:annotation>

<xs:documentation>boolean: The user cannot change the predefined assignement at the call</xs:documentation>

</xs:annotation>

</xs:enumeration>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="Remanence_TE">

<xs:restriction base="xs:string">

<xs:enumeration value="SetInIDB"/>

<xs:enumeration value="NonRetain"/>

<xs:enumeration value="Retain"/>

</xs:restriction>

</xs:simpleType>

<xs:element name="Section" type="Section_T"/>

<xs:complexType name="Section_T">

<xs:sequence>

<xs:element ref="Sections" minOccurs="0" maxOccurs="1">

<xs:annotation>

<xs:documentation>Base Class</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element ref="Member" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="Name" type="SectionName_TE" use="required"/>

</xs:complexType>

<xs:element name="Sections" type="Sections_T"/>

<xs:complexType name="Sections_T">

<xs:sequence>

<xs:element ref="AttributeList" minOccurs="0" maxOccurs="1"/>

<xs:element ref="Section" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="Datatype" type="SimaticType_TE"/>

<xs:attribute name="Version" type="xs:string" use="optional"/>

</xs:complexType>

<xs:element name="StartValue" type="StartValue_T"/>

<xs:complexType name="StartValue_T">

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute name="Path" type="IndexPath_TP"/>

<xs:attribute name="ConstantName" type="SimaticName_TP"/>

<xs:attribute name="IsBulkValue" type="xs:boolean" default="false"/>

<xs:attribute name="Informative" type="xs:boolean" default="false"/>

</xs:extension>

```
            </xs:simpleContent>
        </xs:complexType>
</xs:schema>
```

1.1.3 SW.PlcBlocks.Access.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:include schemaLocation="SW.Common.xsd"/>
        <xs:element name="Access" type="Access_T"/>
        <xs:complexType name="Access_T">
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for NumBLs. NumBLs is informative. Not for LAD/FBD.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:choice>
                    <xs:element ref="Label"/>
                    <xs:element ref="Constant"/>
                    <xs:element ref="CallInfo">
                        <xs:annotation>
                            <xs:documentation>call of a user block. Not in Graph ActionList.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element ref="Instruction">
                        <xs:annotation>
```

<xs:documentation>call of an instruction. Not for LAD/FBD, Graph ActionList.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element ref="Indirect">

<xs:annotation>

<xs:documentation>STL specific</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element ref="Statusword"/>

<xs:element ref="Expression">

<xs:annotation>

<xs:documentation>SCL specific</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element ref="Symbol"/>

<xs:element ref="Address">

<xs:annotation>

<xs:documentation>for absolute addresses</xs:documentation>

</xs:annotation>

</xs:element>

</xs:choice>

<xs:group ref="Comment_G" minOccurs="0"/>

</xs:sequence>

<xs:attribute name="Scope" use="required">

<xs:simpleType>

<xs:restriction base="Scope_TE"/>

```
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="Type" type="SimaticName_TP"/>
            <xs:attribute name="UId" type="xs:int" use="optional">
                <xs:annotation>
                    <xs:documentation>Not allowed in STL</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>
        <xs:element name="Address" type="Address_T"/>
        <xs:complexType name="Address_T">
            <xs:attribute name="Area" type="Area_TE" use="required"/>
            <xs:attribute name="BlockNumber" type="xs:int" use="optional">
                <xs:annotation>
                    <xs:documentation>for DB access</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="BitOffset" type="xs:int" use="required">
                <xs:annotation>
                    <xs:documentation>In general it is Byte * 8 + Bit. But if it is used for addressing a DB we will find the number of the DB here (e.g. "DB12" ->12).</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="Informative" type="xs:boolean" default="false">
                <xs:annotation>
                    <xs:documentation>if true, the import unnoted it</xs:documentation>
                </xs:annotation>
```

```
            </xs:attribute>
        </xs:complexType>
        <xs:element name="AbsoluteOffset" type="AbsoluteOffset_T"/>
        <xs:complexType name="AbsoluteOffset_T">
            <xs:attribute name="BitOffset" type="xs:int" use="required">
                <xs:annotation>
                    <xs:documentation>Byte * 8 + Bit</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>
        <xs:simpleType name="Area_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="None"/>
                <xs:enumeration value="Periphery">
                    <xs:annotation>
                        <xs:documentation>only used for incomplete
pointer</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="PeripheryInput"/>
                <xs:enumeration value="PeripheryOutput"/>
                <xs:enumeration value="Input"/>
                <xs:enumeration value="Output"/>
                <xs:enumeration value="Memory"/>
                <xs:enumeration value="FB"/>
                <xs:enumeration value="FC"/>
                <xs:enumeration value="DB">
                    <xs:annotation>
```

<xs:documentation>partly qualified access with DB register</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="DI">

<xs:annotation>

<xs:documentation>partly qualified access with DI register</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="Timer"/>

<xs:enumeration value="Counter"/>

<xs:enumeration value="LocalC">

<xs:annotation>

<xs:documentation>Classic Local Stack</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="LocalN">

<xs:annotation>

<xs:documentation>Nena Local Stack</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="DBc">

<xs:annotation>

<xs:documentation>fully qualified access</xs:documentation>

</xs:annotation>

</xs:enumeration>

<xs:enumeration value="DBr">

```
                    <xs:annotation>
                        <xs:documentation>fully qualified access</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="DBv">
                    <xs:annotation>
                        <xs:documentation>fully qualified access</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="BlockType_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="DB"/>
                <xs:enumeration value="FB"/>
                <xs:enumeration value="FC"/>
                <xs:enumeration value="OB"/>
                <xs:enumeration value="UDT"/>
                <xs:enumeration value="FBT"/>
                <xs:enumeration value="FCT"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="CallInfo">
            <xs:complexType>
                <xs:complexContent>
                    <xs:extension base="CallInfo_T"/>
                </xs:complexContent>
```

```
            </xs:complexType>
        </xs:element>
        <xs:complexType name="CallInfo_T">
            <xs:annotation>
                <xs:documentation>Not for LAD/FBD. </xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for BlockNumber. BlockNumber is informative.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="DateAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for ParameterModifiedTS. ParameterModifiedTS is informative</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:sequence minOccurs="0">
                    <xs:element ref="Access" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:group ref="Comment_G" minOccurs="0"/>
                <xs:sequence minOccurs="0">
                    <xs:element ref="Token">
                        <xs:annotation>
                            <xs:documentation>the COMMA, only if separated. Not for LAD/FBD</xs:documentation>
                        </xs:annotation>
                    </xs:element>
```

<xs:group ref="Comment_G" minOccurs="0"/>

</xs:sequence>

<xs:element ref="Instance" minOccurs="0"/>

<xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="Name" type="SimaticName_TP" use="required"/>

<xs:attribute name="BlockType" type="BlockType_TE" use="optional"/>

</xs:complexType>

<xs:element name="Component" type="Component_T"/>

<xs:complexType name="Component_T">

<xs:sequence minOccurs="0">

<xs:annotation>

<xs:documentation>For the indices of an array</xs:documentation>

</xs:annotation>

<xs:element ref="Access" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="Name" type="xs:string" use="required"/>

<xs:attribute name="SliceAccessModifier" type="SliceAccessModifier_TP" default="undef"/>

<xs:attribute name="SimpleAccessModifier" type="SimpleAccessModifier_TP" default="None"/>

</xs:complexType>

<xs:element name="Constant" type="Constant_T"/>

<xs:complexType name="Constant_T">

<xs:sequence>

<xs:element ref="ConstantValue"/>

<xs:element ref="StringAttribute" minOccurs="0" maxOccurs="2">

<xs:annotation>

<xs:documentation>for Format and FormatFlags. They are informative..</xs:documentation>

</xs:annotation>

</xs:element>

<xs:element ref="IntegerAttribute" minOccurs="0" maxOccurs="1">

<xs:annotation>

<xs:documentation>for Size. Size is informative.</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

<xs:attribute name="Name" type="SimaticName_TP"/>

</xs:complexType>

<xs:element name="ConstantValue" type="ConstantValue_T"/>

<xs:complexType name="ConstantValue_T">

<xs:simpleContent>

<xs:extension base="xs:string"/>

</xs:simpleContent>

</xs:complexType>

<xs:element name="Expression" type="Expression_T"/>

<xs:complexType name="Expression_T">

<xs:sequence maxOccurs="unbounded">

<xs:choice>

<xs:element ref="Access"/>

<xs:element ref="Token"/>

</xs:choice>

<xs:group ref="Comment_G" minOccurs="0"/>

</xs:sequence>

<xs:attribute name="UId" type="xs:int"/>

