UNIVERSIDAD
DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

Tesis Doctoral

MARCO DE TRABAJO PARA LA GENERACIÓN DE SOFTWARE PARA LA GESTIÓN DE SISTEMAS DE ENERGÍA SOLAR

Autoría:
Ildefonso Martínez Marchena

Dirección:
Llanos Mora López

Mayo, 2015

La Dra. Llanos Mora López, Titular de Universidad en el Departamento de Lenguajes y Ciencias de la Computación de la Universidad de Málaga

CERTIFICA

Que Ildefonso Martínez Marchena, Ingeniero en Informática, ha realizado bajo su dirección la tesis doctoral titulada MARCO DE TRABAJO PARA LA GENERACIÓN DE SOFTWARE PARA LA GESTIÓN DE SISTEMAS DE ENERGÍA SOLAR, que se recoge en la presente memoria, cumpliendo todos los requisitos legales para optar al grado de Doctor, por lo que autoriza su lectura y defensa pública.

Y para que así conste y tenga los efectos oportunos, firmo este certificado en

Málaga, a 10 de mayo de 2015

Dra. Llanos Mora López

A Laura y Encarni

Agradecimientos

Durante el desarrollo de este trabajo he tenido la gran suerte de estar rodeado de grandes personas que, gracias a su apoyo, sacrificio, esfuerzo y ayuda han permitido que esta tesis vea la luz. Me gustaría que estas primeras líneas fuesen de un agradecimiento especial a Encarni, mi mujer, por su continuo ánimo y apoyo haciéndose cargo de muchas de mis responsabilidades personales para poder yo dedicarle tiempo al desarrollo de este trabajo. Sin ella esta tesis no sería posible ya que siempre ha estado ahí para ofrecerme su mano a la que agarrarme en los momentos en los que he necesitado levantarme.

A mi hija Laura, que espero algún día pueda devolverle todas las horas que no le he podido dedicar, por sus continuas visitas a los pies de mi escritorio buscando juego cada vez que veía que pasaba varias horas sin salir del despacho.

A mis padres, de los cuales he aprendido que el esfuerzo y la dedicación son las principales obligaciones para con su trabajo que un hombre debe tener y poder llegar a sentirse orgulloso de si mismo.

Y a mi directora de tesis, Llanos Mora y a Mariano Sidrach, por su paciencia, total disponibilidad y sugerencias en todas las cuestiones técnicas y teóricas de este trabajo, por guiarme con su gran conocimiento y experiencia, y de cuyas facetas de científicos y brillantes investigadores no hacen más que impresionarme día a día, pero más aún, si cabe, por sus valores y calidad humana donde han sabido demostrar y dejar huella en todo lo que han hecho a lo largo de sus carreras personales y profesionales.

A todos, gracias.

Parte de este trabajo de investigación ha sido financiado por la Junta de Andalucía (proyecto No. TICP–6441)

Resumen

El auge que están alcanzando los sectores relacionados con la gestión de sistemas energéticos y en especial, los relacionados con las energías renovables, están propiciando la necesidad de disponer de mecanismos para la monitorización y la vigilancia energética de los mismos. El objetivo de esta tesis es la de proponer y desarrollar un marco de trabajo, integrado con una metodología y procedimientos, para generar programas para la monitorización y supervisión de sistemas de energía solar.

Para el desarrollo de este marco de trabajo se ha propuesto una arquitectura software basada en capas, cada una de las cuales incluye componentes desarrollados a partir de interfaces especializados que ofrecen funcionalidades para la definición, modelizado, comunicación, almacenamiento, supervisión, evaluación y predicción del funcionamiento de sistemas energéticos. Esta arquitectura permite describir las partes de un sistema energético y la interacción entre ellas, así como los patrones que se utilizan para su composición y las restricciones de estos patrones. Otra aportación de esta tesis es la definición de los estándares utilizados para el establecimiento de conexiones y coordinación entre los componentes de cada capa.

La propuesta que se hace de utilizar un marco de trabajo para desarrollar software para la gestión de sistemas energéticos facilita la construcción de nuevo software de gestión, evita los detalles de bajo nivel y minimiza los tiempos de desarrollo. El marco de trabajo puede extenderse y seguir un esquema desacoplado al no existir relaciones directas entre clases a favor de las relaciones establecidas entre interfaces. Mecanismos como la inyección de dependencias o la implementación por delegación de interfaces hacen posible que este marco de trabajo desarrollado sea altamente desacoplado y fácilmente extensible y configurable.

Otra importante aportación de la tesis es la propuesta de un sistema de comunicaciones que permite mecanismos de intercambio de información utilizando estándares abiertos. Uno de los grandes problemas en el desarrollo de sistemas de monitorización y supervisión de sistemas energéticos es la diversidad de dispositivos y protocolos existentes. Para resolver este problema, el sistema de comunicaciones propuesto está totalmente separado de las demás funcionalidades y se basa en la utilización del estándar OPC. Esto permite disponer de un mecanismo único, sencillo y eficiente de comunicación con todos los dispositivos con los que se debe intercambiar información que forman parte de las instalaciones. El marco de trabajo presentado proporciona además,

un conjunto de clases que puede ser directamente utilizado en la gestión de sistemas energéticos para la comunicación con la mayor parte de los dispositivos disponibles.

Para el desarrollo e implementación del marco de trabajo se han utilizado metodologías ágiles lo que ha permitido poner de manifiesto buenas prácticas para el desarrollo de sistemas software que garantizan una calidad en el resultado final en base a conceptos de rendimiento, mantenimiento y extensibilidad.

Por otra parte, se han propuesto modelos de evaluación del funcionamiento de sistemas de energía solar basados en estadística descriptiva e inferencial y modelos de predicción de la producción de estos sistemas basados en técnicas de aprendizaje automático. Además, se ha propuesto también un modelizado genérico para la representación de las instalaciones energéticas que permite utilizar los modelos de evaluación y predicción propuestos.

Finalmente, se han desarrollado varios programas de monitorización y supervisión para sistemas reales que han permitido comprobar la capacidad del marco desarrollado para la gestión de sistemas energéticos reales. Se presentan, a modo de ejemplo de la validez del marco de trabajo desarrollado, los resultados obtenidos para algunas de estas instalaciones.

Índice general

Índice de figuras

Índice de Tablas

"Hay tres maneras de adquirir sabiduría: primero, por la reflexión, que es la más noble; segundo, por imitación, que es la más sencilla; y tercero, por la experiencia, que es la más amarga"

– Confucio

1

Introducción y objetivos

1.1. Introducción

El importante desarrollo que están alcanzando las tecnologías que permiten el intercambio de datos e información a través de la red está facilitando el desarrollo de programas y servicios en muchos ámbitos distintos, ya que se posibilita un acceso a la información desde cualquier lugar y ya actualmente, desde diversos y distintos dispositivos. Uno de los sectores en los que se está haciendo un uso importante de estas tecnologías es el relacionado con la gestión y supervisión de sistemas energéticos y más concretamente en los sistemas de energía solar, y entre estos, los de energía solar fotovoltaica.

El número de instalaciones de energía solar de este tipo ha experimentado un importante crecimiento en los últimos años debido a muchos factores, entre ellos: a la necesidad de utilizar fuentes energéticas que contribuyan a la reducción de las emisiones de carbono, al establecimiento de políticas de apoyo para la implantación de este tipo de sistemas, a la mejora de la eficiencia de estos sistemas y a la importante reducción en el precio de todos los componentes que los integran. De acuerdo con los datos publicados en el "Global Market Outlook for Photovoltaics 2014-2018", [Ass14] en el año 2013 se instalaron más de 38.4GW de sistemas fotovoltaicos.

Conforme aumenta el número de instalaciones de energía solar, surge cada vez más, la necesidad de disponer de soluciones que permitan una gestión de las mismas de cara a poder lograr un mejor funcionamiento de sus componentes. Entre las tareas a las que debería dar respuesta este software se pueden citar: modelizado de la instalación

energética recogiendo toda la configuración física y lógica de la misma, recogida de datos de los dispositivos que la conforman (monitorización), análisis y evaluación del funcionamiento de cada instalación (incluidos los diferentes subsistemas), generación de informes de funcionamiento y producción, publicación de los datos orientados a multicanales y multidispositivos, gestión de alarmas y eventos y, en el caso de determinado tipos de sistemas conectados a red, previsión de la energía que la planta producirá el día siguiente, en una escala horaria.

En el caso de sistemas energéticos convencionales todas estas tareas se viene realizando mediante los sistemas conocidos como sistemas SCADA (Supervisory Control and Data Acquisition). El tamaño de estos sistemas energéticos justifica la utilización de este tipo de aplicaciones, ya que el coste asociado a la gestión y mantenimiento es totalmente asumible dentro de los costes del sistema en general. En los últimos años, están surgiendo sin embargo, gran cantidad de instalaciones energéticas basadas en la utilización de fuentes de energías renovables, cuyo tamaño y coste no justifica la utilización de SCADAS por lo que la existencia de alternativas más apropiadas para este tipo de sistemas son altamente deseables.

En general, estos sistemas energéticos, especialmente los de pequeña y mediana potencia, se ubican en emplazamientos dispersos, y no suelen disponer de sistemas propios de monitorización, ya que no hay personal en las plantas que pueda hacer esa vigilancia in situ. Sin embargo, poder disponer de información del funcionamiento de los sistemas energéticos es muy importante para poder intervenir en caso de que se detecten problemas o fallos, tal y como se apunta en [SWKL97].

Tradicionalmente, para monitorizar sistemas de energéticos de pequeño y medio tamaño, los fabricantes de los dispositivos que forman parte de la instalación, suministran su propio software con las funcionalidades mínimas para su mantenimiento básico. Estos programas suelen estar instalados en las propias plantas por lo que se hace necesaria una supervisión personal y desde dichas ubicaciones.

Con este planteamiento tradicional, han aparecido en los últimos años varios problemas, a saber:

1. Utilización de distintos programas para recuperar la información de las plantas, dependiendo, en general, del software desarrollado por los fabricantes de alguno de los subsistemas (normalmente de los inversores).

2. Necesidad de contar con especialistas en el sistema que se está monitorizando para poder hacer una evaluación correcta del funcionamiento del mismo.

3. Imposibilidad de integrar la información de distintas plantas en una sola herramienta o incluso, en algunos casos, no poder integrar la información proveniente de todos los dispositivos del sistema al tener que utilizar un software diferente para cada uno.

4. Necesidad de modificar los programas de monitorización cuando se modifica algún

componente de la instalación.

5. A veces, la necesidad de desplazarse a cada instalación para tener acceso a la información.

6. Los fabricantes de los subsistemas que incluyen la toma de datos no suelen facilitar controladores eficientes de comunicación.

7. Cada fabricante desarrolla su propio protocolo de intercambio de información.

Por otra parte, esta supervisión, para que sea útil, debe hacerse por parte de técnicos especializados en este tipo de sistemas, con el objeto de poder detectar y diagnosticar problemas de funcionamiento de las plantas ya que el propósito de este tipo de sistemas suele ser exclusivamente la recogida de datos y no la evaluación del funcionamiento. Al depender fuertemente el funcionamiento de una instalación de características propias de la misma, se hace difícil incorporar en los programas de monitorización, modelos de evaluación que sean generales y válidos para todos los sistemas.

De manera general, se puede afirmar que los programas desarrollados para la gestión de sistemas energéticos deben tener en cuenta la complejidad y características de los mismos, entre otras funcionalidades deben ser capaces de dar respuesta a las cuestiones que se recogen en el trabajo de [WKS+01] y que para el caso de sistemas energéticos se pueden sintetizar de la siguiente forma:

1. Adaptabilidad: capacidad de reconfigurar fácilmente las aplicaciones cuando se cambie la configuración del sistema energético. Esto es muy importante que ya en los sistemas energéticos, durante su periodo de funcionamiento, pueden producirse cambios en los dispositivos que los integran, como pueden ser sustitución de algún componente incluso de otra marca o fabricante.

2. Extensibilidad: posibilidad de incorporar nuevos algoritmos y modelos de gestión sin necesidad de volver a diseñar los componentes que hay ya en el sistema.

3. Interoperabilidad: posibilidad de operar con distintas plataformas hardware y, especialmente, distintos protocolos de red. Es también importante que por las características del emplazamiento disperso de los sistemas energéticos (especialmente los de baja potencia) esté prevista la utilización de redes inalámbricas o diferente tipo de sistema de comunicación en cada caso.

4. Arquitectura abierta: se debe permitir la configuración y el intercambio de componentes, es decir se deben desarrollar arquitecturas software que sean flexibles y soporten herramientas y algoritmos de distintas fuentes y dominios.

5. Arquitectura moderna: se espera de los sistemas software que proporcionen el mayor número de funcionalidades y además hagan uso de la tecnología cada vez más avanzada en este ámbito.

Además de estos requisitos generales, el dominio de la aplicación que se propone en esta tesis hace que se pueda dar respuesta también a estos requisitos específicos:

- Acceso remoto a la información almacenada en los distintos dispositivos de las plantas para poder evaluar el funcionamiento de las mismas. Es decir, posibilidad de conectarse a los dispositivos de la instalación para obtener los datos registrados. Esta conexión debería poder hacerse utilizando distintos protocolos y sistemas.
- Integración de dispositivos de diferentes marcas y modelos y de distintos fabricantes.
- Capacidad para almacenar todos los datos de la planta, no solo los registrados sino también aquellos parámetros calculados y toda la información general de la planta y sus dispositivos.
- Posibilidad de procesar la información para evaluar el funcionamiento de la planta y para detectar posibles problemas utilizando diferentes modelos en función de la configuración de cada planta.
- Posibilidad de ejecutar tareas programadas para automatizar las tareas de recuperación de la información, análisis y evaluación.
- Facilidades para disponer de la información general y de la información crítica como alarmas y eventos a través de distintos medios.
- Incorporación de mecanismos que permitan envíos de alertas al gestor o propietarios de la plantas.
- Información accesible desde distintos canales a diferentes servicios y utilizando diversos protocolos.

Unos de los problemas más frecuentes e importantes en el dominio de la gestión energética, es que las herramientas clásicas desarrolladas para la monitorización y control de este tipo de sistemas no suelen permitir la conectividad de diferentes sistemas y aplicaciones por la ausencia de interfaces estándares de intercambio de información. Esto es así, especialmente, en las aplicaciones que se han desarrollado para sistemas energéticos de pequeña y media potencia. Sin embargo, dado el importante incremento que este tipo de sistemas está experimentando en los últimos años, y la variedad de soluciones que han aparecido, se constata que sería muy conveniente poder contar con componentes estándar que permitieran la conectividad de distintos sistemas de manera más sencilla.

Los sistemas tipo SCADA (Supervisory Control and Data Acquisition) que se utilizan en la monitorización y gestión de plantas de energía de pequeña y, especialmente, media potencia están normalmente basados en desarrollo muy genéricos y poco orientados al dominio de la energía fotovoltaica. Además, este tipo de soluciones son siempre

propietarias y no permiten conectividad con otro tipo de soluciones, por lo que cuando se opta por una de estos paquetes, es difícil, si no imposible, cambiar a otras soluciones. Así, cuando son necesarios cambios en los sistemas desarrollados, es necesario recurrir al mismo tipo de sistema con el que se desarrolló el programa. Esto implica otra serie de problemas adicionales, como son:

- Es necesario utilizar programas diferentes para recuperar la información de estos sistemas, dependiendo del software desarrollado por cada fabricante. Normalmente, no es sencillo conectar las aplicaciones y datos de sistemas diferentes. Es decir, es difícil integrar la información de distintas instalaciones.

- El software de supervisión debe actualizarse si se modifica alguno de los subsistemas de la planta.

- A veces, la información solo es accesible de manera local, especialmente para pequeñas instalaciones.

- Cada fabricante desarrolla su propio protocolo de comunicaciones, en muchos casos propietarios, no estandarizados y desconocidos, y a veces, los controladores de comunicación suministrados por los fabricantes no son eficientes.

La falta de estándares en el desarrollo de las interfaces de nivel de aplicación hace que sea difícil interconectar las diferentes aplicaciones desarrolladas por diferentes fabricantes. Esto no es un problema específico de los sistemas de monitorización y control. La necesidad de utilizar estándares en el proceso de desarrollo de software ya ha sido considerada en muchos otros ámbitos de los sistemas industriales de control y monitorización, y se ha puesto de manifiesto desde hace tiempo, [Ple94], [God97], [DO03].

En el caso de los sistemas de supervisión y gestión para instalaciones energéticas, se pueden destacar las siguientes soluciones propuestas:

- Una de las primeras propuestas fue hecha por [BM98]. Presentan un sistema de adquisición de datos para la monitorización del funcionamiento de sistemas fotovoltaicos. Proponen la utilización de una arquitectura basada en el microprocesador MC68B09. La adquisición de los datos se controla a partir de un temporizador programable. El microsistema permite el tratamiento de los datos en tiempo real. Los datos se transfieren al ordenador a través de un puerto serie bidireccional usando el protocolo RS232C. El sistema propuesto es responsable de la adquisición secuencial y el tratamiento preliminar de los datos y de la transferencia diaria de los mismos al ordenador central. El sistema tiene algunas limitaciones relacionadas con la capacidad de memoria. Para cada instalación es necesario instalar un microsistema y un módem para la transferencia de datos al ordenador central.

- [KKV03] proponen la utilización de una unidad de adquisición de datos local para registrar todas las medidas del sistema. Los datos que se adquieren son transmitidos a una unidad central por medio de un enlace RF y un puerto serie. El problema de este sistema es que se incrementa considerablemente el coste de la instalación, especialmente para plantas energéticas pequeñas.

- En el trabajo de [WL07] se propone la utilización de un PLC industrial con acceso a Internet para la monitorización y control de un sistema híbrido (eólico, fotovoltaico y con acumulación de energía basada en baterías). El PLC conecta con un ordenador remoto. El software incluye algunas funciones útiles para monitorizar y controlar este tipo de instalaciones, por la utilización del PLC. El sistema desarrollado no puede considerarse genérico ya que está desarrollado para una configuración concreta de instalación.

- [GLP07] proponen un sistema basado en la utilización de un DAS (Data Acquisition System) que se instala en la planta y es el responsable de registrar todos los parámetros del sistema. El DAS está basado en la arquitectura del NI Field-Point. Se utiliza LabVIEW en tiempo real. El software se ha desarrollado utilizando LabVIEW Virtual Instrument (VI). Cuenta también con un módulo para la gestión de las comunicaciones. El principal problema de esta solución es que es necesario instalar un DAS en cada planta y que se utiliza software propietario.

- [SPG+08] proponen la utilización de un registrador de datos o datalogger especializado para la supervisión tanto de la estación meteorológica como los subsistemas de la planta (inversores). Este DT está conectado a un terminal y enlazado a Internet a través de LAN. El problema de su utilización es que el DT (Sunny Boy Controller Plus) está diseñado para trabajar solo con un tipo de subsistemas (inversor SWR700).

- [GTCS09] proponen una arquitectura basada en la utilización de unidades terminal remotas (RTU) para conectar con un SCADA central. De acuerdo con estos autores, la utilización de este tipo de sistemas está justificada solo en el caso de sistemas energéticos de gran tamaño debido a la inversión que suponen.

Los principales problemas de las diferentes soluciones analizadas para instalaciones energéticas de pequeña y mediana potencia son tanto el coste de los sistemas como la necesidad de desarrollar un sistema específico dependiendo de los componentes que componen la instalación y de las características de conectividad de las que disponen.

1.2. Objetivos

Para dar respuesta a los requerimientos enumerados anteriormente así como para resolver los problemas que plantean la mayoría de los sistemas que se están implementando, en esta tesis se propone el desarrollo de un marco de trabajo compuesto por

clases e interfaces que permite la creación de aplicaciones y sistemas para el control y monitorización de sistemas energéticos genéricos. La versatilidad de este marco de trabajo se demostrará extendiéndolo de manera que permita trabajar en el dominio de las energías renovables y más concretamente en sistemas de energía solar fotovoltaica.

Unos de los objetivos perseguidos en este trabajo son permitir tanto la integración de diferentes tecnologías y protocolos de comunicación utilizados en sistemas energéticos, así como la vigilancia remota y la evaluación automática del funcionamiento, integrando componentes que permitan la monitorización, la supervisión y la integración de componentes que se encarguen de automatizar las tareas de evaluación del funcionamiento y predicción de la producción de estos sistemas.

Los objetivos planteados en este trabajo son los siguientes:

- Proponer una arquitectura software y un marco de trabajo que permitan desarrollar programas para la gestión de instalaciones energéticas en los que se integren de manera homogénea componentes de distintas tecnologías.
- Proponer una metodología de desarrollo para la construcción del marco de trabajo de manera que se garantice su funcionamiento y extensibilidad en condiciones cambiantes.
- Desarrollar un conjunto integrado de clases e interfaces que puedan ser utilizados por los desarrolladores de sistemas de adquisición de datos y de sistemas gestión energética, de manera que se facilite el desarrollo de nuevo software
- Proponer un sistema que pueda ser utilizado en entornos reales donde puedan existir condiciones y requisitos cambiantes.
- Modelizar el dominio de los sistemas de energía solar fotovoltaica y probar en este tipo de sistemas el marco de trabajo propuesto.
- Elaborar un modelo de predicción y evaluación de la producción de energía en sistemas fotovoltaicos, utilizando para ellos modelos estadísticos y de aprendizaje computacional.
- Integrar los modelos de evaluación y predicción del funcionamiento de estos sistemas en los programas desarrollados de manera que estas tareas pueden realizarse sin la necesidad de que haya personal especializado en las plantas.

1.3. Organización de la tesis

Esta tesis se encuentra organizada en 11 capítulos, incluida esta introducción. En el capítulo 2, denominado "Marcos de Trabajo, Arquitectura Software y Metodologías de

desarrollo", se hace un repaso del estado del arte de los marcos de desarrollo y de las metodologías apropiadas para una implementación exitosa de este tipo de soluciones.

En el capítulo 3, "Marco de trabajo para el desarrollo de software para la gestión de sistemas energéticos", se propone una arquitectura y un marco de diseño basado en capas, así como se detallan soluciones apropiadas y buenas prácticas para disponer de una solución robusta, flexible y de alto rendimiento haciendo uso de paradigmas de desarrollo como la inyección de dependencias y la implementación basada en interfaces.

En el capítulo 4 se aborda el problema del intercambio de información entre los dispositivos que forman parte de una instalación energética y cómo resolver los detalles de unificar los diferentes protocolos y medios de comunicación para disponer de una solución homogénea a este problema. Se muestra además una propuesta de clases y de interfaces que forman parte del marco de desarrollo planteado para disponer de soluciones para la integración de los diferentes dispositivos y la inclusión de nuevos de manera sencilla al sistema inicial.

En el capítulo 5 se presenta la parte del marco desarrollado donde se enfatiza en la importancia de disponer de una caracterización apropiada o modelizado de las instalaciones energéticas que se pretenden monitorizar. Se plantea además la importancia de dotar al marco de trabajo de mecanismos para el almacenamiento y recuperación de manera óptima de la información tanto del modelizado como del funcionamiento de las instalaciones. Es por ello que, en el capítulo 6 se presenta una solución a este problema así como los algoritmos necesarios para disponer de una coherencia apropiada de la información y los datos descargados de las instalaciones.

El capítulo 7 recoge las clases e interfaces que dan forma al marco de trabajo y se utilizan y sirven de apoyo al resto del sistema. Esta capa es sumamente importante ya que es la que define el núcleo del marco de trabajo y proporciona la potencia expresiva para la construcción del resto de capas.

El capítulo 8 introduce las necesidades reales en cuanto a la gestión y monitorización de sistemas fotovoltaicos y pone en evidencia la capacidad del marco de trabajo para la inclusión de modelos de evaluación y predicción de manera sencilla, permitiendo dotar a los sistemas basados en el marco presentado de la capacidad de incorporar modelos apropiados para cada necesidad.

En los capítulos 9 y 10, se extiende el marco y los resultados de los capítulos anteriores para cubrir el dominio de la energía solar fotovoltaica de manera que, lo que ha sido desarrollado como un marco genérico para la gestión de instalaciones energéticas, puede ser fácilmente extendido y ampliado para la gestión de instalaciones de energía solar, disponiendo además de la capacidad de añadir módulos, como los modelos de predicción y evaluación, de manera sencilla.

En el último capítulo se exponen líneas de trabajo abiertas así como las conclusiones extraídas durante el desarrollo de esta tesis.

"El software es como la entropía: difícil de atrapar, no pesa, y cumple la Segunda Ley de la Termodinámica, es decir, tiende a incrementarse"

–Norman Augustine

2

Marcos de trabajo, arquitectura software y metodologías de desarrollo

2.1. Introducción

La aparición de nuevos paradigmas para el desarrollo y la ingeniería del software permite crear sistemas cada vez más grandes y con más funcionalidades sin perder la calidad o fiabilidad como ocurría hasta ahora cuando los sistemas software crecían de manera considerable. Esto ha sido posible, entre otros factores, a la utilización de marcos de trabajo, arquitecturas software y a la utilización de nuevas metodologías de desarrollo.

En este capítulo se hace una propuesta de diseño de marco de trabajo para el ámbito de la monitorización, evaluación y gestión de los sistemas energéticos. En concreto, el dominio de aplicación en el que se utilizará la arquitectura que se propone en esta tesis es el de los sistemas energéticos genéricos y se propone la extensión de sus funcionalidades para su utilización en el ámbito de los sistemas energéticos fotovoltaicos.

De entre las primeras propuestas hechas para el control y gestión de sistemas industriales basadas en la utilización de las tecnologías de información y comunicación incipientes se pueden destacar, entre otras, la que se hace en [AOB76] para el control de procesos experimentales, el sistema de prototipado rápido para la supervisión a través de Internet de procesos industriales desarrollado por [LTC01] y la propuesta de [TCD05] para la integración de servicios Web reconfigurables en el control de la logística de transporte y almacenamiento de productos. Más recientemente, con la aparición

de los denominados *marcos de trabajo* el planteamiento de estos sistemas ha experimentado un cambio sustancial. Así, por citar solo algunos ejemplos relevantes, en el trabajo de [GH09] se presenta una propuesta para utilizar redes de sensores inalámbricos en entornos y procesos industriales, en [FCTB10] se propone un marco de trabajo para la monitorización y vigilancia autónoma de equipos industriales y en [MPSD10] se presenta una propuesta de arquitectura software para el acceso y gestión de datos en aplicaciones con fuentes de datos variables y configurables.

Una de las propuestas del trabajo que se presenta en esta memoria es, precisamente, utilizar una arquitectura software que se desarrolla a partir de marcos de trabajo y que está basada en componentes, partiendo de las propuestas generales que se hacen en [Szy98], [SLM98]; este tipo de arquitectura ha sido ampliamente utilizada en distintos dominios de otros campos de sistemas industriales, como por ejemplo, para el control autónomo de vehículos aéreos (sin pilotos) [WKS+01], en la manufacturación de sistemas [FSH01], y en algunos de los trabajos citados anteriormente. También en el ámbito de este tipo de sistemas, el desarrollo de componentes que se basen en estándares, de manera que se facilite la conectividad de los mismos, es fundamental, tal y como se ha señalado previamente en varios trabajos, [WKS+01], [FSH01].

Además es importante destacar que la propuesta que se haga debe permitir la integración de distintos tipos de componentes, que pueden ir desde drivers para sistemas hardware hasta funcionalidades que permitan la implementación de algoritmos de evaluación. De esta manera el marco de trabajo y la arquitectura que se proponen permitirán dar respuesta a todas las necesidades detectadas en la gestión de sistemas energéticos. En este sentido, será importante que la solución propuesta contemple la adaptación rápida de los componentes y aplicaciones desarrollados, para resolver de manera sencilla las posible reconfiguraciones de los sistemas, inclusión de nuevos elementos o modificación de los existentes.

En las siguientes secciones de este capítulo se describen los fundamentos de los que se parte para las propuestas que se hacen en este trabajo. Se revisa los marcos de trabajo y las arquitecturas software y se justifica la metodología de desarrollo que se ha aplicado.

2.2. Arquitecturas software y marcos de trabajo

En este trabajo se propone una arquitectura software así como una metodología de desarrollo para la construcción de un marco de trabajo para la creación de software de monitorización y gestión de sistemas energéticos. Una arquitectura software describe las partes de un sistema y la interacción entre ellas, así como los patrones que se utilizan para su composición y las restricciones que se imponen a la utilización de esos patrones.

Las arquitecturas software y marcos de trabajo se han utilizado en muchos otros dominios, entre los que se pueden citar los trabajos de [BPE+98] y [BFSS09] en el

ámbito de desarrollos de aplicaciones para seguridad; en [TVK09] se propone una plataforma basada en agentes distribuidos para la automatización de sistemas de control; [ZPLZ09] proponen un sistema de decisión para extraer las características dominantes que permiten predecir el desgaste en un sistema industrial; en [BFTH10] se propone un sistema que permite la creación de aplicaciones móviles; [DR99] proponen una arquitectura que permite el desarrollo de sistemas inteligentes para fabricación; en [JZA+06] se propone un marco de trabajo para simulaciones en el ámbito de la física; y en [GOGM13] se presenta una arquitecta software basada en componentes para la integración de plataformas de robótica industrial con multisensores.

Una de las posibles definiciones de lo que es un marco de trabajo, también conocidos como *framework*, es la que se propone en [JF88]

A framework is a set of classes that embodies an abstract design for solutions to a family related problems.

Es decir, un marco de trabajo es un conjunto de clases e interfaces que permite encapsular un diseño que puede ser utilizado para problemas y sistemas que estén relacionados. En el marco de trabajo también se especifican la forma en la que sus instancias interactúan. Un marco de trabajo es por tanto un diseño reutilizable de un sistema (total o parcialmente). De esta manera, una de las principales ventajas que se consigue con la utilización de marcos de trabajo es la reducción del coste y la reutilización del software que se desarrolla [FS97].

En un marco de trabajo se pueden definir dos niveles de abstracción, por una parte la arquitectura y por otra la implementación del marco de trabajo. A los marcos de trabajo también se les denomina *arquitectura software específica de un dominio*. Se pasa por tanto de una descripción de la arquitectura a un marco de trabajo cuando esa arquitectura se utiliza en un sistema de dominio específico.