```
</xs:complexType>
<xs:simpleType name="Format_TE">
	<xs:restriction base="xs:string">
		<xs:enumeration value="Real"/>
		<xs:enumeration value="Bin"/>
		<xs:enumeration value="DecSigned"/>
		<xs:enumeration value="DecUnsigned"/>
		<xs:enumeration value="Pointer"/>
		<xs:enumeration value="CharSequence"/>
		<xs:enumeration value="DecSequence"/>
		<xs:enumeration value="Hex"/>
		<xs:enumeration value="S5Count"/>
		<xs:enumeration value="Time"/>
		<xs:enumeration value="Date"/>
		<xs:enumeration value="TimeOfDay"/>
		<xs:enumeration value="S5Time"/>
		<xs:enumeration value="Bool"/>
		<xs:enumeration value="Oct"/>
		<xs:enumeration value="Bcd"/>
		<xs:enumeration value="DateAndTime"/>
		<xs:enumeration value="String"/>
		<xs:enumeration value="Any"/>
		<xs:enumeration value="Number"/>
		<xs:enumeration value="Char"/>
		<xs:enumeration value="HexSequence"/>
	</xs:restriction>
</xs:simpleType>
<xs:simpleType name="FormatFlags_TE">
	<xs:restriction base="xs:string">
```

```
                        <xs:pattern value="None"/>
                        <xs:pattern
value="((Lower|Format|Size|Under|Exp|TypeQualifier)(,\s*)?)*"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:element name="Indirect" type="Indirect_T"/>
        <xs:complexType name="Indirect_T">
                <xs:sequence minOccurs="0">
                        <xs:choice minOccurs="0" maxOccurs="unbounded">
                                <xs:element ref="Token"/>
                                <xs:group ref="Comment_G"/>
                        </xs:choice>
                        <xs:element ref="Access" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="Width" type="Width_TE" use="required"/>
                <xs:attribute name="Area" type="Area_TE" use="optional"/>
                <xs:attribute name="Register" type="Register_TE" use="optional"/>
                <xs:attribute name="BitOffset" use="optional"/>
        </xs:complexType>
        <xs:element name="Instance" type="Instance_T"/>
        <xs:element name="Instruction" type="Instruction_T"/>
        <xs:complexType name="Instruction_T">
                <xs:sequence>
                        <xs:sequence minOccurs="0">
                                <xs:element ref="Access" maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:group ref="Comment_G" minOccurs="0"/>
                        <xs:sequence minOccurs="0">
                                <xs:element ref="Token">
```

<xs:annotation>
<xs:documentation>the COMMA, only if separated</xs:documentation>
</xs:annotation>
</xs:element>
<xs:group ref="Comment_G" minOccurs="0"/>
</xs:sequence>
<xs:element ref="TemplateValue" minOccurs="0" maxOccurs="unbounded"/>
<xs:element ref="Instance" minOccurs="0"/>
<xs:element ref="Parameter" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute name="Name" use="required">
<xs:simpleType>
<xs:restriction base="SimaticName_TP">
<xs:minLength value="1"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="Version" type="VersionString_TP"/>
</xs:complexType>
<xs:simpleType name="InterfaceFlags_TP">
<xs:restriction base="xs:string">
<xs:pattern value="None"/>
<xs:pattern value="((Mandatory|S7_Visible)(,\s*)?)*"/>
</xs:restriction>
</xs:simpleType>
<xs:element name="Label" type="Label_T"/>
<xs:complexType name="Label_T">

```
            <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
        </xs:complexType>
        <xs:element name="Parameter" type="Parameter_T"/>
        <xs:complexType name="Parameter_T">
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for NumBLs. NumBLs is informative</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="StringAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for InterfaceFlags. InterfaceFlags is informative</xs:documentation>
                        <xs:documentation>The type of the value should be InterfaceFlags_TP</xs:documentation>
                        <xs:documentation>The default value is "S7_Visible"</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:group ref="Comment_G" minOccurs="0"/>
                <xs:group ref="Access_G" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
            <xs:attribute name="Section" type="SectionName_TE" use="required"/>
            <xs:attribute name="Type" type="SimaticType_TE"/>
            <xs:attribute name="TemplateReference" type="xs:string"/>
        </xs:complexType>
        <xs:simpleType name="Register_TE">
```

```
            <xs:restriction base="xs:string">
                <xs:enumeration value="AR1"/>
                <xs:enumeration value="AR2"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Scope_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Undef">
                    <xs:annotation>
                        <xs:documentation>Symbols we do not know what they are</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
                <xs:enumeration value="GlobalConstant"/>
                <xs:enumeration value="LocalConstant"/>
                <xs:enumeration value="GlobalVariable"/>
                <xs:enumeration value="LocalVariable"/>
                <xs:enumeration value="Instruction"/>
                <xs:enumeration value="Label"/>
                <xs:enumeration value="Unnamed">
                    <xs:annotation>
                        <xs:documentation>includes Indirect, Statusword, Expressions, Address</xs:documentation>
                    </xs:annotation>
                </xs:enumeration>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SimpleAccessModifier_TP">
            <xs:restriction base="xs:string">
```

```
                        <xs:pattern
value="None|((Periphery|QualityInformation)(,\s*)?)*"/>
                    </xs:restriction>
                </xs:simpleType>
                <xs:simpleType name="SimpleType_TE">
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="undef"/>
                        <xs:enumeration value="Bool"/>
                        <xs:enumeration value="Byte"/>
                        <xs:enumeration value="Char"/>
                        <xs:enumeration value="Word"/>
                        <xs:enumeration value="Int"/>
                        <xs:enumeration value="DWord"/>
                        <xs:enumeration value="DInt"/>
                        <xs:enumeration value="Real"/>
                        <xs:enumeration value="LReal"/>
                        <xs:enumeration value="Timer"/>
                        <xs:enumeration value="S5Time"/>
                        <xs:enumeration value="ARef"/>
                        <xs:enumeration value="Any"/>
                        <xs:enumeration value="Time"/>
                        <xs:enumeration value="S5Count"/>
                        <xs:enumeration value="Counter"/>
                        <xs:enumeration value="Block_DB"/>
                        <xs:enumeration value="Block_FB"/>
                        <xs:enumeration value="Block_FC"/>
                        <xs:enumeration value="Block_SFB"/>
                        <xs:enumeration value="Block_UDT"/>
                        <xs:enumeration value="Multi_FB"/>
```