Por tanto, un marco de trabajo también puede ser definido como un conjunto de clases e interfaces cooperativas que construyen un diseño reutilizable para un tipo específico de software. Un marco de trabajo proporciona, por una parte, la arquitectura y, por otra, la implementación. Para su diseño se utilizan clases abstractas y se definen sus responsabilidades y colaboraciones, usualmente a través de la utilización de interfaces. A partir del mismo, se puede desarrollar una aplicación haciendo subclases, implementando interfaces y componiendo instancias a partir de las clases definidas.

Desde el punto de vista del desarrollo de software, se puede sintetizar que un marco de trabajo es una estructura de soporte definida, a partir de la cuál se pueden organizar y desarrollar distintos proyectos de software.

Un marco de trabajo difiere de una librería de clases ya que:

- En una librería de clases, el control del flujo se encuentra en el código de la aplicación que realiza llamadas a los métodos de la librería de clases mientras

que en un marco de trabajo el control del flujo está en el propio código, que realiza llamadas al código de la aplicación, mecanismo comúnmente denominado *inversión del control*.

- Los marcos de trabajo son *el servicio* mientras que en una librería de clases los servicios se obtienen heredando funciones de las librerías de clases.
- En un marco de trabajo hay una mayor abstracción de los servicios ya que se oculta su complejidad y se automatizan las características estándares y se ofrece mecanismo de excepciones. En una librería de clases el usuario ha de determinar qué métodos ofrece la librería y en qué orden deben ser invocados.
- En un marco de trabajo, una vez construido, la cantidad de código a implementar suele ser menor que en una librería de clases.
- El coste de mantenimiento es menor en un marco de trabajo.
- En un marco de trabajo se reduce la complejidad, ya que el usuario tiene que escribir muy poco código mientras que en una librería de clases el usuario tiene que crear nuevas clases para cada aplicación que se desarrolle además de desarrollar la propia aplicación.

Los marcos de trabajo, en general, incluyen:

- Soporte para el desarrollo de programas.
- Bibliotecas de componentes.
- Posibilidad de incluir scripting (Lenguaje de scripting).
- Software para desarrollar y unir diferentes componentes de un proyecto de desarrollo de programas.

Una gran ventaja de abordar la construcción del software mediante marcos de trabajo es que estos permiten:

- Facilitar el desarrollo de software ya que proporcionan código que no se tendrá que reescribir.
- Evitar los detalles de bajo nivel, permitiendo concentrar más esfuerzo y tiempo en identificar los requerimientos de software.
- Minimizar los tiempos de desarrollo.
- Construir una arquitectura consistente entre aplicaciones.

Algunas de las desventajas que se pueden citar cuando se utilizan marcos de trabajo es que limitan la flexibilidad de los usuarios, en el sentido de que los componentes que este construya deben tener en cuenta las restricciones impuestas por la arquitectura en la que se basan. Esta limitación no es tan importante si el diseño de la arquitectura se plantea de manera adecuada al dominio de la aplicación.

Los marcos de trabajo pueden diseñarse y desarrollarse utilizando lo que se conoce como *patrones de diseño*. Un patrón de diseño es una arquitectura que soluciona problemas concretos dentro de la arquitectura general de un sistema. Sirven además para documentar el diseño que se realiza. En la propuesta que se presenta, se hace uso, en gran medida, de los patrones de diseño más conocidos y utilizados, [GHJV95].

En este trabajo se abordan tanto los aspectos relacionados con la arquitectura de diseño software como las propuestas concretas que permiten generar diferentes productos finales; como se describe en un capítulo posterior de la memoria, alguno de estos ya han sido probados en instalaciones energéticas reales.

Los requisitos iniciales que se le han exigido al marco de trabajo y siempre teniendo en cuenta que su aplicación debe ser directa y efectiva, han sido, tal y como se ha comentado en una sección previa:

- Sistema extensible y ampliable
- Altamente desacoplado
- De alto rendimiento

2.3. Metodologías ágiles para el desarrollo del marco de trabajo

Desde el comienzo del trabajo, se ha puesto especial cuidado en el ciclo de desarrollo del software. En cada ciclo de desarrollo se han extraido conclusiones que han sido aplicadas para mejorar sustancialmente el trabajo, apoyándose en la utilización de metodologías ágiles, que permiten una adaptación rápida y eficaz al cambio. La redefinición del trabajo realizado ha desembocado en un marco de trabajo que permite la creación de aplicaciones y herramientas de apoyo de manera sencilla, fácil y de alto rendimiento. Para poder cumplir con los requisitos exigidos al marco de trabajo, desde el punto de vista de su desarrollo, se ha realizado un estudio de las metodologías del software así como de los lenguajes de programación y entornos de desarrollo apropiados.

La utilización de lenguajes orientados a objetos, en una primera instancia, puede parecer que es lo más más apropiado, ya que permiten aplicar multitud de paradigmas para el desarrollo del marco de trabajo como son la encapsulación, polimorfismo y

utilización de patrones de diseño [GHJV95] lo que hace que se pueda disponer de un sistema ampliable, extensible. Sin embargo, para el desarrollo del marco de trabajo se planteó como requisito que se pudiera integrar en los sistemas operativos actuales con herramientas y librerías de desarrollo de terceros, por lo que la utilización de herramientas y metodologías formales no se hacen apropiados para este caso.

Una de las características más críticas que debía tener en cuenta el sistema era el rendimiento del mismo, ya que debía permitir desarrollar aplicaciones para sistemas reales donde existen dispositivos con requisitos cambiantes y de diversa naturaleza, es decir, no se pretendía disponer solo de un marco teórico sino que, como se presenta en capítulos posteriores, este marco de trabajo debía poder ser utilizado, y por lo tanto validado, en instalaciones reales.

Respecto a la selección del lenguaje de implementación del marco de trabajo se analizaron varias opciones. Una de las posibilidades que se analizaron fue la utilización de Java; sin embargo, no se obtuvieron buenos resultados debido a que la velocidad y el tratamiento a bajo nivel de dispositivos electrónicos complejos resultó ser un hándicap insalvable. La opción de utilización de C o C++ resultaba también muy apropiada debido a su alto rendimiento y la posibilidad de ofrecer interfaces de conexión con librerías y APIs de terceros pero no disponer de un paradigma de programación orientada a objeto pura al contrario que Java también hizo que se desestimara la utilización de este lenguaje. Finalmente se ha optado por utilizar Object Pascal ya que combina lo bueno de una programación a objetos pura donde aplicar metodologías de desarrollo orientado a objetos y componentes, [Mey99], y el rendimiento de ejecución y utilización de recursos comparable a C y C++.

Debido a que uno de los objetivos principales de este trabajo era disponer de una solución final o marco de trabajo totalmente operativo, con la dificultad añadida de que pueda ser un sistema que permita tanto su extensión como modificación, la decisión de la metodología elegida para su desarrollo ha sido sumamente importante.

Las metodologías tradicionales, como el desarrollo en cascada (Figura 2.1) (que es un enfoque metodológico que dispone el proceso de desarrollo en etapas y una vez finalizadas se dispone del sistema final), han dejado de ser apropiadas para sistemas no cerrados y que pueden cambiar como es el caso de los sistemas energéticos: el sistema final se pretende que siga siendo un sistema vivo y que tenga capacidad para reaccionar ante modificaciones y cambios de requisitos que no pueden preverse a priori.

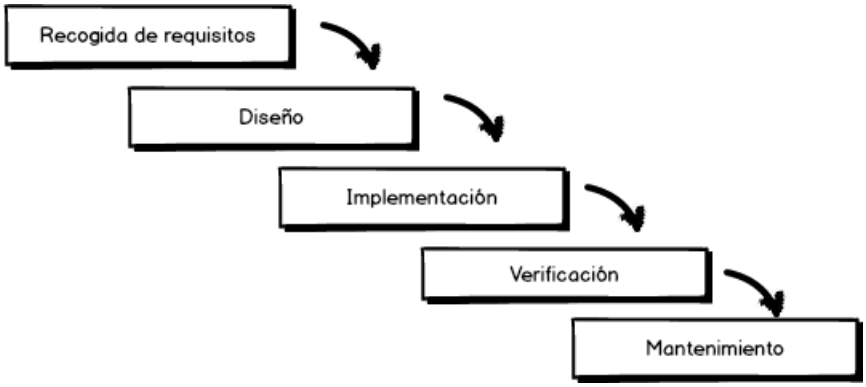

Figura 2.1: Metodología de desarrollo en cascada

Por ello, para los desarrollos de este trabajo se ha elegido la utilización de metodologías ágiles. Estas metodologías se basan en un desarrollo iterativo e incremental (Figura 2.2), [Gri91], de manera que se establece un ciclo de vida en cada iteración y se aplican diversas estrategias y procedimientos para mejorar la calidad del producto final.

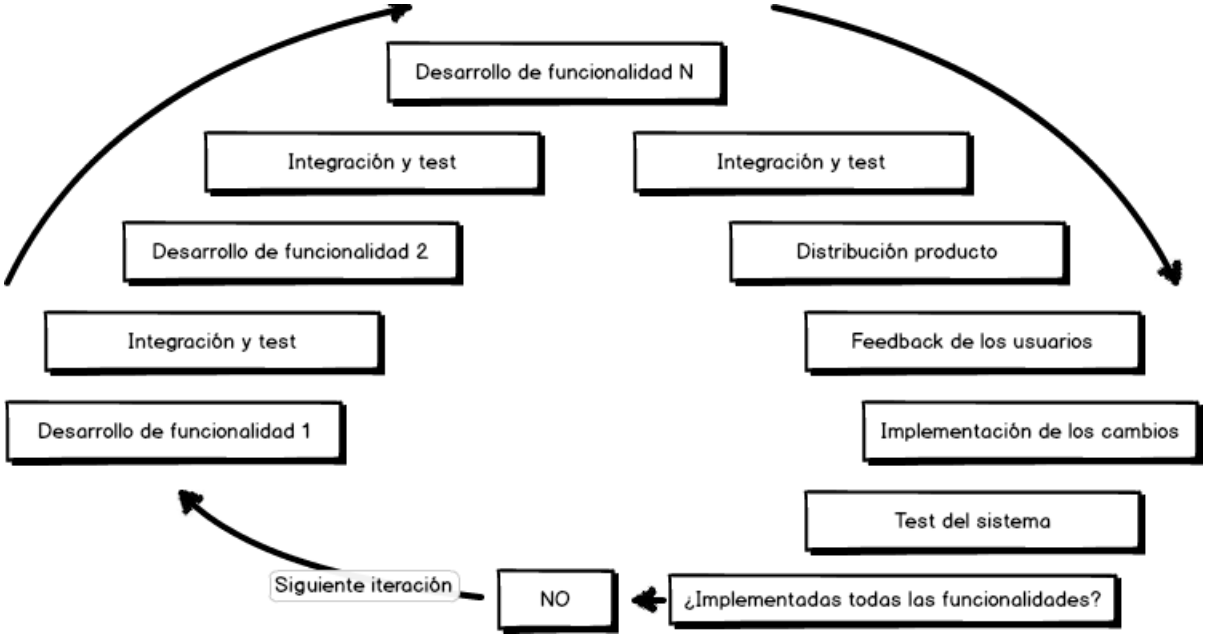

Figura 2.2: Metodología de desarrollo ágil

Las metodologías ágiles fueron inicialmente propuestas en febrero de 2001, en una reunión en la que estuvieron expertos en distintos tipos de metodología (Programación Extrema, SCRUM, DSDM, Desarrollo de Software Adaptativo, Crystal, Desarrollo Guiado por Características y otros similares) y cuyo objetivo era consensuar una alternativa para el proceso de desarrollo de software. Tras esta reunión hicieron público

el "Manifiesto para el Desarrollo Ágil de Software"[1] que valora:

- Individuos e interacciones sobre procesos y herramientas.
- Software funcionando sobre documentación extensiva.
- Colaboración con el cliente sobre negociación contractual.
- Respuesta ante el cambio frente seguir un plan.

El desarrollo de software, según este manifiesto, se basa en doce principios:

1. *Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.*
2. *Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.*
3. *Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.*
4. *Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.*
5. *Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.*
6. *El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.*
7. *El software funcionando es la medida principal de progreso.*
8. *Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.*
9. *La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.*
10. *La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.*
11. *Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.*

[1] Firmantes del manifiesto: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

El principal objetivo de la utilización de las metodologías ágiles es el de minimizar los riesgos. Para conseguir este objetivo los desarrollos que se hacen se van liberando en cortos periodos de tiempo. De esta manera, en cada ciclo o iteración se intentan añadir pocas funcionalidades y se hace énfasis en que la calidad del código sea la mejor posible. Normalmente, en este tipo de metodología se utilizan técnicas de refactorización [Ker04], pruebas unitarias e integración continua para el control, el seguimiento y garantizar la calidad del producto que está siendo desarrollado.

2.3.1. Programación extrema

La programación extrema o eXtreme Programming (XP) es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, [Bec99]. Es una de las metodologías ágiles de desarrollo del software. XP se diferencia de las metodologías tradicionales, al igual que el resto de las metodologías ágiles, en que pone mayor énfasis en la capacidad de adaptación al cambio de requisitos frente a la previsibilidad. El cambio de requisitos se considera dentro de las metodologías ágiles como algo natural, inevitable e incluso deseable, por lo que la metodología que mejor pueda enfrentarse a esta situación conseguirán una mayor tasa de éxito aproximándose de esta manera al comportamiento natural y de la manera menos traumática a lo que se considera la vida de un producto software. En este caso se invierte menos esfuerzo intentando definir todos los requisito para así disponer de mayores recursos al tener que evitar controlar los cambios de los mismos. Esto hace justamente algo deseable cuando nos enfrentamos al desarrollo de un sistema sujeto al cambio como un marco de trabajo.

La Programación Extrema puede considerarse como la utilización y aplicación de las mejores metodologías de desarrollo durante todo el ciclo de desarrollo del software aplicándolas de manera dinámica y continua.

Algunos de los valores sobre los que se sustenta la programación extrema son:

- Simplicidad: Es la base de la XP. Cuanto más simple esté desarrollado más facilidad de mantenimiento y menos errores podrán generarse. La simplicidad no solo se aplica al código fuente sino también a la documentación premiando el código autocomentado ya que su detallada expresividad desde el punto de vista sintáctico no afecta al rendimiento del mismo. En este sentido, se potencia el uso de técnicas de refactorización, donde el cambio sintáctico no afecta al comportamiento semántico del código, y que permiten simplificarlo significativamente, dotándolo de una mayor expresividad
- Comunicación: fundamentalmente a través del código autocomentado. El mejor canal de comunicación debe ser el propio código. Debe estar autocomentado de-

jando los comentarios explícitos solo para aquello que no vaya a variar como el objetivo de una clase o procedimiento. Otra de las formas de que el código se exprese debe ser a través de las pruebas unitarias donde se muestra realmente su funcionalidad con los tests aplicados. Todo esto sumado a la estrecha colaboración con otros programadores o el cliente (programación a pares), quien puede indicar qué prioridad tienen las funcionalidades a implementar, hacen que la comunicación sea un elemento de gran valor durante el ciclo de desarrollo software.

- Retroalimentación: Es importante que el usuario final se encuentre involucrado durante todo el desarrollo del proyecto disponiendo en todo momento de sus comentarios y necesidades, así como informarle continuamente de los avances gracias a los ciclos cortos de liberación del producto que está siendo desarrollado y que propone esta metodología.

Las características de la Programación Extrema la hacen apropiada como metodología con la que se ha abordado este proyecto y se ha materializado la arquitectura subyacente. Así, algunas de las técnicas más importantes que se pueden utilizar para aplicar esta metodología de manera adecuada son la refactorización, el desarrollo dirigido por pruebas y la integración continua.

Las técnicas de refactorización de código se presentaron inicialmente en 1991 en la tesis doctoral de William G. Griswold [Gri91]; una catalogación y explicación detallada de las mismas se recoge en [Fow99]. La refactorización se puede aplicar para mejorar la calidad y expresividad del código fuente. Es una técnica que ni resuelve errores ni añade ninguna funcionalidad pero su aplicación hace que el código sea más claro, que se reduzca su complejidad y que se genere un código más fácil de mantener. La aplicación de las técnicas de refactorización pueden aplicarse de manera indeterminada sobre todos los puntos del código fuente o puede aplicarse siguiendo una estrategia definida para, por ejemplo, llegar a aflorar patrones subyacentes dentro del código y así poder transformarlo en patrones conocidos, [GHJV95]. Esto se denomina refactorizar hacia patrones de diseño [Ker04].

Otra de las técnicas, como la aplicación de un ciclo de desarrollo dirigido por pruebas, puede describirse con los siguientes pasos (figura 2.3):

- Se escribe un test para una clase o módulo determinado, y se desarrolla el test hasta lograr compilarlo.
- Inicialmente el test aplicado a dicha clase o módulo no funciona ya que todavía no se ha implementado la funcionalidad requerida.
- Implementar la funcionalidad requerida y conseguir que pase el test.
- Una vez pasado el test sobre la funcionalidad buscada se aplican técnicas de refactorización.

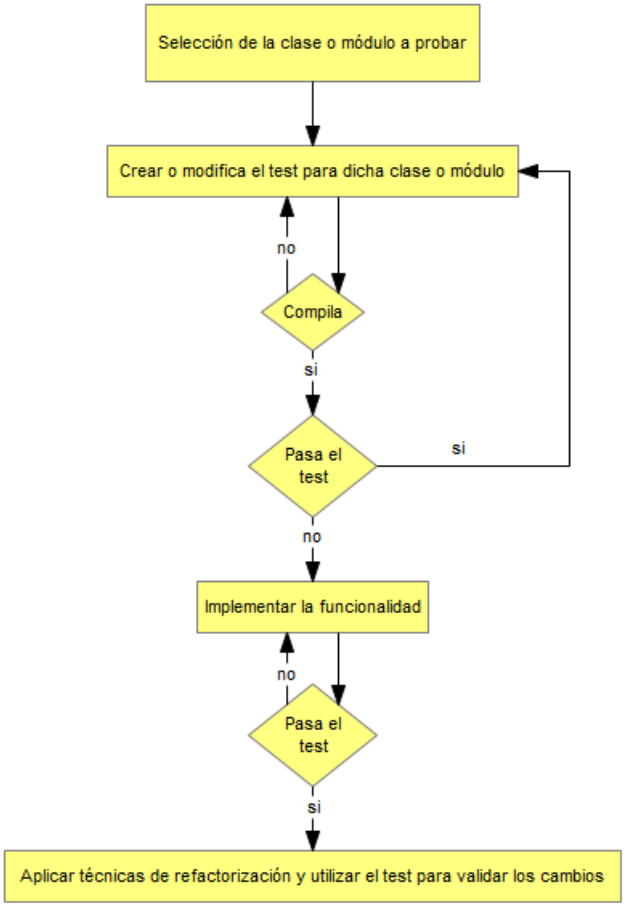

Figura 2.3: Ciclo de desarrollo basado en pruebas unitarias

El desarrollo es, por tanto, totalmente guiado por las pruebas. Además, la utilización de pruebas unitarias ayuda a la documentación, ya que el test describe por si mismo la funcionalidad a implementar, y permite mantener el código más desacoplado ya que la detección de un error se detecta en cada test y la propia naturaleza de la aplicación de un test hace que la implementación deba mantenerse aislada para poder ser probada.

Por último, de acuerdo con la propuesta hecha en [Bec99], la programación no es solo un problema de divide y vencerás, sino un problema de divide, prueba, integra y vencerás. Por ello, cada desarrollo que se ha hecho en este trabajo ha ido integrándose de manera continua. El este sentido, durante el desarrollo del marco de trabajo se han identificado unas etapas recurrentes y la comprobación para pasar de una etapa a otra se ha hecho de forma automatizada, en el sentido de que se ha comprobado que la base completa del código compila sin errores y cada unidad pasa los tests correspondientes. Disponer de estos mecanismos ha permitido, por una parte, tener la seguridad de que lo que se ha ido desarrollando funcionaba, y por otra, acortar el tiempo de producción.

Algunos de los pasos que se han seguido para asegurar la integración continua son:

- Integración de los diferentes recursos externos necesarios que forman parte de los

requisitos necesarios para compilar o ejecutar el proyecto.

- Vigilancia del repositorio de código del proyecto en busca de modificaciones para compilar o ejecutar el producto.
- En caso de compilación correcta ejecutar los test unitarios que forman parte de proyecto.
- Generación del producto y/o despliegue sobre un sistema o entorno de desarrollo de producción o pruebas.

2.4. Conclusiones

En este capítulo se ha hecho una revisión de lo que es un marco de trabajo y su utilización en distintos campos, ya que la propuesta de este trabajo es desarrollar un marco de trabajo para la gestión de sistemas energéticos.

Para el desarrollo de este marco de trabajo y debido a su complejidad, se ha considerado que deben utilizarse metodologías que permitan tener en cuenta la especificidad de este tipo de sistemas, fundamentalmente los cambios que se requerirán a los sistemas de manera que permitan la integración de nuevos componentes y/o módulos como la incorporación de modelos de análisis y mecanismos de evaluación y predicción del funcionamiento. Para dar respuesta a esto, se han analizado las distintas metodologías de desarrollo que pueden utilizarse y se han descrito aquellas que se proponen utilizar para el desarrollo de la parte práctica de este trabajo.

Se propone la utilización de metodologías ágiles, así como un proceso de construcción y despliegue automatizado que permita la ejecución de test de unidad para evitar la inclusión de nuevos errores y garantizar así una calidad no solo en el resultado final sino del producto al completo durante todas las fases de su desarrollo.

"Programar sin una arquitectura o diseño en mente es como explorar una gruta solo con una linterna: no sabes dónde estás, dónde has estado ni hacia dónde vas"

– Danny Thorpe

3

Marco de trabajo para el desarrollo de software para la gestión de sistemas energéticos

3.1. Introducción

Una vez establecidos los conceptos de marco de trabajo y arquitectura software así como las distintas metodologías de desarrollo que pueden ser utilizadas, en este capítulo se presenta el marco de trabajo y la arquitectura que se proponen para el desarrollo del software que permita la gestión de sistemas energéticos. En este ámbito, el marco de trabajo que se propone puede ser definido como:

Un conjunto integrado de clases e interfaces que pueden ser utilizadas por los desarrolladores de sistemas de adquisición de datos, gestión energética, evaluación del funcionamiento y predicción de la producción de sistemas energéticos para el desarrollo de soluciones software que den respuesta a estas necesidades.

Como en cualquier marco de trabajo, se hace una propuesta de la arquitectura del mismo para que sea posible después desarrollar una estructura de clases e interfaces que implementen las funcionalidades requeridas y que pueda ser fácilmente ampliada y utilizada gracias a las técnicas de herencia, polimorfismo, composición e implementación. El marco de trabajo se ha desarrollado utilizando metodologías ágiles como las expuestas en el capítulo 2 para garantizar una calidad en el software resultante. A partir del mismo, se pueden desarrollar programas para la gestión de sistemas energéticos de ma-

nera sencilla dando solución a los problemas de definición, modelizado, comunicación, publicación y análisis de dichos sistemas.

Según lo propuesto por [Sch97], en el dominio de la gestión de sistemas energéticos de media y baja potencia, está justificado el desarrollo de un marco de trabajo ya que se cumplen las dos condiciones que se apuntan en esa propuesta: el mercado en el que se utilizará empieza a ser muy competitivo y exige el desarrollo de aplicaciones en plazos muy cortos de tiempo y el dominio de aplicación es también complejo y, como se apuntó en el capítulo 1 implica el desarrollo de aplicaciones que resuelven los mismos problemas para distintos sistemas. En el trabajo de [SAL+] se analizan los requisitos y se hace una primera propuesta de una arquitectura reconfigurable para su utilización en dispositivos electrónicos y recursos de energía renovable.

3.2. Propuesta de arquitectura software

La arquitectura software que se propone en esta tesis se basa en la separación del software en varias capas, cada una de las cuales incluye una serie de componentes basados en interfaces especializados que ofrecen funcionalidades para la definición, modelizado, comunicación, almacenamiento, supervisión, evaluación y predicción del funcionamiento de sistemas energéticos genéricos. Además, tiene en cuenta las similitudes que presentan los sistemas energéticos genéricos, dejando las particularidades de cada tipo de sistema para su consideración como extensiones al marco de trabajo. En la misma se definen los estándares que se utilizarán para el establecimiento de conexiones y coordinación entre los componentes de cada capa.

Como una parte muy importante de la propuesta, se propone una solución al problema de la unificación de los mecanismos de comunicación con todos los dispositivos que forman parte de una instalación. Para ello se desarrollarán mecanismos que permitan resolver de una manera sencilla los problemas de interconexión. La propuesta de utilización de este mecanismo de comunicaciones se extiende también para los componentes específicos de análisis, evaluación y predicción basados en sistemas inteligentes y de aprendizaje automático que se proponen en esta tesis y que se describen en detalle en el capítulo 8. Por otra parte, el desarrollo del marco de trabajo en varias capas permite desacoplar la complejidad del sistema y además facilita que se puedan implementar nuevas funcionalidades y/o componentes de manera más rápida y sencilla.

En concreto la arquitectura propuesta para la gestión de sistemas energéticos tiene en cuenta:

- La definición o caracterización del sistema que se desea gestionar, por lo que debemos disponer de mecanismos para el modelizado de las instalaciones.
- El tipo de servicio que se ofrece, que puede ir desde solo la monitorización hasta

la predicción del funcionamiento, pasando por otro tipo de servicios como son supervisión y vigilancia energética, análisis y evaluación.

- Las formas de interacción entre los usuarios y el sistema, lo que incluye los distintos perfiles y roles de usuario que habrá que considerar.

- La integración de los diferentes dispositivos que forman parte del sistema y de los que hay que resolver la forma de intercambio de información entre ellos centralizando en la medida de lo posible los datos que cada uno suministra.

- Los mecanismos de interconexión entre el sistema completo y el exterior para disponer de un canal de comunicación en caso de sistemas remotos o de publicación externa.

- Las necesidades o requisitos finales a los que dar solución para disponer de una visión tanto cuantitativa como cualitativa de un sistema energético completo.

Se debe dar respuesta a estos requisitos de manera flexible para que se puedan, a partir de la misma, extender y desarrollar las soluciones software en función de los subsistemas y configuración de cada sistema energético.

El diseño de la arquitectura software para el desarrollo del marco de trabajo se ha basado en la separación del software en varias capas y en la definición de las relaciones entre los componentes en cada capa. La representación gráfica de la arquitectura propuesta se muestra en la figura 3.1. En esta representación se muestran de forma visual los componentes de la arquitectura y sus relaciones.

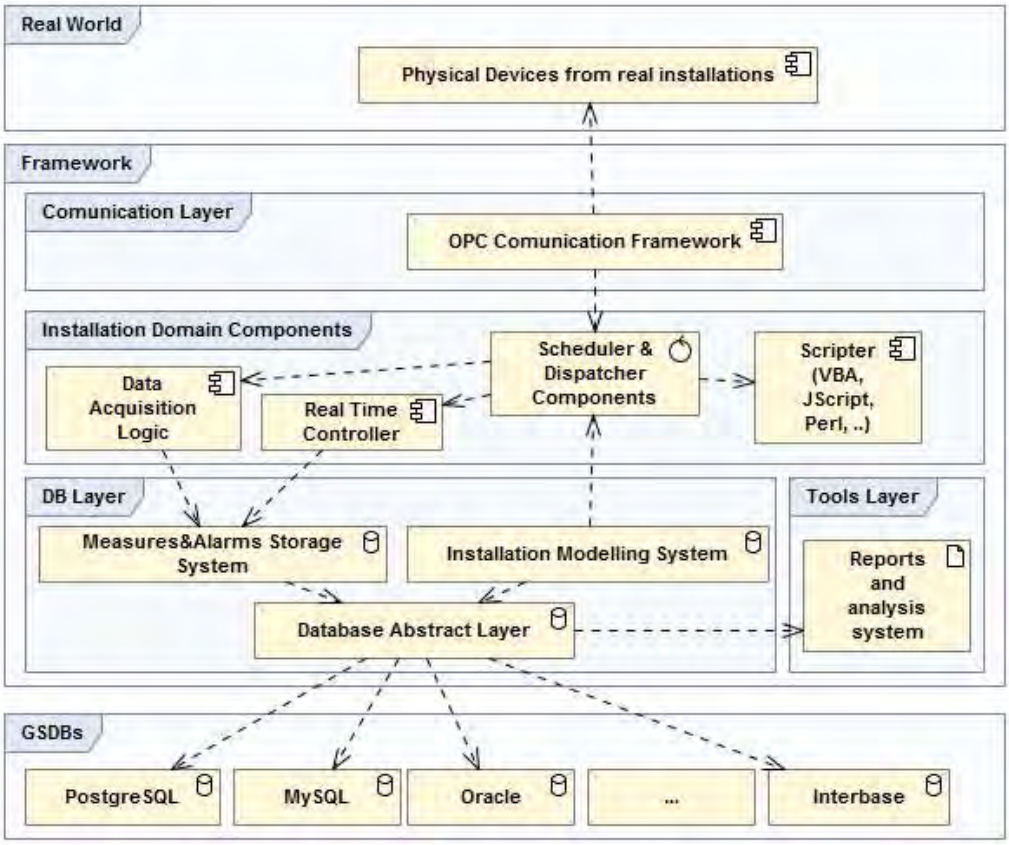

Figura 3.1: Arquitectura software propuesta

3.2.1. Definición de las capas del marco de trabajo

Las capas que se han incluido son las siguientes:

1. Acceso a la base de datos: la capa más baja proporciona un acceso abstracto a la gestión de la base de datos utilizada, tanto para almacenar el modelizado de los programas de gestión y evaluación que se desarrollan a partir del marco de trabajo como para recuperar la información de los dispositivos que se integran en cada sistema para el que se construye una aplicación. Gracias a utilizar una capa intermedia, se evita tener una dependencia estricta con un sistema gestor de base de datos determinado. Este nivel de desacoplo permite la utilización de cualquier sistema gestor de base de datos, ya sea propietario o de código abierto. En esta capa se incluyen los siguientes componentes:

   - El sistema de almacenamiento de los valores de los canales y alarmas de cada dispositivo, que es el responsable de guardar los datos recibidos desde la capa superior y de proporcionar los mecanismos de persistencia para estos datos.

- El sistema de caracterización de la instalación, que proporciona herramientas para modelizar las instalaciones y sus subsistemas, incluyendo su representación de una forma genérica y ofrece, por tanto, un modelo con persistencia de datos para la monitorización y evaluación de dichos sistemas.
- La capa abstracta de base de datos, que proporciona la comunicación con el gestor de la base de datos. Esta capa establece las asociaciones entre los datos recuperados y el modelizado del sistema.