```
                <xs:enumeration value="Multi_SFB"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="SliceAccessModifier_TP">
            <xs:restriction base="xs:string">
                <xs:pattern value="([xbwdXBWD]\d+)|undef"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="Statusword" type="Statusword_T">
            <xs:annotation>
                <xs:documentation>Only for S7-300/400/WinAC</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:complexType name="Statusword_T">
            <xs:attribute name="Combination" type="Statusword_TE" use="required"/>
        </xs:complexType>
        <xs:simpleType name="Statusword_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="BR"/>
                <xs:enumeration value="OV"/>
                <xs:enumeration value="OS"/>
                <xs:enumeration value="EQ"/>
                <xs:enumeration value="NE"/>
                <xs:enumeration value="GT"/>
                <xs:enumeration value="LT"/>
                <xs:enumeration value="GE"/>
                <xs:enumeration value="LE"/>
```

```
                    <xs:enumeration value="UO"/>
                    <xs:enumeration value="NU"/>
                </xs:restriction>
            </xs:simpleType>
            <xs:element name="Symbol" type="Symbol_T"/>
            <xs:complexType name="Symbol_T">
                <xs:sequence>
                    <xs:sequence maxOccurs="unbounded">
                        <xs:choice>
                            <xs:element ref="Component"/>
                            <xs:element ref="AbsoluteOffset"/>
                            <xs:element ref="Token">
                                <xs:annotation>
                                    <xs:documentation>the DOT; only if separated. Not in Graph ActionList, not in LAD/FBD.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                        </xs:choice>
                        <xs:group ref="Comment_G" minOccurs="0"/>
                    </xs:sequence>
                    <xs:element ref="Address" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>additional address for a symbol. it is informative</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="UId" type="xs:int">
                    <xs:annotation>
```

<xs:documentation>Not allowed in STL</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Scope" type="Scope_TE"/>

</xs:complexType>

<xs:complexType name="Instance_T">

<xs:sequence>

<xs:sequence maxOccurs="unbounded">

<xs:choice>

<xs:element ref="Component"/>

<xs:element ref="AbsoluteOffset"/>

<xs:element ref="Token">

<xs:annotation>

<xs:documentation>the DOT; only if separated. Not in Graph ActionList, not in LAD/FBD.</xs:documentation>

</xs:annotation>

</xs:element>

</xs:choice>

<xs:group ref="Comment_G" minOccurs="0"/>

</xs:sequence>

<xs:element ref="Address" minOccurs="0">

<xs:annotation>

<xs:documentation>additional address for a symbol. it is informative</xs:documentation>

</xs:annotation>

</xs:element>

</xs:sequence>

<xs:attribute name="UId" type="xs:int">

<xs:annotation>

<xs:documentation>Not allowed in STL</xs:documentation>

</xs:annotation>

</xs:attribute>

<xs:attribute name="Scope" type="Scope_TE" use="required"/>

</xs:complexType>

<xs:element name="TemplateValue" type="TemplateValue_T"/>

<xs:complexType name="TemplateValue_T">

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute name="Name" type="SimaticName_TP" use="required"/>

<xs:attribute name="Type" type="TemplateType_TE" use="required"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

<xs:simpleType name="TemplateType_TE">

<xs:restriction base="xs:string">

<xs:enumeration value="Cardinality"/>

<xs:enumeration value="Type"/>

<xs:enumeration value="Operation"/>

</xs:restriction>

</xs:simpleType>

<xs:simpleType name="Width_TE">

<xs:restriction base="xs:string">

<xs:enumeration value="None"/>

<xs:enumeration value="Bit"/>

<xs:enumeration value="Byte"/>

<xs:enumeration value="Word"/>

```
                <xs:enumeration value="Offset"/>
                <xs:enumeration value="Double"/>
                <xs:enumeration value="Pointer"/>
                <xs:enumeration value="Long"/>
                <xs:enumeration value="Any"/>
                <xs:enumeration value="Block"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:group name="Access_G">
            <xs:sequence>
                <xs:element ref="Access"/>
                <xs:group ref="Comment_G" minOccurs="0"/>
            </xs:sequence>
        </xs:group>
</xs:schema>
```

1.1.4 SW.PlcBlocks.CompileUnitCommon.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:include schemaLocation="SW.PlcBlocks.Access.xsd"/>
        <xs:element name="LabelDeclaration" type="LabelDeclaration_T"/>
        <xs:complexType name="LabelDeclaration_T">
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for NumBLs. NumBLs is informative</xs:documentation>
                    </xs:annotation>
```

```
                </xs:element>
                <xs:element ref="Label"/>
                <xs:group ref="Comment_G" minOccurs="0"/>
                <xs:sequence minOccurs="0">
                    <xs:element ref="Token">
                        <xs:annotation>
                            <xs:documentation>the COLON; only if separated</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:group ref="Comment_G" minOccurs="0"/>
                </xs:sequence>
            </xs:sequence>
            <xs:attribute name="UId" type="xs:int">
                <xs:annotation>
                    <xs:documentation>Not allowed in STL</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>
</xs:schema>
```

1.1.5 SW.PlcBlocks.Graph.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="SW.PlcBlocks.Access.xsd"/>
    <xs:include schemaLocation="SW.PlcBlocks.LADFBD.xsd"/>
    <xs:complexType name="Action_T">
```

```
        <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="Access"/>
                <xs:element ref="Token"/>
                <xs:group ref="Comment_G" minOccurs="0"/>
        </xs:choice>
        <xs:attribute name="Event" type="Event_TE"/>
        <xs:attribute name="Interlock" type="xs:boolean"/>
        <xs:attribute name="Qualifier" type="Qualifier_TE"/>
</xs:complexType>
<xs:simpleType name="Event_TE">
        <xs:restriction base="xs:string">
                <xs:enumeration value=""/>
                <xs:enumeration value="A1"/>
                <xs:enumeration value="L0"/>
                <xs:enumeration value="L1"/>
                <xs:enumeration value="R1"/>
                <xs:enumeration value="S0"/>
                <xs:enumeration value="S1"/>
                <xs:enumeration value="V0"/>
                <xs:enumeration value="V1"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Qualifier_TE">
        <xs:restriction base="xs:string">
                <xs:enumeration value=""/>
                <xs:enumeration value="CD"/>
                <xs:enumeration value="CR"/>
                <xs:enumeration value="CS"/>
                <xs:enumeration value="CU"/>
```

```
                <xs:enumeration value="D"/>
                <xs:enumeration value="L"/>
                <xs:enumeration value="N"/>
                <xs:enumeration value="ON"/>
                <xs:enumeration value="OFF"/>
                <xs:enumeration value="R"/>
                <xs:enumeration value="S"/>
                <xs:enumeration value="TD"/>
                <xs:enumeration value="TF"/>
                <xs:enumeration value="TL"/>
                <xs:enumeration value="TR"/>
        </xs:restriction>
</xs:simpleType>
<xs:element name="Actions" type="Actions_T"/>
<xs:complexType name="Actions_T">
        <xs:sequence>
                <xs:element ref="Title" minOccurs="0"/>
                <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="Action"/>
                </xs:sequence>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="AlarmSupportingLanguageModule_T">
        <xs:sequence>
                <xs:element ref="Title" minOccurs="0"/>
                <xs:element ref="AlarmText" minOccurs="0"/>
                <xs:element ref="FlgNet"/>
        </xs:sequence>
```