2. Capa del Dominio: esta capa permite caracterizar cada sistema energético para el que se quiera construir una aplicación de gestión a partir del marco de trabajo desarrollado. Esta caracterización se hace de manera genérica e incluye la caracterización de los subsistemas que lo componen; se utiliza un modelo con persistencia de datos. Se realiza una descripción conceptual del sistema junto con todos los dispositivos que lo integran lo que permite que sean tratados de manera abstracta y consistente. Los elementos y dispositivos del sistema energético se modelizan junto con todos los canales y fuentes de datos, así como sus características físicas y técnicas. El marco de trabajo junto con la base de datos proporciona un canal de acceso al repositorio de componentes (dispositivos y modelos implementados) para facilitar el diseño y modelizado de cada sistema. En esta capa se desarrollan los siguientes componentes relacionados con el dominio de aplicación del marco de trabajo:

   - Generador de scripts que permite incluir lenguajes de guión o archivos de procesamiento por lotes programados en distintos lenguajes para modificar los valores de los canales y medidas, tal y como se explicará más adelante. También se permite generar medidas estimadas utilizando para ello funciones programadas en las aplicaciones finales generadas.
   - El subsistema de tiempo real permite la conexión física con las plantas que se monitorizan y evalúan en tiempo real, estableciendo una comunicación con cada dispositivo de la instalación siguiendo la configuración adecuada definida anteriormente.
   - El subsistema de adquisición de datos permite recuperar los datos históricos almacenados en los subsistemas de cada uno de los dispositivos que conforman una instalación. Se conecta a ellos siguiendo la particularidad de cada uno definida anteriormente y descarga la información relevante que es necesaria para su posterior análisis.
   - El planificador y gestor de las conexiones es el sistema responsable de definir y ejecutar las tareas programadas o planificadas para la conexión y recuperación de la información de los dispositivos de una instalación dada utilizando los componentes de la capa anteriormente definidos.

   Esta capar permite utilizar tanto la capa comunicación para conectar con cada dispositivo como programar scripts para que sean ejecutados. Esta capa establece la asociación entre los datos recuperados y el modelizado de sistemas.

3. La capa de comunicaciones está conectada a la capa de dominio de la instalación proporcionando el modelizado de la conexión y la capacidad del intercambio de

información con cada dispositivo. La recuperación de la información puede realizarse bien utilizando *tiempo real* o bien mediante un *planificador* para recuperar la información almacenada en cada dispositivo. Como ya se ha mencionado, después de que los datos se hayan recuperado desde de los distintos dipositivos, es posible utilizar scripts que ejecuten las operaciones especificadas, en la capa de modelizado, para cada uno de los elementos incluidos en cada dispositivo con el fin de realizar las transformaciones necesarias para su análisis o representación.

4. La capa de herramientas es donde se desarrollan las utilidades que permiten, entre otras, la elaboración de informes o la publicación de la información almacenada en las bases de datos. Esta capa tiene acceso a todos los datos almacenados en la base de datos así como el modelizado y caracterización de cada sistema.

Con esta arquitectura basada en capas o niveles se permite que se construyan soluciones para la gestión de instalaciones genéricas donde cada capa es la responsable de cubrir cada una de las necesidades que surgen para la gestión de este tipo de instalaciones. Gracias al modelizado de la instalación y de cada dispositivo, a la definición e implementación del protocolo de comunicaciones y a la caracterización de cada canal de información se puede disponer de un sistema de tratamiento de la información de la instalación que se vaya a gestionar.

En los siguiente capítulos se presentarán los modelos de datos y estructuras necesarias para la implementación de la arquitectura de capas propuesta, de manera que se pueda representar de manera abstracta cualquier sistema energético que se quiera monitorizar y gestionar partiendo de la descripción de cada uno de los subsistemas y dispositivos que lo integran.

3.3. Marco de trabajo propuesto

La implementación de esta arquitectura de capas se realizará utilizando clases e interfaces que junto a las relaciones entre ellas formarán parte del marco de trabajo a partir del que se podrán desarrollar aplicaciones de manera más rápida, ya que proporcionará gran parte del código necesario para el manejo de los distintos escenarios. Permitirá también la extensión de servicios que las clases proporcionan de una manera sencilla. Así, el marco de trabajo proporcionado por la biblioteca de clases que se propone es un importante y facilitador punto de partida para construir nuevas aplicaciones permitiendo una reducción importante en el tiempo de desarrollo de estas aplicaciones y la reutilización y portabilidad del código desarrollado.

Desde el punto de vista funcional las clases que se proponen se pueden clasificar de la siguiente forma:

- Clases orientadas a la interfaz de usuario

- Clases de propósito general para ayuda a la implementación
- Clases para la gestión de usuarios
- Clases para la gestión de plantas y dispositivos
- Clases para la gestión y configuración de los parámetros de las comunicaciones y protocolos
- Clases para la utilización modelos de evaluación y predicción

3.4. Propuesta de metodología de desarrollo

La metodología que se propone aplicar para el desarrollo del marco de trabajo se basa en la utilización de metodologías ágiles aplicando ciclos iterativos, según lo que se ha descrito en el capítulo 2. Esto permite ir refinando el diseño y la implementación llegando a ciertas conclusiones y buenas prácticas. Con esta metodología se puede desarrollar un buen diseño que sea altamente modular, que tenga las mínimas dependencias entre clases e interfaces y se dispone de un sistema altamente desacoplado pero fácilmente extensible.

La **propuesta** que se hace en este trabajo pasa por la **utilización masiva de interafaces, la poca utilización de los mecanismos de herencia frente a los mecanismos de composición y la implementación contra abstracciones**. Basándose en estas premisas se propone una solución al problema del diamante en la arquitectura software que ha sido de mucha utilidad en el desarrollo del marco de trabajo.

3.4.1. Implementación contra abstracciones

Uno de los objetivos de un desarrollo altamente desacoplado consiste en separar la definición de la implementación y el elemento disponible para realizar este tipo de arquitectura es la utilización de interfaces. De esta manera, un interfaz define una relación de servicios disponibles sin tener en cuenta la implementación de los mismos.

Así, siempre que se establezca un interfaz para definir reglas o servicios disponibles se estará creando una dependencia contra servicios sin tener en cuenta los detalles de la implementación de los mismos por lo que no existirán referencias a otros puntos del código. Además existe la ventaja de que al poder separar completamente la implementación de la lógica, se pueden realizar pruebas o test unitarios sobre las interfaces y cambiar la implementación de manera sencilla para probar el código que se va generando.

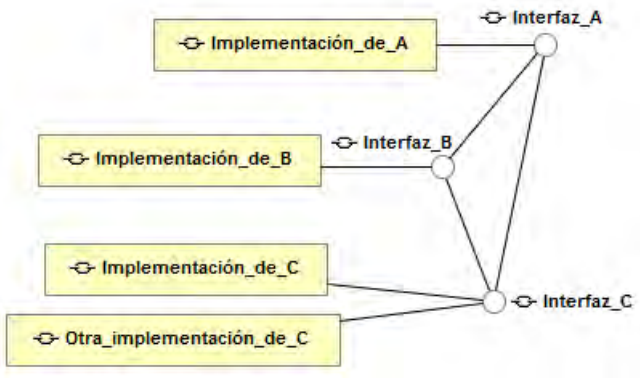

Figura 3.2: Conjunto de interfaces que definen el marco de trabajo

El marco de trabajo desarrollado consistirá por lo tanto, en un conjunto de interfaces relacionadas y clases que implementan dichas interfaces y pueden ser extendidas cumpliendo la implementación a la que se comprometan según su interfaz, según se muestra, como ejemplo, en la Fig. 3.2.

3.4.2. Desarrollo utilizando interfaces

Durante el desarrollo de este marco de trabajo ha sido necesario desarrollar una gran cantidad de interfaces para la definición del mismo. Así, la declaración de la interfaz estipula la relación entre otras interfaces y declara el comportamiento que debe implementar la clase que declare dicho interfaz, según se muestra en la Fig. 3.3.

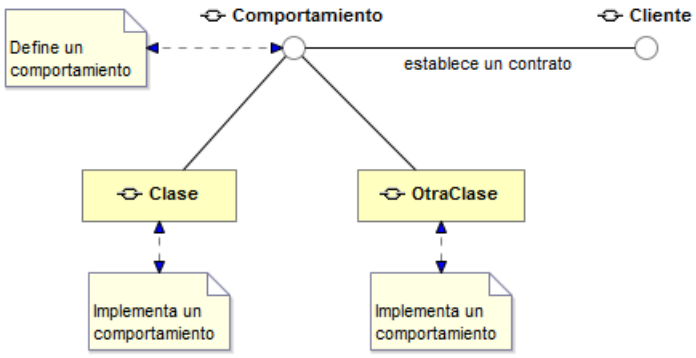

Figura 3.3: Implementación contra abstracciones

De esta manera, una interfaz se relaciona con otra simplemente por su declaración pero deben ser la clases que las implementan las encargadas de hacer valido dicho comportamiento a nivel semántico.

3.4.3. Composición frente herencia: una propuesta de solución al problema del diamante en la arquitectura software

Los mecanismos principales disponibles para la extensión y relación entre clases son la composición y la herencia. Durante mucho tiempo se ha premiado el uso de la herencia frente a otros mecanismos pero han surgido muchos problemas debido a que es muy difícil crear una jerarquía y una relación entre clases poco acopladas. Este problema, comúnmente conocido como el *problema del diamante*, ver Fig. 3.4, ha llevado a redefinir la forma en las que modelizar las arquitecturas de clases en otras donde se eviten estos problemas.

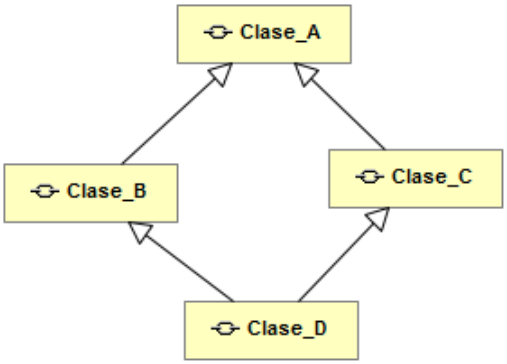

Figura 3.4: Herencia múltiple o problema del diamante

Para dar solución a este problema, los lenguajes de programación como C++ han permitido la herencia múltiple utilizando mecanismos para desambigüar en caso de que existan métodos o atributos repetidos en las clases superiores. Sin embargo, el lenguaje de programación que se ha utilizado en este trabajo, basado en Object Pascal, no permite la utilización de la herencia múltiple, al ceñirse a un paradigma de programación orientada a objetos más puro, por lo que se ha tenido que buscar una solución más apropiada.

Haciendo uso de la capacidad de poder asignar varios interfaces a una misma clase, la propuesta que se hace es la de asignar a las clases con varios ancestros la definición de comportamiento, en lugar de implementaciones. De esta forma se pasa de una herencia múltiple a la asignación de múltiples definiciones de interfaz, según se muestra en la Fig. 3.5.

En muchos casos, esta solución resuelve los problemas asociados a la herencia múltiple, pero en el caso de que la finalidad sea la de reutilizar código implementado por el ancestro, acarreará un nuevo problema. El hecho de resolver de esta manera el problema del diamante conlleva la sobrecarga de la implementación de las clases inferiores al tener que soportar la reescritura del código de las clases superiores para cumplir con los requisitos que imponen las interfaces. Una mejora a esa primera solución es la que se muestra en la Fig. 3.6, que evita la situación descrita.

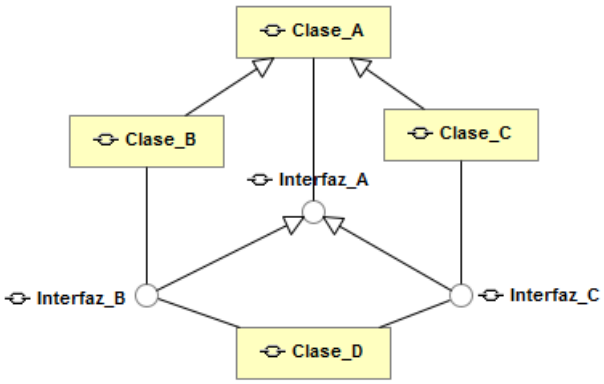

Figura 3.5: Primera solución al problema del diamante

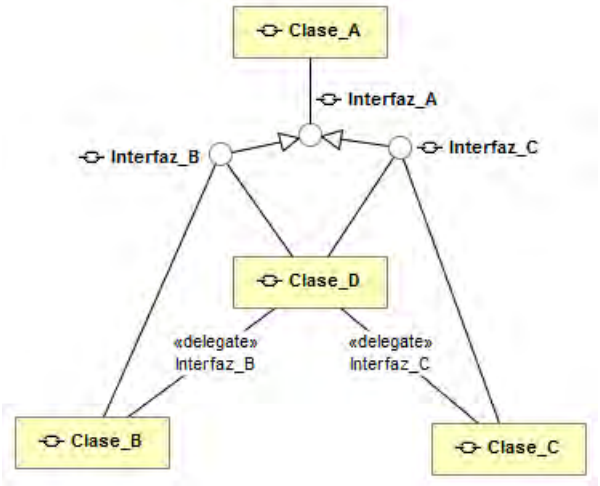

Figura 3.6: Mejora a la primera solución propuesta al problema del diamante

En este caso, la clase inferior no necesita implementar los interfaces superiores ya que puede implementarlos por delegación sobre referencias a clases que internamente puede crear. Esta solución revela una estructura adecuada y permite generar una arquitectura altamente desacoplada haciendo uso de la declaración, delegación y composición de interfaces.

3.4.4. Implementación por delegación de interfaces

La implementación por delegación de interfaces es un mecanismo que permite a una clase pasar la responsabilidad de cumplimiento de una interfaz a otra clase que normalmente suele ser un atributo de la misma creada con esa finalidad.

Se puede afirmar que: *Dado un conjunto de interfaces, existirá al menos una clase que implemente dicho interfaz comprometiéndose a validarlo sintácticamente y será validado semánticamente si dicha implementación supera los test asociados a dicho interfaz haciendo uso de dicha clase.*

En el siguiente código se muestra la propuesta para resolver el problema de la herencia múltiple utilizando interfaces y aplicando la delegación para no tener que reimplementar los métodos asociados a dichas interfaces:

Listing 3.1: Delegación de la implementación de un interfaz

```
1 Class_D = class(TObjec,Interfaz_B, Interfaz_C)
2 private
3   fImplIntf_B : Interfaz_B;
4   fImplIntf_C : Interfaz_C;
5 pubic
6   property ImplementaIntf_B : Interfaz_B read fImplIntf_B write fImplIntf_B implements Interfaz_B;
7   property ImplementaIntf_C : Interfaz_C read fImplIntf_C write fImplIntf_C implements Interfaz_C;
8 end
```

Una vez delegada la implementación del interfaz a los atributos de la clase es posible asignar a los mismos clases que realmente implementan el comportamiento deseado. Esto es se puede realizar en el interior del constructor de la clase asignando la implementación deseada, de acuerdo al siguiente código:

Listing 3.2: Implementación de la interfaz por delegación de manera implícita

```
1 constructor Class_D;
2 begin
3   fImplIntf_B := Clase_que_implementa_interfaz_B.create;
4   fImplIntf_C := Clase_que_implementa_interfaz_C.create;
5 end
```

Haciendo uso de la composición de clases y la delegación de interfaces, el marco de trabajo puede extenderse y seguir un esquema desacoplado al no existir relaciones directas entre clases en favor de las relaciones establecidas entre interfaces. Así, una clase puede cambiar de implementación cambiando la clase sobre la que delega la definición de la interfaz que debe cumplir sin cambiar las relaciones entre la lógica establecida por el marco de trabajo y sus interfaces. Sin embargo, aunque de esta manera se está dotando al marco de trabajo de la capacidad de ser fácilmente conectable y configurable, también se está introduciendo un punto de rigidez al establecer dentro de dicha clase la decisión de qué implementación utilizar. Para resolver esto se propone la utilización en la implementación del concepto de inyección dependencias.

3.4.5. Inyección de dependencias

La inyección de dependencias consiste en utilizar un mecanismo de asignación de manera externa de las dependencias que una clase pueda tener permitiendo desacoplar dicha clase del resto del sistema, según se muestra en la Fig. 3.7. Este mecanismo permite además poder probar dicha clase ya que puede ser utilizada de manera aislada y conectarla a su vez con diferentes implementaciones.

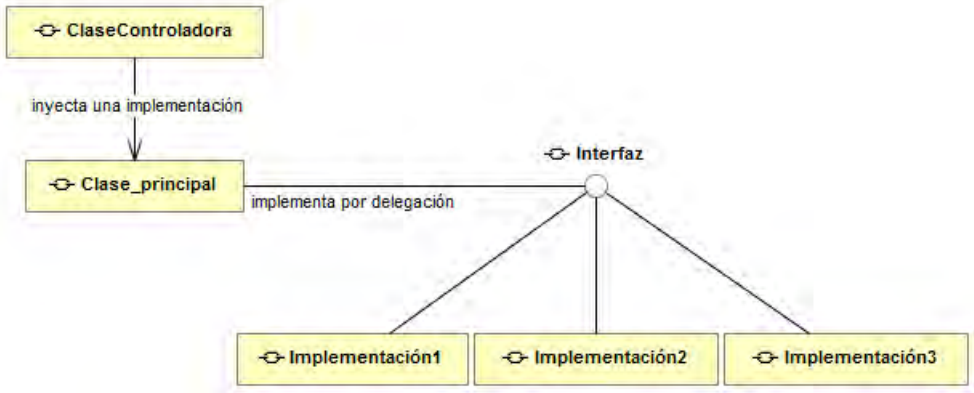

Figura 3.7: Inyección de dependencias

Un mecanismo básico de la inyección de dependencias es la inyección en el constructor, según se muestra en el siguiente código:

Listing 3.3: Inyección de dependencias durante la construcción de la instancia de clase

```
1  constructor Class_D( aClassB : Interfaz_B; aClassC : Interfaz_C );
2  begin
3    fImplIntf_B := aClassB;
4    fImplIntf_C := aClassC;
5  end
6
7  begin
8    // Se intecta a la clase D dos implementaciones particulares
9    Class_D.create(Implementacion1IntfB.create, Implementacion3IntfC.create);
10 end
```

En este ejemplo se muestra cómo se inyecta la dependencia durante la construcción del objecto permitiendo de esta manera que la rigidez que produce la introducción de una dependencia se realice en el nivel más externo del código. Este mecanismo será muy utilizado en el desarrollo del marco de trabajo propuesto al haber demostrado ser una buena práctica en el diseño de software.

3.5. Conclusiones

En este capítulo se ha descrito de forma general la arquitectura software y el marco de trabajo propuesto para el desarrollo de sistemas de gestión energética. Para el marco de trabajo propuesto se ha optado por utilizar una arquitectura basada en capas donde cada una de ellas tiene una función determinante para el conjunto del sistema. Se ha descrito cada una de las capas y se ha puesto de manifiesto que esta organización permite disponer de una distribución de las responsabilidades de manera aislada y altamente desacoplada.

Además se han descrito algunos de los problemas que surgen cuando se diseña la arquitectura de un sistema y se han propuesto soluciones que evitan estos problemas. La solución propuesta proporciona características de algo grado de desacoplo, altamente independiente y fácilmente extensible y configurable, facilitando así soluciones para problemas comunes en el desarrollo software así como proporcionando indicaciones para una implementación eficiente de la arquitectura. Haciendo uso de las técnicas de desarrollo orientado hacia abstracciones, se dispondrá de una metodología para la codificación eficiente y desacoplada de todos los sistemas que serán necesarios para desarrollar el marco de trabajo.

Por último, tal y como se explicó en el capítulo 2, a la hora de afrontar un desarrollo software es necesario disponer de una metodología apropiada. En este capítulo se ha complementado la elección de la metodología con varias propuestas que facilitan seguir unos paradigmas de programación que permiten poner de manifiesto una potencia expresiva suficiente para plasmar la arquitectura de manera precisa y eficiente.

"El mayor problema en la comunicación es la ilusión de que se ha logrado"

– George Bernard Shaw

4

Capa de comunicaciones abstracta

4.1. Introducción

En este capítulo se describe la capa de comunicaciones abstracta. Se propone un sistema de comunicaciones con los dispositivos que está totalmente separado de las demás funcionalidades que se incluyen en el marco de trabajo. El objetivo de la propuesta que se hace es conseguir una conectividad abierta utilizando estándares abiertos. Por otra parte, basándose en el modelo que se propone en esta tesis, la integración de modelos de evaluación y predicción en los sistemas de gestión energética que se desarrollen a partir del marco de trabajo propuesto, se hará partiendo del mismo esquema de abstracción que se propone para las comunicaciones. Como se justificará más adelante, en este mismo capítulo, esto permitirá separar estos modelos del sistema, de manera que se puedan modificar, ajustar, etc. para cada instalación sin necesidad de tener que rediseñar el sistema. Así, la *salida y resultados* de estos modelos, se proporcionará al sistema como un componente más.

Como es conocido, uno de los grandes retos a los que debe dar respuesta cualquier sistema de monitorización y gestión de plantas energéticas es, sin duda, la comunicación del sistema con todos los dispositivos que forman parte del sistema. Ésta debe ser eficiente, versátil y flexible.

Además, es necesario que se puedan integrar los sistemas de automatización para distintos sistemas energéticos, como pueden ser parques eólicos, sistemas energía solar térmica, sistemas de energía solar fotovoltaica, tanto aislados como conectados a la red eléctrica, sistemas de cogeneración, etc.

Debido a la gran cantidad de dispositivos, marcas, fabricantes y modelos distintos de dispositivos que pueden formar parte de un sistema energético, implementar un protocolo de comunicaciones para cada uno de ellos supondría un gran esfuerzo y una gran inversión en tiempo, teniendo en cuenta, además, que no siempre se dispone de los protocolos para todos los tipos de dispositivos que se quieran incorporar. Lo que se plantea en este trabajo es utilizar el estándar OPC (OLE/COM [Gor01], Object Linking and Embedding / Component Model), por ser un sistema estándar y homogéneo de comunicaciones.

Se trata de conseguir:

- Generalización en las comunicaciones. Así, la propuesta de utilizar un estándar, a nivel de implementación permite:
  - Facilidad y flexibilidad en el desarrollo: Al tener una versión implementada de comunicaciones con el protocolo y ser este un estándar necesitará relativamente pocos cambios para reutilizarlo con distintos dispositivos y añadir distintas funcionalidades.
  - Versatilidad en la monitorización de distintos dispositivos: El protocolo se basa en un estándar aprobado y bien definido que implementan distintos modelos de dispositivos y que podrán ser incorporados al marco de trabajo modificando el servidor de comunicaciones que se desarrolle.
- La utilización de distintos canales de comunicación: Al tener implementado un servidor OPC genérico se tiene la posibilidad de establecer comunicación de forma remota con dispositivos utilizando distintos medios como pueden ser tecnologías GSM, GPRS o IP.
- Creación de un modelizado de dispositivos genéricos que no tiene un tipo ni modelo concretos, con lo que la inclusión de los dispositivos y el almacenamiento de los datos que de ellos se extraiga, permitirá una fácil e inmediata integración en la base de datos.

Para dar respuesta a estos requisitos se propone descomponer el sistema de comunicaciones en tres niveles:

- Nivel del dispositivo: en este nivel se tiene en cuenta el dispositivo concreto con el que se va a intercambiar la información y en él se configuran los parámetros específicos de cada dispositivo.
- Nivel del protocolo: Se define el protocolo de comunicaciones entre el dispositivo y el sistema de gestión.
- Nivel del medio físico: se define el medio físico por el que se va a efectuar el intercambio de información entre el sistema y el dispositivo. En este nivel se configuran los parámetros específicos del medio físico elegido.

4.2. Tecnología OPC

Para establecer las comunicaciones con los distintos dispositivos físicos y *virtuales* que forman parte de un sistema energético, correspondientes al nivel de protocolo, se propone, como se ha comentado anteriormente, la utilización del estándar de comunicación basado en la tecnología OPC, que permite resolver de una manera sencilla los problemas de interconexión, según la arquitectura OPC Cliente/Servidor que se muestra en la figura 4.1.

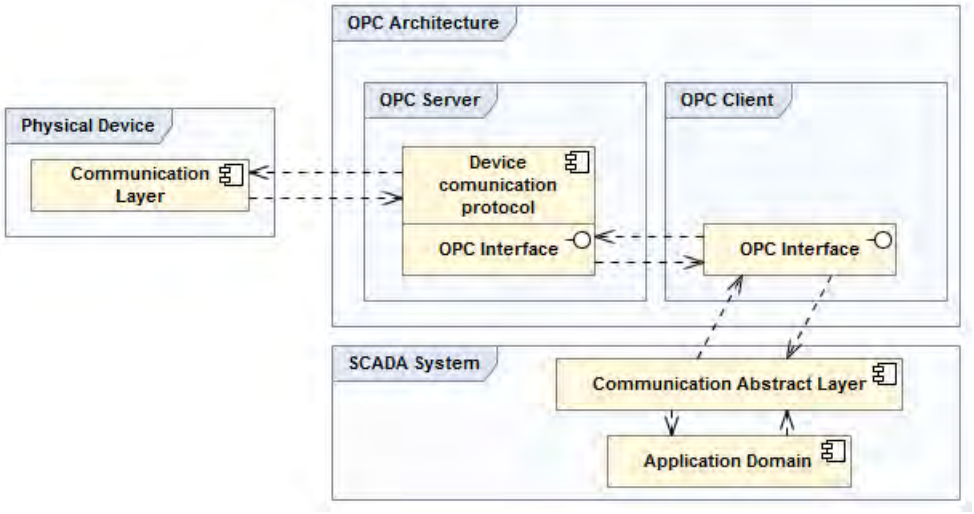

Figura 4.1: Arquitectura OPC Cliente/Servidor

La tecnología OPC está basada en la tecnología OLE/COM (Objetct Linking and Embedding/ Component Model from Microsoft), [Gor01], [LLH+05]. Por una parte, OPC es un estándar abierto para comunicaciones utilizado en el sector industrial para conectar los sistemas de supervisión y adquisición de datos e interfaces hombre-máquina con los sistemas físicos de control, [Hol04], y se ha utilizado ya en el ámbito de la monitorización de sistemas energéticos, por ejemplo, en [GCTL08]. Por otra parte, OPC es un estándar que permite el desarrollo de componentes para interconectar sistemas dispersos creando soluciones robustas y proporcionando interoperabilidad de una manera eficiente, ya que la utilización de este estándar permite reducir tiempos y costes en el desarrollo de sistemas de control. OPC define un estándar de intercambio de información y las reglas de negociación entre dispositivos de diferentes tipos utilizando el paradigma cliente/servidor. Además, OPC permite de manera sencilla soluciones totalmente escalables que facilitan futuros cambios y la ampliación de los sistemas que se desarrollan usando este estándar. El diseño de interfaces OPC soporta arquitecturas distribuidas.

En la actualidad el estándar OPC está apoyado por más de 200 empresas e instituciones como son Microsoft, CERN, Compac o National Instruments, lo que garantiza un mantenimiento, revisión y mejoras continuas del estándar para dar respuestas a los nuevos desarrollos. La evolución del estándar OPC está soportado por la fundación

OPC que es la responsable de crear las especificaciones y dirige la evolución de las mismas.

El acceso a servidores OPC remotos se realiza utilizando la tecnología Distributed Component Object Model (DCOM) [Gor01]. Esta tecnología permite que componentes heterogéneos operen en distintas plataformas y protocolos de red. Los objetivos que se persiguen al usar esta tecnología es que así será posible dar respuesta de manera más rápida y eficaz a cambios en la configuración de los sistemas que se gestionen, permitiendo además interoperar en ambientes heterogéneos y cambiantes e incorporar nuevas tecnologías (de los susbsistemas que integran los sistemas energéticos objeto de esta tesis).

Algunos de los problemas que se han planteado a la utilización del estándar OPC y a las tecnologías DCOM en sistemas industriales son el coste computacional que tiene esta tecnología con las implementaciones actuales de los protocolos, [MT07], lo que hace que no sean muy adecuadas para sistemas en tiempo real. Sin embargo, para el desarrollo de programas de gestión y evaluación de sistemas energéticos de pequeña y media potencia, que son el dominio de aplicación de este trabajo, esto no es una barrera ya que todas estas tareas pueden hacerse con cierto retraso sin que esto suponga ningún problema (hay que tener en cuenta que lo que se propone en este trabajo es un desarrollo para la gestión y evaluación, no la actuación sobre las plantas, tarea que está siendo desarrollada hasta la fecha por personal especializado).

En un escenario típico donde convergen multitud de dispositivos diferentes se hace necesario disponer y desarrollar un controlador apropiado para cada dispositivo. En sistemas pequeños esto puede ser asumido pero cuando deseamos tener una solución general para cualquier tipo de problema vemos que esta complejidad añadida hace que cualquier configuración se vea repercutida negativamente por la necesidad de interconexión entre todos los sistemas como puede verse en la figura 4.2.

OPC define un estándar de intercambio de información y las reglas de negociación entre dispositivos de diferentes tipos utilizando el paradigma cliente/servidor. Así cualquier dispositivo que posea un software de control de tipo OPC podrá conectarse con cualquier software cliente OPC, lo que permite que los sistemas basados en este estándar cuenten con una gran flexibilidad y conectividad, y la capacidad de añadir diferentes dispositivos a un software de adquisición de datos, gestión y supervisión sin necesidad de modificar todo lo que se ha desarrollado previamente, como se muestra en la figura 4.3.

Otra de las ventajas importantes asociada al uso de esta tecnología es el grado de reutilización del software: los componentes que se crean para un sistema pueden ser reutilizados en otros sistemas que se basen en la misma tecnología, de manera que se pueden conectar e integrar dispositivos heterogéneos. Desde este punto de vista, esta tecnología permite que los desarrollos se centran más en las funcionalidades de los componentes.