```
            <xs:attribute name="ProgrammingLanguage" type="ProgrammingLanguage_TE" use="required"/>
        </xs:complexType>
        <xs:element name="AlarmText" type="AlarmText_T"/>
        <xs:complexType name="AlarmText_T">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Temporary change for enable of empty alarm text because of the graph alarm handling reconstruction.</xs:documentation>
                </xs:annotation>
                <xs:element ref="MultiLanguageText"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="Branch_T">
            <xs:attribute name="Number" type="xs:int" use="required"/>
            <xs:attribute name="Type" type="Branch_TE" use="required"/>
            <xs:attribute name="Cardinality" type="xs:int" use="required"/>
        </xs:complexType>
        <xs:simpleType name="Branch_TE">
            <xs:restriction base="xs:string">
                <xs:enumeration value="SimBegin"/>
                <xs:enumeration value="SimEnd"/>
                <xs:enumeration value="AltBegin"/>
                <xs:enumeration value="AltEnd"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:element name="Branches" type="Branches_T"/>
        <xs:complexType name="Branches_T">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
                <xs:element ref="Branch"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="BranchRef" type="BranchRef_T"/>
<xs:complexType name="BranchRef_T">
        <xs:attribute name="Number" type="xs:int" use="required"/>
        <xs:attribute name="In" type="xs:int"/>
        <xs:attribute name="Out" type="xs:int"/>
</xs:complexType>
<xs:element name="Connection" type="Connection_T"/>
<xs:complexType name="Connection_T">
        <xs:sequence>
                <xs:element ref="NodeFrom"/>
                <xs:element ref="NodeTo"/>
                <xs:element ref="LinkType"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="EndConnection"/>
<xs:element name="Graph" type="Graph_T"/>
<xs:complexType name="AlarmsSettings_T">
        <xs:sequence>
                <xs:element ref="AlarmSupervisionCategories"/>
                <xs:element ref="AlarmInterlockCategory"/>
                <xs:element ref="AlarmWarningCategory"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="AlarmsSettings" type="AlarmsSettings_T"/>
<xs:complexType name="Graph_T">
        <xs:sequence>
```

```
                <xs:element ref="PreOperations"/>
                <xs:element ref="Sequence" maxOccurs="unbounded"/>
                <xs:element ref="PostOperations"/>
                <xs:element ref="AlarmsSettings"/>
        </xs:sequence>
</xs:complexType>
<xs:element name="IdentRef" type="IdentRef_T"/>
<xs:complexType name="IdentRef_T">
        <xs:sequence>
                <xs:element ref="Comment" minOccurs="0"/>
                <xs:element ref="ViewInfo" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="PartAttribute_G"/>
</xs:complexType>
<xs:element name="Interlock" type="AlarmSupportingLanguageModule_T"/>
<xs:element name="Interlocks" type="Interlocks_T"/>
<xs:complexType name="Interlocks_T">
        <xs:sequence>
                <xs:element ref="Interlock"/>
        </xs:sequence>
</xs:complexType>
<xs:simpleType name="Link_TE">
        <xs:restriction base="xs:string">
                <xs:enumeration value="Direct"/>
                <xs:enumeration value="Jump"/>
        </xs:restriction>
</xs:simpleType>
<xs:complexType name="Node_T">
        <xs:choice>
```

```
                    <xs:element ref="StepRef"/>
                    <xs:element ref="TransitionRef"/>
                    <xs:element ref="BranchRef"/>
                    <xs:element ref="EndConnection"/>
                </xs:choice>
            </xs:complexType>
            <xs:element name="NodeFrom" type="Node_T"/>
            <xs:element name="NodeTo" type="Node_T"/>
            <xs:element name="LinkType" type="Link_TE"/>
            <xs:element name="PermanentOperation" type="PermanentOperation_T"/>
            <xs:complexType name="PermanentOperation_T">
                <xs:sequence>
                    <xs:element ref="Title" minOccurs="0"/>
                    <xs:element ref="FlgNet" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="ProgrammingLanguage"
type="ProgrammingLanguage_TE" use="required"/>
            </xs:complexType>
            <xs:complexType name="PermanentOperations_T">
                <xs:sequence>
                    <xs:element ref="Title" minOccurs="0"/>
                    <xs:element ref="Comment" minOccurs="0"/>
                    <xs:element ref="PermanentOperation" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
            <xs:element name="PostOperations" type="PermanentOperations_T"/>
            <xs:element name="PreOperations" type="PermanentOperations_T"/>
            <xs:simpleType name="ProgrammingContext_TE">
```

```
        <xs:restriction base="xs:string">
                <xs:enumeration value="Plain"/>
                <xs:enumeration value="GraphTransition"/>
                <xs:enumeration value="GraphSupervision"/>
                <xs:enumeration value="GraphInterlock"/>
                <xs:enumeration value="GraphActions"/>
                <xs:enumeration value="PreOperation"/>
                <xs:enumeration value="PostOperation"/>
        </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ProgrammingLanguage_TE">
        <xs:restriction base="xs:string">
                <xs:enumeration value="STL"/>
                <xs:enumeration value="FBD"/>
                <xs:enumeration value="LAD"/>
                <xs:enumeration value="FBD_IEC"/>
                <xs:enumeration value="LAD_IEC"/>
                <xs:enumeration value="GRAPH"/>
                <xs:enumeration value="DB"/>
                <xs:enumeration value="SDB"/>
                <xs:enumeration value="DB_CPU"/>
                <xs:enumeration value="FB_IDB"/>
                <xs:enumeration value="SFB_IDB"/>
                <xs:enumeration value="DT_DB"/>
                <xs:enumeration value="SCL"/>
        </xs:restriction>
</xs:simpleType>
<xs:element name="Sequence" type="Sequence_T"/>
<xs:complexType name="Sequence_T">
```

```
            <xs:sequence>
                <xs:element ref="Title" minOccurs="0"/>
                <xs:element ref="Comment" minOccurs="0"/>
                <xs:element ref="Steps"/>
                <xs:element ref="Transitions"/>
                <xs:element ref="Branches"/>
                <xs:element ref="Connections"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="Step" type="Step_T"/>
        <xs:complexType name="Step_T">
            <xs:sequence>
                <xs:element ref="Comment" minOccurs="0"/>
                <xs:element ref="Actions"/>
                <xs:element ref="Supervisions"/>
                <xs:element ref="Interlocks"/>
            </xs:sequence>
            <xs:attribute name="IsMissing" type="xs:boolean" default="false"/>
            <xs:attribute name="Number" type="xs:int" use="required"/>
            <xs:attribute name="Init" type="xs:boolean" default="false"/>
            <xs:attribute name="Name" use="required"/>
            <xs:attribute name="MaximumStepTime" type="xs:string"
use="optional"/>
            <xs:attribute name="WarningTime" type="xs:string" use="optional"/>
        </xs:complexType>
        <xs:element name="StepRef" type="StepRef_T"/>
        <xs:complexType name="StepRef_T">
            <xs:attribute name="Number" type="xs:int" use="required"/>
        </xs:complexType>
```