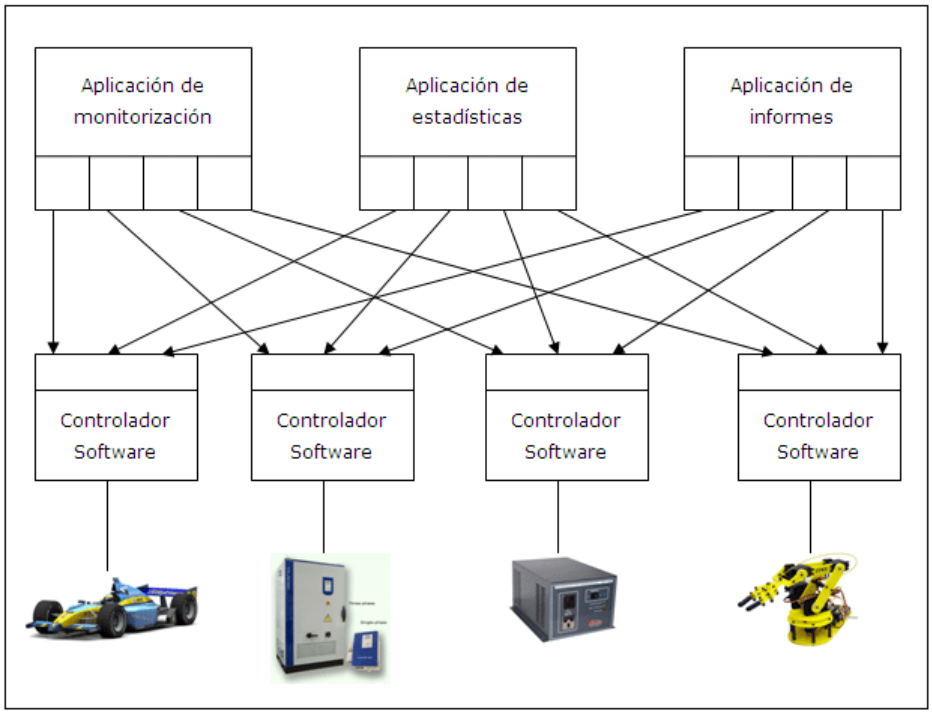

Figura 4.2: Problema de la interconexión entre multitud de dispositivos

OPC dispone de una serie de especificaciones estándares. El primer estándar (originalmente llamado *Especificación OPC* y ahora *Especificación de Acceso a Datos - Data Access Specification*) surgió como resultado de la colaboración que se estableció entre Microsft y algunas de las empresas más importantes en el desarrollo de sistemas de automatización. Las especificaciones definieron un estándar de objetos, interfaces y métodos basados basado en la tecnología de modelo de objeto (OLE COM) y el modelo de objectos distribuido (DCOM), para su uso en el proceso de control y monitorización en sistemas de automatización. Por ello, la tecnología COM/DCOM proporciona un marco de trabajo para el desarrollo de productos software. Las especificaciones originales sirvieron para la estandarización de los procesos de adquisición de datos. En la actualidad, y por las ventajas que supone, estas se han ido extendiendo también a las comunicaciones entre otro tipo de datos. Las actuales especificaciones OPC incluyen:

- OPC Data Access (OPC DA): esta especificación es usada para el acceso a datos en tiempo real desde cualquier dispositivo hacia o desde el software cliente u otros dispositivos de visualización. Proporciona capacidades avanzadas de inspección de items e incorpora la capacidad de trabajar con XML.
- OPC Alarmas y Eventos: se proporcionan notificaciones de alarmas y eventos bajo demanda (en lugar de estar continuamente recuperando y comprobando condiciones de los datos utilizando OPC DA). Incluye procesos de alarma, acciones, mensajes de información y seguimientos de eventos para su posterior estudio.
- OPC Batch: proporciona a la tecnología OPC la especialización para realizar procesos por lotes.

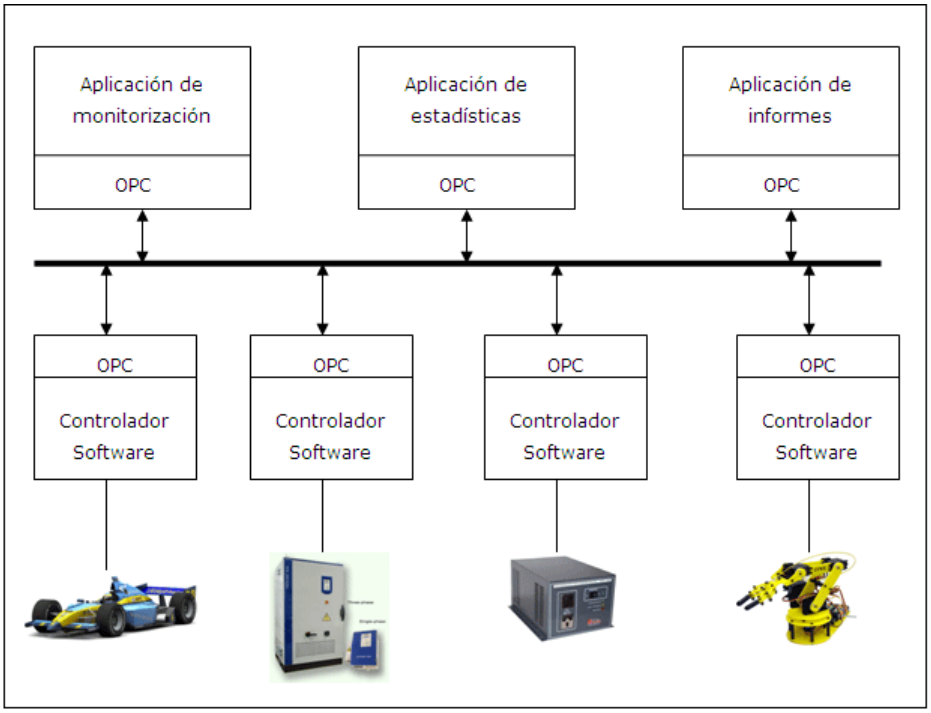

Figura 4.3: Solución al problema de la interconexión entre multitud de dispositivos

- OPC Data eXchange: esta especificación eleva la tecnología OPC cliente/servidor al paradigma servidor/servidor con comunicación a través de redes Field-Bus Ethernet, lo que proporciona interoperatividad entre diferentes distribuidores añadiendo además configuración remota y servicios de diagnostico y monitorización.

- OPC Historical Data Access (OPC HDA): proporciona el acceso a la información ya almacenada en los diferentes dispositivos. Permite establecer comunicaciones tanto a sistemas hardware, como puede ser un registrador de datos serie o a los archivos históricos de sistemas tipo SCADA. HDA puede recuperar dicha información de manera uniforme y homogénea.

- OPC Security: especifica como controlar los accesos de los clientes OPC a los servidores de manera que la información sensible esté protegida y evitar que se realicen modificaciones no autorizadas. Se protege la información proporcionada por los servidores OPC que es importante para los sistemas para que no sea actualizada de manera errónea, lo que podría tener consecuencias significativas para los procesos de la planta.

- OPC XML-DA: proporciona reglas flexibles y consistentes y formatos para la publicación de los datos de un sistema utilizando XML para su representación y SOAP o Servicios Web para su publicación.

- OPC Complex Data: es una especificación complementaria a OPC-DA y XML-DA que permite a los servidores publicar y describir tipos de datos más complejos como estructuras binarias y documentos XML.

- OPC Commands: conjunto de interfaces que permite a los servidores y clientes OPC identificar y enviar ordenes que los dispositivos podrían interpretar directamente.

En la figura 4.4 se muestra el diagrama de servidores OPC y clientes OPC y su integración en la arquitectura propuesta en el capítulo 3.

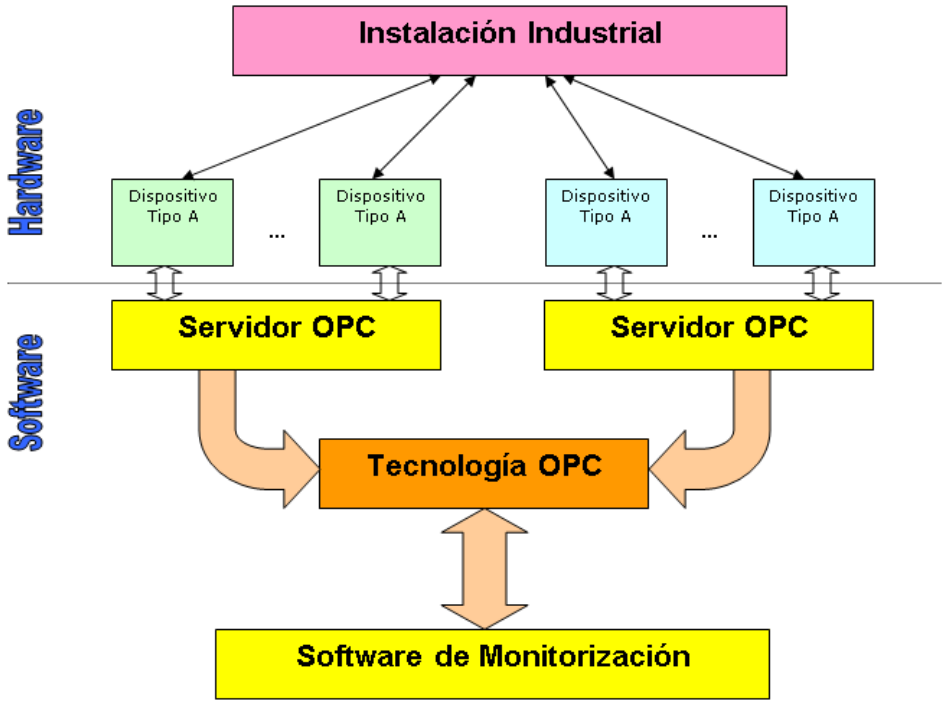

Figura 4.4: Diagrama de servidores OPC y clientes OPC y su integración en la arquitectura propuesta

Por otra parte se debe tener en cuenta la incorporación de los parámetros que intervienen en el sistema de comunicaciones, a saber:

- El tipo o modelo del dispositivo al que se conecta: contadores, distintos modelos de inversores y de distintos fabricantes, células calibradas, sensores de temperatura, registradores de datos, etc.
- El medio que se utiliza para conectarse a los dispositivos: GSM, RTB, IP, directo al puerto serie, etc.
- Cualquier otro tipo de parámetro necesario que tengan los dispositivos a conectar.

Teniendo esto en cuenta, se ha desarrollado también un mecanismo de configuración que genera un fichero de configuración con los datos específicos de cada planta, de manera que el mismo OPC pueda ser utilizando con distintos medios de conexión. Así, se consigue que los servidores OPC desarrollados para cada dispositivo pueden ser

utilizados en distintas plantas que tenga ese dispositivo. Para esto cada planta dispone de un fichero de configuración donde se detalla la forma de conectarse así como el número de teléfono, si fuese necesario, para comunicarse y demás parámetros para hacer funcionar correctamente al protocolo asociado con dicha instalación.

La principal separación entre los dispositivos físicos y el marco de trabajo desarrollado se realiza apoyándose en la tecnología OPC. Por una parte el marco desarrollado permite la comunicación con cualquier dispositivo que cumpla con el estándar OPC de comunicación y para ello cada dispositivo debe disponer de un servidor OPC o driver que encapsule los mecanismos y protocolos de intercambio de información para poder ser utilizado.

El marco de trabajo que se ha desarrollado permite la creación de manera rápida de servidores OPCm permitiendo de esta manera la integración de dispositivos aunque no se tengan dicho servidor, siempre que se disponga del protocolo o de alguna API suministrada por el fabricante. La creación de un servidor OPC se establece alrededor de una clase principal que permite disponer de toda la funcionalidad OPC. Esta clase principal proporciona al diseñador de la clase base de la que heredar para crear un servidor OPC para cada dispositivo.

Utilizando el estándar OPC se logra separar la parte de representación de los datos de la conexión e interoperabilidad de los sistemas físicos y lógicos de monitorización. Esto permite a su vez una gran independencia y generalidad, ya que cualquier dispositivo o sistema que utiliza este protocolo podrá intercambiar información con este sistema.

Figura 4.5: Capa de abstracción de comunicaciones y protocolos

En la figura 4.5 se muestra el esquema de la capa de comunicaciones para algunos de los dispositivos integrados en sistemas energéticos como ejemplo de aplicación de esta tecnología.

4.3. Marco de trabajo para la generación de servidores OPC-DA

El marco de trabajo propuesto dispone de todos los elementos necesarios para la construcción de aplicaciones de monitorización y gestión de sistemas energéticos. Una de las partes más complejas e importantes es la de la comunicación con todos los dispositivos de una planta, por lo que disponer de mecanismos para acelerar el desarrollo y disponer de una solución robusta, son características altamente deseables en este marco de trabajo. Como se ha comentado anteriormente, las comunicaciones con todos los dispositivos se realizarán utilizando el estándar OPC así que disponer de los controladores o servidores OPC que se comunican con cada dispositivo es necesario si se quiere intercambiar o recuperar información de los mismos.

En la mayoría de casos, los fabricantes de los dispositivos no incorporan aún servidores OPC para sus productos por lo que disponer de mecanismos para poder desarrollarlos de manera rápida y sencilla permitirá acelerar el desarrollo de la aplicación así como integrar dichos dispositivos dentro del sistema que se está implementando.

Durante la creación de este marco de trabajo y para su inmediata aplicabilidad ha sido necesario desarrollar varios servidores OPC para facilitar la comunicación de los dispositivos con los sistemas desarrollados. En muchos casos no era posible disponer del protocolo nativo o incluso el dispositivo no contaba con una forma oficial de comunicarse por lo que ha sido necesario recurrir a técnicas de ingeniería inversa o la utilización de espías de protocolos para conocer la forma nativa de comunicación. Una vez conocido o desarrollado el protocolo, este ha sido incorporado y encapsulado dentro de una jerarquía de clases para envolver dicho protocolo con las funcionalidades que lo harán convertirse en un servidor OPC independiente y listo para ser incorporado al sistema.

4.3.1. Servidores OPC-DA

Un servidor OPC-DA es una servidor que proporciona datos instantáneos de un dispositivo. Así, un cliente OPC puede comunicarse con un servidor OPC-DA para interrogar sobre los canales o medidas en tiempo real. Durante la instalación de un servidor OPC, este se registra dentro del sistema y así se permite que cualquier cliente OPC puede preguntar al sistema operativo qué servidores OPC están disponibles. Una vez seleccionado, el servidor OPC arranca y comienza por un lado a intercambiar datos con el dispositivo y por otro lado, le pasa estos datos al cliente OPC a una frecuencia determinada por el propio cliente.

En la figura 4.6 puede verse el diagrama de secuencia típico entre un sistema de monitorización y un dispositivo utilizando un cliente y un servidor OPC-DA.