```
<xs:element name="Steps" type="Steps_T"/>
<xs:complexType name="Steps_T">
	<xs:sequence maxOccurs="unbounded">
		<xs:element ref="Step"/>
	</xs:sequence>
</xs:complexType>
<xs:element name="Connections" type="Connections_T"/>
<xs:complexType name="Connections_T">
	<xs:sequence maxOccurs="unbounded">
		<xs:element ref="Connection"/>
	</xs:sequence>
</xs:complexType>
<xs:element name="Supervision" type="AlarmSupportingLanguageModule_T"/>
<xs:element name="Supervisions" type="Supervisions_T"/>
<xs:complexType name="Supervisions_T">
	<xs:sequence>
		<xs:element ref="Supervision"/>
	</xs:sequence>
</xs:complexType>
<xs:element name="Title" type="Comment_T"/>
<xs:complexType name="Transition_T">
	<xs:sequence>
		<xs:element ref="Comment" minOccurs="0"/>
		<xs:element ref="FlgNet"/>
	</xs:sequence>
	<xs:attribute name="IsMissing" type="xs:boolean" default="false"/>
	<xs:attribute name="Name" use="required"/>
	<xs:attribute name="Number" type="xs:int" use="required"/>
```

```
            <xs:attribute name="ProgrammingLanguage" type="ProgrammingLanguage_TE" use="required"/>
        </xs:complexType>
        <xs:element name="TransitionRef" type="TransitionRef_T"/>
        <xs:complexType name="TransitionRef_T">
            <xs:attribute name="Number" type="xs:int" use="required"/>
        </xs:complexType>
        <xs:element name="Transitions" type="Transitions_T"/>
        <xs:complexType name="Transitions_T">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="Transition"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="Action" type="Action_T"/>
        <xs:element name="Transition" type="Transition_T"/>
        <xs:element name="Branch" type="Branch_T"/>
        <xs:element name="AlarmSupervisionCategories" type="AlarmSupervisionCategories_T"/>
        <xs:complexType name="AlarmSupervisionCategories_T">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="AlarmSupervisionCategory"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="AlarmSupervisionCategory" type="AlarmSupervisionCategory_T"/>
        <xs:complexType name="AlarmSupervisionCategory_T">
            <xs:sequence>
                <xs:element ref="Token" minOccurs="0">
                    <xs:annotation>
```

```
                                        <xs:documentation>Enabler
token</xs:documentation>
                                    </xs:annotation>
                                </xs:element>
                            </xs:sequence>
                            <xs:attribute name="Id" type="xs:unsignedShort" use="required"/>
                            <xs:attribute name="DisplayClass" use="required">
                                <xs:simpleType>
                                    <xs:restriction base="xs:unsignedShort">
                                        <xs:minInclusive value="0"/>
                                        <xs:maxInclusive value="16"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                        </xs:complexType>
                        <xs:element name="AlarmInterlockCategory" type="AlarmCategory_T"/>
                        <xs:element name="AlarmWarningCategory" type="AlarmCategory_T"/>
                        <xs:complexType name="AlarmCategory_T">
                            <xs:attribute name="Id" type="xs:unsignedShort" use="required"/>
                        </xs:complexType>
</xs:schema>
```

1.1.6 SW.PlcBlocks.InstanceSupervisions.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
                        <xs:include schemaLocation="SW.Common.xsd"/>
                        <xs:element name="SupervisionFB">
```

```
            <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="StateStruct">
            <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="Number" type="xs:int"/>
        <xs:element name="Multiinstance">
            <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="BlockTypeSupervisionNumber" type="xs:int"/>
        <xs:element name="BlockInstSupervisionGroups" type="BlockInstSupervisionGroupsType"/>
        <xs:element name="BlockInstSupervisionGroup">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="Multiinstance" minOccurs="0"/>
                    <xs:element ref="BlockInstSupervision" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="BlockInstSupervision">
            <xs:complexType>
```

```
                <xs:sequence>
                    <xs:element ref="Number"/>
                    <xs:element ref="StateStruct"/>
                    <xs:element ref="BlockTypeSupervisionNumber"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:complexType name="BlockInstSupervisionGroupsType">
            <xs:sequence>
                <xs:element ref="BlockInstSupervisionGroup" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
</xs:schema>
```

1.1.7 SW.PlcBlocks.LADFBD.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="SW.PlcBlocks.CompileUnitCommon.xsd"/>
    <xs:element name="Powerrail" type="Powerrail_T"/>
    <xs:complexType name="Powerrail_T"/>
    <xs:element name="Openbranch" type="Openbranch_T"/>
    <xs:complexType name="Openbranch_T"/>
    <xs:element name="OpenCon" type="OpenCon_T"/>
    <xs:complexType name="OpenCon_T">
        <xs:attribute name="UId" type="xs:int" use="required"/>
    </xs:complexType>
    <xs:element name="CallRef" type="CallRef_T"/>
```

```
        <xs:complexType name="CallRef_T">
                <xs:sequence>
                        <xs:element ref="BooleanAttribute" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>for ENENO. ENENO is informative.</xs:documentation>
                                </xs:annotation>
                        </xs:element>
                        <xs:element ref="CallInfo"/>
                        <xs:group ref="PartSequence_G"/>
                </xs:sequence>
                <xs:attributeGroup ref="PartAttribute_G"/>
        </xs:complexType>
        <xs:complexType name="Equation_T">
                <xs:simpleContent>
                        <xs:extension base="xs:string"/>
                </xs:simpleContent>
        </xs:complexType>
        <xs:element name="FlgNet" type="FlgNet_T"/>
        <xs:complexType name="FlgNet_T">
                <xs:sequence>
                        <xs:element ref="Labels" minOccurs="0"/>
                        <xs:element ref="Parts"/>
                        <xs:element ref="Wires" minOccurs="0"/>
                </xs:sequence>
        </xs:complexType>
        <xs:simpleType name="GateName_TE">
                <xs:restriction base="xs:string"/>
        </xs:simpleType>
```

```
<xs:element name="IdentCon" type="IdentCon_T"/>
<xs:complexType name="IdentCon_T">
    <xs:attribute name="UId" type="xs:int" use="required"/>
</xs:complexType>
<xs:element name="InstructionRef" type="InstructionRef_T"/>
<xs:complexType name="InstructionRef_T">
    <xs:sequence>
        <xs:element ref="BooleanAttribute" minOccurs="0">
            <xs:annotation>
                <xs:documentation>for ENENO. The ENENO is not informative for InstructionRef</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element ref="Instance" minOccurs="0"/>
        <xs:group ref="PartSequence_G"/>
    </xs:sequence>
    <xs:attributeGroup ref="PartAttribute_G"/>
    <xs:attribute name="Name" use="required">
        <xs:simpleType>
            <xs:restriction base="SimaticName_TP">
                <xs:minLength value="1"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="Version" type="VersionString_TP"/>
    <xs:attribute name="DisabledENO" type="xs:boolean" default="false"/>
</xs:complexType>
<xs:element name="Invisible" type="Invisible_T">
    <xs:annotation>
```

<xs:documentation>The invisible pins of this part.</xs:documentation>

</xs:annotation>

</xs:element>

<xs:complexType name="Invisible_T">

<xs:attribute name="Name" type="xs:string" use="optional">

<xs:annotation>

<xs:documentation>The name of the invisible pin.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

<xs:element name="Labels" type="Labels_T"/>

<xs:complexType name="Labels_T">

<xs:sequence minOccurs="0" maxOccurs="unbounded">

<xs:element ref="LabelDeclaration"/>

</xs:sequence>

</xs:complexType>

<xs:complexType name="Neg_T">

<xs:attribute name="Name" type="xs:string" use="optional">

<xs:annotation>

<xs:documentation>The name of the negated pin.</xs:documentation>

</xs:annotation>

</xs:attribute>

</xs:complexType>

<xs:element name="Negated" type="Neg_T">

<xs:annotation>

<xs:documentation>The negated pins of this part.</xs:documentation>

```
            </xs:annotation>
        </xs:element>
        <xs:complexType name="AutomaticTyped_T">
            <xs:attribute name="Name" type="xs:string" use="optional">
                <xs:annotation>
                    <xs:documentation>The name of the automatic chosen template parameter. Not for InstructionRef</xs:documentation>
                </xs:annotation>
            </xs:attribute>
        </xs:complexType>
        <xs:element name="AutomaticTyped" type="AutomaticTyped_T"/>
        <xs:element name="Part" type="Part_T"/>
        <xs:element name="Equation" type="Equation_T"/>
        <xs:complexType name="Part_T">
            <xs:sequence>
                <xs:element ref="BooleanAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for ENENO. ENENO is informative.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element ref="Equation" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>The equation of this part. This is only used for the Calculate box.</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:group ref="PartSequence_G"/>
            </xs:sequence>
            <xs:attribute name="Gate" type="GateName_TE"/>
```

```
            <xs:attributeGroup ref="PartAttribute_G"/>
            <xs:attribute name="DisabledENO" type="xs:boolean" default="false"/>
        </xs:complexType>
        <xs:attributeGroup name="PartAttribute_G">
            <xs:attribute name="UId" type="xs:int" use="required"/>
        </xs:attributeGroup>
        <xs:element name="NameCon" type="NameCon_T"/>
        <xs:complexType name="NameCon_T">
            <xs:attribute name="UId" type="xs:int" use="required"/>
            <xs:attribute name="Name" type="PinName_TE" use="required"/>
        </xs:complexType>
        <xs:element name="Parts" type="Parts_T"/>
        <xs:complexType name="Parts_T">
            <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:element ref="Access"/>
                <xs:element ref="Part"/>
                <xs:element ref="CallRef"/>
                <xs:element ref="InstructionRef"/>
            </xs:choice>
        </xs:complexType>
        <xs:group name="PartSequence_G">
            <xs:sequence>
                <xs:element ref="TemplateValue" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element ref="AutomaticTyped" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element ref="Invisible" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element ref="Negated" minOccurs="0"
maxOccurs="unbounded"/>
```

```
                <xs:element ref="Comment" minOccurs="0"/>
        </xs:sequence>
</xs:group>
<xs:simpleType name="PinName_TE">
        <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:element name="Wire" type="Wire_T"/>
<xs:complexType name="Wire_T">
        <xs:choice maxOccurs="unbounded">
                <xs:element ref="Powerrail"/>
                <xs:element ref="NameCon"/>
                <xs:element ref="IdentCon"/>
                <xs:element ref="Openbranch"/>
                <xs:element ref="OpenCon"/>
        </xs:choice>
        <xs:attribute name="UId" type="xs:int" use="required"/>
        <xs:attribute name="Name" type="SimaticName_TP"/>
</xs:complexType>
<xs:element name="Wires" type="Wires_T"/>
<xs:complexType name="Wires_T">
        <xs:sequence maxOccurs="unbounded">
                <xs:element ref="Wire"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="TemplateReference_T">
        <xs:attribute name="Name" type="SimaticName_TP" use="required"/>
        <xs:attribute name="StartsWith" type="xs:int"/>
</xs:complexType>
<xs:element name="TemplateReference" type="TemplateReference_T"/>
```

```
</xs:schema>
```

1.1.8 SW.PlcBlocks.STL.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:include schemaLocation="SW.PlcBlocks.CompileUnitCommon.xsd"/>
	<xs:element name="StlStatement" type="StlStatement_T"/>
	<xs:complexType name="StlStatement_T">
		<xs:sequence minOccurs="0">
			<xs:group ref="Comment_G" minOccurs="0"/>
			<xs:element ref="LabelDeclaration" minOccurs="0"/>
			<xs:sequence>
				<xs:element ref="StlToken">
					<xs:annotation>
						<xs:documentation>missing for empty lines</xs:documentation>
					</xs:annotation>
				</xs:element>
				<xs:group ref="Comment_G" minOccurs="0"/>
			</xs:sequence>
			<xs:element ref="Access" minOccurs="0"/>
		</xs:sequence>
		<xs:attribute name="UId" type="xs:int">
			<xs:annotation>
				<xs:documentation>Not allowed in STL</xs:documentation>
			</xs:annotation>
		</xs:attribute>
```

```
        </xs:complexType>
        <xs:element name="StlToken" type="StlToken_T"/>
        <xs:complexType name="StlToken_T">
            <xs:sequence>
                <xs:element ref="IntegerAttribute" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>for NumBLs. NumBLs is informative</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:sequence minOccurs="0">
                    <xs:group ref="Comment_G" minOccurs="0"/>
                    <xs:element ref="Token">
                        <xs:annotation>
                            <xs:documentation>e.g 0 1 for NOP 0, NOP 1; STW for L STW or DILG for L DILG; only if separated by comment</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:sequence>
            <xs:attribute name="UId" type="xs:int" use="optional">
                <xs:annotation>
                    <xs:documentation>Not allowed in STL</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="Text" type="STL_TE" use="required"/>
            <!--<xs:attribute name="NumBLs" type="xs:int" default="0"/>-->
        </xs:complexType>
```

```
<xs:element name="StatementList" type="StatementList_T"/>
<xs:complexType name="StatementList_T">
<xs:sequence>
<xs:element ref="StlStatement" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="STL_TE">
<xs:restriction base="xs:string">
<xs:enumeration value="A"/>
<xs:enumeration value="AN"/>
<xs:enumeration value="O"/>
<xs:enumeration value="ON"/>
<xs:enumeration value="X"/>
<xs:enumeration value="XN"/>
<xs:enumeration value="S"/>
<xs:enumeration value="R"/>
<xs:enumeration value="Assign"/>
<xs:enumeration value="Rise"/>
<xs:enumeration value="Fall"/>
<xs:enumeration value="L"/>
<xs:enumeration value="T"/>
<xs:enumeration value="LAR1"/>
<xs:enumeration value="LAR2"/>
<xs:enumeration value="TAR1"/>
<xs:enumeration value="TAR2"/>
<xs:enumeration value="Extend">
<xs:annotation>
<xs:documentation>SE, SV</xs:documentation>
```

```
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="Free"/>
        <xs:enumeration value="LC"/>
        <xs:enumeration value="OffDelay">
            <xs:annotation>
                <xs:documentation>SF, SA</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="Retentive">
            <xs:annotation>
                <xs:documentation>SS</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="OnDelay">
            <xs:annotation>
                <xs:documentation>SD, SE</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="Pulse">
            <xs:annotation>
                <xs:documentation>SP, SI</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="CD"/>
        <xs:enumeration value="CU"/>
        <xs:enumeration value="CALL"/>
        <xs:enumeration value="CC"/>
        <xs:enumeration value="UC"/>
```

```
<xs:enumeration value="OPEN_DB">
        <xs:annotation>
                <xs:documentation>AUF</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="OPEN_DI">
        <xs:annotation>
                <xs:documentation>AUF DI</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="LT_I"/>
<xs:enumeration value="LT_R"/>
<xs:enumeration value="LT_D"/>
<xs:enumeration value="LE_I"/>
<xs:enumeration value="LE_R"/>
<xs:enumeration value="LE_D"/>
<xs:enumeration value="EQ_I"/>
<xs:enumeration value="EQ_R"/>
<xs:enumeration value="EQ_D"/>
<xs:enumeration value="GE_I"/>
<xs:enumeration value="GE_R"/>
<xs:enumeration value="GE_D"/>
<xs:enumeration value="GT_I"/>
<xs:enumeration value="GT_R"/>
<xs:enumeration value="GT_D"/>
<xs:enumeration value="NE_I"/>
<xs:enumeration value="NE_R"/>
<xs:enumeration value="NE_D"/>
<xs:enumeration value="JU">
```

```
            <xs:annotation>
                <xs:documentation>SPA</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="JC">
            <xs:annotation>
                <xs:documentation>SPB</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="JO">
            <xs:annotation>
                <xs:documentation>SPO</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="JZ">
            <xs:annotation>
                <xs:documentation>SPZ</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="JP">
            <xs:annotation>
                <xs:documentation>SPP</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="JM">
            <xs:annotation>
                <xs:documentation>SPM</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
```

```
<xs:enumeration value="JN">
        <xs:annotation>
                <xs:documentation>SPN</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JCN">
        <xs:annotation>
                <xs:documentation>SPBN</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JCB">
        <xs:annotation>
                <xs:documentation>SPBB</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JNB">
        <xs:annotation>
                <xs:documentation>SPBNB</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JBI">
        <xs:annotation>
                <xs:documentation>SPBI</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JNBI">
        <xs:annotation>
                <xs:documentation>SPBNI</xs:documentation>
        </xs:annotation>
```

```
</xs:enumeration>
<xs:enumeration value="JOS">
        <xs:annotation>
                <xs:documentation>SPS</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JUN">
        <xs:annotation>
                <xs:documentation>SPU</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JMZ">
        <xs:annotation>
                <xs:documentation>SPMZ</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="JPZ">
        <xs:annotation>
                <xs:documentation>SPZ</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="LOOP"/>
<xs:enumeration value="JL"/>
<xs:enumeration value="ADD"/>
<xs:enumeration value="SLD"/>
<xs:enumeration value="SLW"/>
<xs:enumeration value="SRD"/>
<xs:enumeration value="SRW"/>
<xs:enumeration value="SRSD">
```

```
                        <xs:annotation>
                            <xs:documentation>SSD,
SVD</xs:documentation>
                        </xs:annotation>
                    </xs:enumeration>
                    <xs:enumeration value="SRSW">
                        <xs:annotation>
                            <xs:documentation>SSW,
SVW</xs:documentation>
                        </xs:annotation>
                    </xs:enumeration>
                    <xs:enumeration value="RLD"/>
                    <xs:enumeration value="RRD"/>
                    <xs:enumeration value="BLD"/>
                    <xs:enumeration value="ADDAR1"/>
                    <xs:enumeration value="ADDAR2"/>
                    <xs:enumeration value="INC"/>
                    <xs:enumeration value="DEC"/>
                    <xs:enumeration value="AW"/>
                    <xs:enumeration value="OW"/>
                    <xs:enumeration value="XW"/>
                    <xs:enumeration value="AD"/>
                    <xs:enumeration value="OD"/>
                    <xs:enumeration value="XD"/>
                    <xs:enumeration value="A_BRACK"/>
                    <xs:enumeration value="AN_BRACK"/>
                    <xs:enumeration value="O_BRACK"/>
                    <xs:enumeration value="ON_BRACK"/>
                    <xs:enumeration value="X_BRACK"/>
```

```
                        <xs:enumeration value="XN_BRACK"/>
                        <xs:enumeration value="INV_I">
                                <xs:annotation>
                                        <xs:documentation>KEW,
INV_F</xs:documentation>
                                </xs:annotation>
                        </xs:enumeration>
                        <xs:enumeration value="NEG_I">
                                <xs:annotation>
                                        <xs:documentation>KZW,
NEG_F</xs:documentation>
                                </xs:annotation>
                        </xs:enumeration>
                        <xs:enumeration value="INV_D">
                                <xs:annotation>
                                        <xs:documentation>KED</xs:documentation>
                                </xs:annotation>
                        </xs:enumeration>
                        <xs:enumeration value="NEG_D">
                                <xs:annotation>
                                        <xs:documentation>KZD</xs:documentation>
                                </xs:annotation>
                        </xs:enumeration>
                        <xs:enumeration value="NEG_R">
                                <xs:annotation>
                                        <xs:documentation>NEG_G,
ND</xs:documentation>
                                </xs:annotation>
                        </xs:enumeration>
                        <xs:enumeration value="ABS_R">
```

```
            <xs:annotation>
                <xs:documentation>ABS_G</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="SQRT"/>
        <xs:enumeration value="SQR"/>
        <xs:enumeration value="LN"/>
        <xs:enumeration value="EXP"/>
        <xs:enumeration value="SIN"/>
        <xs:enumeration value="ASIN"/>
        <xs:enumeration value="COS"/>
        <xs:enumeration value="ACOS"/>
        <xs:enumeration value="TAN"/>
        <xs:enumeration value="ATAN"/>
        <xs:enumeration value="RLDA"/>
        <xs:enumeration value="RRDA"/>
        <xs:enumeration value="BTI">
            <xs:annotation>
                <xs:documentation>DEF</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="ITB">
            <xs:annotation>
                <xs:documentation>DUF</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="BTD">
            <xs:annotation>
                <xs:documentation>DED</xs:documentation>
```

```
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="DTB">
        <xs:annotation>
            <xs:documentation>DUD</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="DTR">
        <xs:annotation>
            <xs:documentation>FDG</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="RND">
        <xs:annotation>
            <xs:documentation>GFDN</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="RND_M">
        <xs:annotation>
            <xs:documentation>GFDM</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="RND_P">
        <xs:annotation>
            <xs:documentation>GFDP</xs:documentation>
        </xs:annotation>
    </xs:enumeration>
    <xs:enumeration value="TRUNC"/>
    <xs:enumeration value="ITD">
```

```
            <xs:annotation>
                <xs:documentation>FD</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="CAW">
            <xs:annotation>
                <xs:documentation>TAW</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="CAD">
            <xs:annotation>
                <xs:documentation>TAD</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="TAR1_ACCU1"/>
        <xs:enumeration value="TAR2_ACCU1"/>
        <xs:enumeration value="ADD_I">
            <xs:annotation>
                <xs:documentation>+F</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="SUB_I">
            <xs:annotation>
                <xs:documentation>-F</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="MUL_I">
            <xs:annotation>
                <xs:documentation>xF</xs:documentation>
```

```
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="DIV_I">
            <xs:annotation>
                <xs:documentation>:F</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="ADD_D">
            <xs:annotation>
                <xs:documentation>+D</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="SUB_D">
            <xs:annotation>
                <xs:documentation>-D</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="MUL_D">
            <xs:annotation>
                <xs:documentation>xD</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="DIV_D">
            <xs:annotation>
                <xs:documentation>:D</xs:documentation>
            </xs:annotation>
        </xs:enumeration>
        <xs:enumeration value="MOD_D"/>
        <xs:enumeration value="L_DBLG"/>
```

```
<xs:enumeration value="L_DILG"/>
<xs:enumeration value="L_DBNO"/>
<xs:enumeration value="L_DINO"/>
<xs:enumeration value="ADD_R">
        <xs:annotation>
                <xs:documentation>+G</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="SUB_R">
        <xs:annotation>
                <xs:documentation>-G</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="MUL_R">
        <xs:annotation>
                <xs:documentation>xG</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="DIV_R">
        <xs:annotation>
                <xs:documentation>:G</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="CAC">
        <xs:annotation>
                <xs:documentation>TAK</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="LEAVE"/>
```

```
<xs:enumeration value="PUSH"/>
<xs:enumeration value="POP"/>
<xs:enumeration value="SET"/>
<xs:enumeration value="NEG"/>
<xs:enumeration value="CLR"/>
<xs:enumeration value="BEC">
        <xs:annotation>
                <xs:documentation>BEB</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="BRACKET">
        <xs:annotation>
                <xs:documentation>)</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="SAVE"/>
<xs:enumeration value="NOP_0"/>
<xs:enumeration value="NOP_1"/>
<xs:enumeration value="MCR_BRACK">
        <xs:annotation>
                <xs:documentation>MCR(</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="BRACK_MCR">
        <xs:annotation>
                <xs:documentation>MCR)</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="MCRA"/>
```

```
<xs:enumeration value="MCRD"/>
<xs:enumeration value="ENT"/>
<xs:enumeration value="LAR1_ACCU1"/>
<xs:enumeration value="LAR1_AR2"/>
<xs:enumeration value="LAR2_ACCU1"/>
<xs:enumeration value="TAR1_AR2"/>
<xs:enumeration value="CAR">
        <xs:annotation>
                <xs:documentation>TAR</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="CDB">
        <xs:annotation>
                <xs:documentation>TDB</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="COMMENT"/>
<xs:enumeration value="EMPTY_LINE"/>
<xs:enumeration value="PSEUDO"/>
<xs:enumeration value="MOVE"/>
<xs:enumeration value="MOVE_BLOCK"/>
<xs:enumeration value="L_STW">
        <xs:annotation>
                <xs:documentation>LCC</xs:documentation>
        </xs:annotation>
</xs:enumeration>
<xs:enumeration value="T_STW">
        <xs:annotation>
                <xs:documentation>TCC</xs:documentation>
```

```
                </xs:annotation>
            </xs:enumeration>
            <xs:enumeration value="BE">
                <xs:annotation>
                    <xs:documentation>BEA</xs:documentation>
                </xs:annotation>
            </xs:enumeration>
            <xs:enumeration value="BEU"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

1.1.9 SW.PlcBlocks.TypeSupervisions.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright © Siemens AG 2008-2016. All rights reserved. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:include schemaLocation="SW.Common.xsd"/>
    <xs:element name="Type">
        <xs:simpleType>
            <xs:restriction base="xs:string">
                <xs:enumeration value="Action"/>
                <xs:enumeration value="Interlock"/>
                <xs:enumeration value="Operand"/>
                <xs:enumeration value="Position"/>
                <xs:enumeration value="Reaction"/>
                <xs:enumeration value="MessageText"/>
                <xs:enumeration value="MessageError"/>
            </xs:restriction>
```

```
        </xs:simpleType>
</xs:element>
<xs:element name="TriggeringStatus" type="xs:boolean"/>
<xs:element name="SupervisedStatus" type="xs:boolean"/>
<xs:element name="SupervisedOperand">
        <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required"/>
        </xs:complexType>
</xs:element>
<xs:element name="SubCategory2Number" type="xs:int"/>
<xs:element name="SubCategory1Number" type="xs:int"/>
<xs:element name="Number" type="xs:int"/>
<xs:element name="DelayOperand">
        <xs:complexType>
                <xs:attribute name="Name" type="xs:string" use="required"/>
        </xs:complexType>
</xs:element>
<xs:element name="Conditions">
        <xs:complexType>
                <xs:sequence>
                        <xs:element ref="Condition" maxOccurs="3"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="ConditionOperand">
        <xs:complexType>
                <xs:attribute name="Number" use="required">
                        <xs:simpleType>
                                <xs:restriction base="xs:int">
```

```
                                        <xs:enumeration value="1"/>
                                        <xs:enumeration value="2"/>
                                        <xs:enumeration value="3"/>
                                    </xs:restriction>
                                </xs:simpleType>
                            </xs:attribute>
                            <xs:attribute name="Name" type="xs:string" use="required"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="Condition">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="ConditionOperand"/>
                                <xs:element ref="TriggeringStatus"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="CategoryNumber">
                        <xs:simpleType>
                            <xs:restriction base="xs:int">
                                <xs:minInclusive value="1"/>
                                <xs:maxInclusive value="8"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="BlockTypeSupervisions">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="BlockTypeSupervisionsType"/>
```

```
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="BlockTypeSupervision">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="SupervisedOperand"/>
                    <xs:element ref="SupervisedStatus"/>
                    <xs:element ref="DelayOperand" minOccurs="0"/>
                    <xs:element ref="Conditions" minOccurs="0"/>
                    <xs:element ref="CategoryNumber"/>
                    <xs:element ref="SubCategory1Number" minOccurs="0"/>
                    <xs:element ref="SubCategory2Number" minOccurs="0"/>
                    <xs:element ref="SpecificField" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="Number" type="xs:int" use="required"/>
                <xs:attribute name="Type" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Action"/>
                            <xs:enumeration value="Interlock"/>
                            <xs:enumeration value="Operand"/>
                            <xs:enumeration value="Position"/>
                            <xs:enumeration value="Reaction"/>
                            <xs:enumeration value="MessageText"/>
                            <xs:enumeration value="MessageError"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
```

```
            </xs:complexType>
        </xs:element>
        <xs:element name="AssociatedValues">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="AssociatedValue" maxOccurs="3"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="AssociatedValue">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="AssociatedValueOperand"
minOccurs="0"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="SpecificFieldText">
            <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                    <xs:element ref="MultiLanguageText"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="SpecificField">
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="AssociatedValues" minOccurs="0"/>
                    <xs:element ref="SpecificFieldText" minOccurs="0"/>
```

```
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:complexType name="BlockTypeSupervisionsType">
            <xs:sequence>
                <xs:element ref="BlockTypeSupervision" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="AssociatedValueOperand">
            <xs:complexType>
                <xs:attribute name="Number" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:int">
                            <xs:enumeration value="1"/>
                            <xs:enumeration value="2"/>
                            <xs:enumeration value="3"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Name" type="xs:string" use="required"/>
            </xs:complexType>
        </xs:element>
</xs:schema>
```