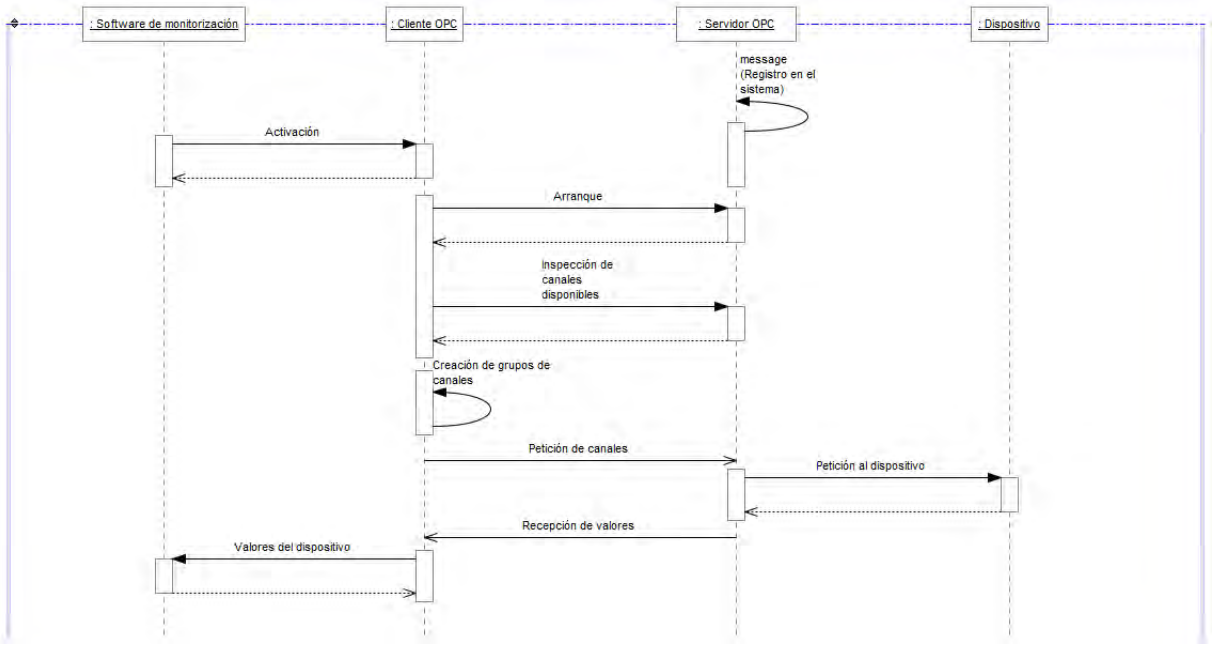

Figura 4.6: Diagrama de secuencia entre un sistema de monitorización y un dispositivo a través de un cliente OPC-DA y un servidor OPC-DA

Es interesante observar que cuando el cliente OPC crea los grupos de medidas que quiere recuperar de un dispositivo podría establecer una frecuencia de recogida diferente a la que el servidor OPC puede recuperar datos del dispositivo debido a diferencias o particularidades del protocolo.

Toda la lógica entre el servidor OPC y cliente OPC es genérica para cualquier servidor OPC por lo que es apropiado encapsularla en una serie de clases para permitir, de manera sencilla, reutilizar toda esta lógica común. Las particularidades, como el protocolo de comunicación con el dispositivo así como los interfaces de usuario necesarios para parametrizar los diferentes servidores OPC a desarrollar es justamente lo que cada clase final de la jerarquía debe implementar, como se muestra en la figura 4.7.

Cuando se desarrolla un servidor OPC para un determinado dispositivo se crea una clase que herede de la jerarquía apropiada. Dicha clase encapsulará el protocolo de intercambio de información con el dispositivo y serán las clases anfitrionas las que se encargarán de proporcional el interfaz de comportamiento de servidor OPC (por ejemplo, las clases marcadas en amarillo intenso en la figura 4.7). Además es posible encapsular información determinada sobre el comportamiento de los dispositivos heredando a su vez de la clase TDevices en el caso que se desee modelizar cualquier particularidad del mismo.

La mayoría de la lógica encargada de tratar con el protocolo y los dispositivos es la clase de la jerarquía denominada TCustomDeviceOPCServer, que se muestra en la figura 4.9. Esta clase dispone de la lista de dispositivos a controlar, los eventos que pueden ser sobreescritos por las clases inferiores para la gestión del ciclo de recogida

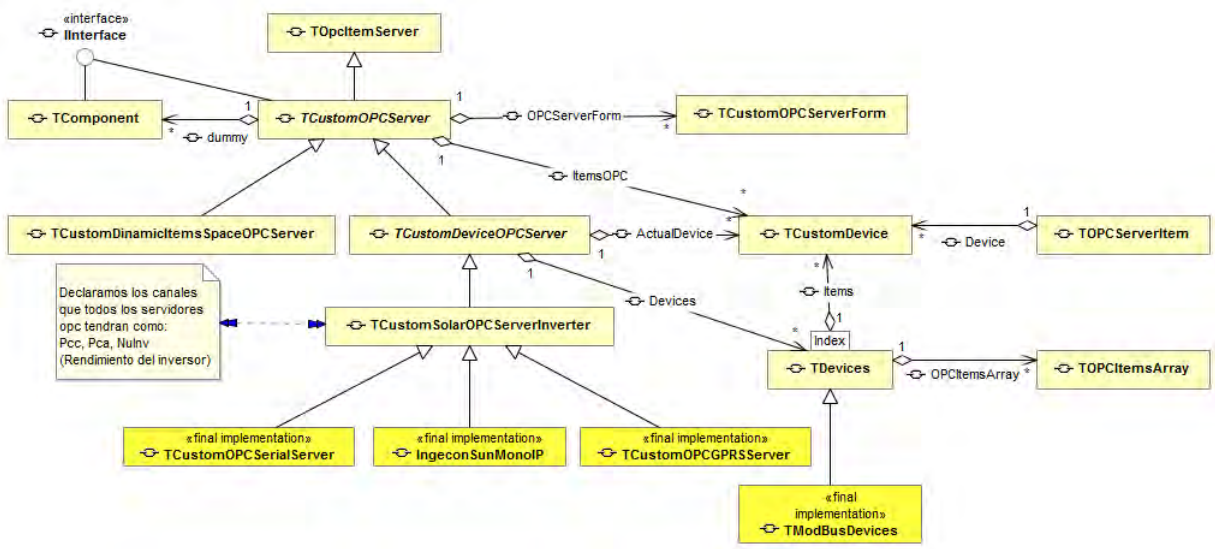

Figura 4.7: Jerarquía para la construcción de servidores OPC-DA

de datos (OnDeviceOK, OnDeviceRetry, etc...), las funciones que devuelven las clases que deben ser utilizadas para la creación de las instancias apropiadas utilizando para ello una factoría de clases (GetDeviceClass, GetDevicesClass y GetConfiguraDeviceFormClass) y una hebra, RoundTimer, que se encargará en cada ciclo de ejecución establecido de preguntar a cada dispositivo por los valores de los puntos de medida que dispone. Esta última información se almacena internamente y puede ser recogida por medio del protocolo estándar por los clientes OPC usando COM+/DCOM [Gor01], [Har01].

Es importante cuando se despliegan múltiples aplicaciones, y sobre todo cuando forman parte de un mismo sistema, que todas dispongan de una apariencia uniforme y parecida dando así impresión de coherencia y facilidad de uso. Para evitar que esta responsabilidad recaiga en la parte de diseño, se ha acoplado a la jerarquía de clases una serie de elementos visuales comunes permitiendo de esta manera que todas las clases compartan los mismos elementos gráficos, según se muestra en la figura 4.10.

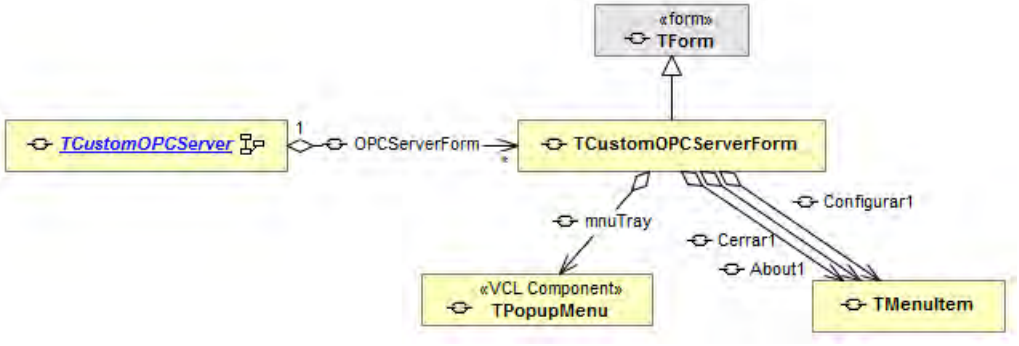

Figura 4.10: Formulario visual de un servidor OPC

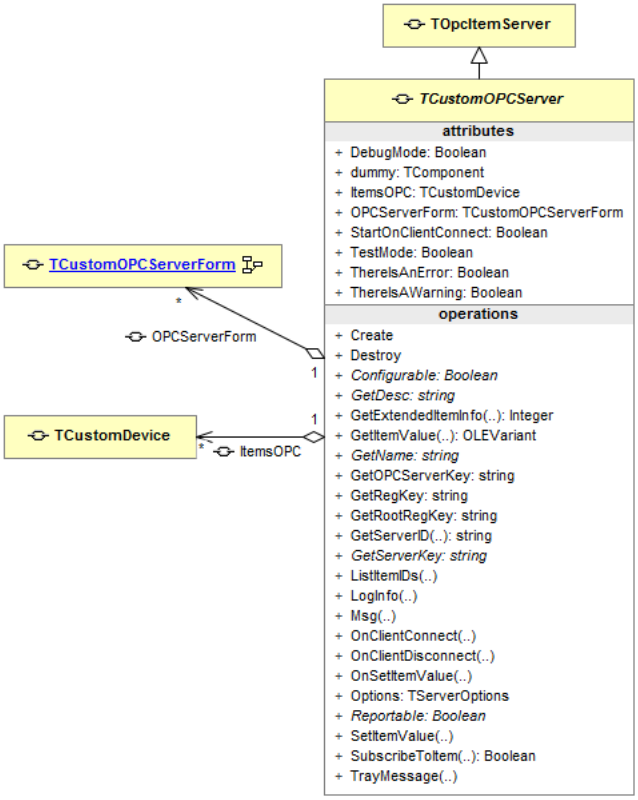

Figura 4.8: Clase encargada de proporcionar el comportamiento de servidor OPC

Así cualquier servidor OPC desarrollado dispondrá de un formulario con un mismo aspecto visual así como elementos de menú con las acciones y funcionalidades comunes, como puede observarse en la figura 4.11.

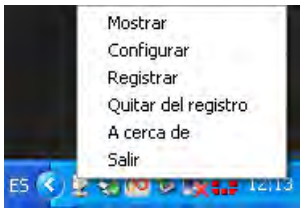

Figura 4.11: Sistema de acciones comunes de cualquier servidor OPC

4.3.2. Servidor OPC-DA para inversores en plantas solares

Uno de los subsistemas fundamentales en una instalación de energía solar de tipo fotovoltaico son los inversores que se encargan de transformar la corriente continua, que proviene de las placas solares, en corriente alterna para que pueda ser usada de manera directa o para que pueda inyectarse a la red. Los inversores disponibles en el mercado pueden ser bastantes diferentes en algunas de sus características como puede ser los puertos de comunicación, la potencia, características físicas, etc.. es por ello que

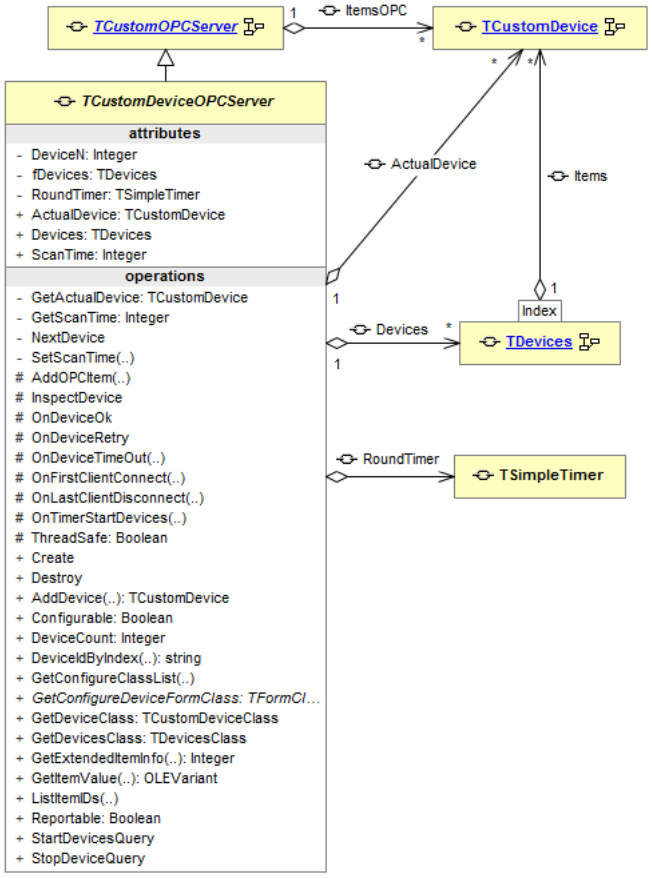

Figura 4.9: Clase encargada de proporcionar la comunicación entre los dispositivos y el interfaz OPC

el marco de trabajo debe disponer de la capacidad de permitir indicar dichas diferencias utilizando los mecanismos de particularización. Es cierto que entre todos ellos existe similitudes y que se pueden aprovechar los mecanismos de herencia para reunir en las clases superiores de la jerarquía de clases los parámetros comunes que tendrán todos los inversores desarrollados con el consiguiente ahorro de código (Fig. 4.12).

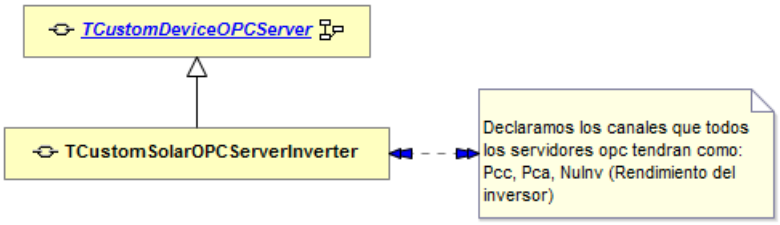

Figura 4.12: Generalización de la clase común de inversor solar

Así, en la clase TCustomSolarOPCServerInverter se declaran los canales que disponen todos los inversores, permitiendo así que los inversores que se utilicen dispongan ya de esos canales y facilitar así la creación de los mismos.

Listing 4.1: Declaración en el constructor

```
1  constructor TCustomSolarOPCServerInverter.Create;
2  begin
3
4    inherited;
5
6    // Canales Obligatorios
7    AddOPCItem(MS_PCC,[iaRead],varDouble or varArray,0,nil);
8    AddOPCItem(MS_PCA,[iaRead],varDouble or varArray,0,nil);
9    AddOPCItem(MS_NU ,[iaRead],varDouble or varArray,0,nil);
10
11   // Estado del inversor
12   AddOPCItem(ST_ENCENDIDO,[iaRead],varBoolean,0,nil,OPC_QUALITY_GOOD);
13   AddOPCItem(ST_MARCHA ,[iaRead],varBoolean,0,nil,OPC_QUALITY_GOOD);
14 end;
```

En el código correspondiente al constructor de la clase que proporciona la funcionalidad genérica para disponer de un servidor OPC para inversores solares puede verse la declaración de los canales que compartirán todos los servidores de este tipo. En este caso canales de *potencia continua y alterna y rendimiento del inversor* así como algunas señales de alarmas.

En los siguientes apartados se describen algunos de los servidores OPC desarrollados para inversores de distintos tipos utilizando el marco de trabajo propuesto y que ponen a prueba su validez.

Servidor OPC-DA para inversores monofásicos con protocolo Modbus

De entre los inversores monofásicos, uno de los protocolos de comunicación más utilizado con los inversores es el basado en ModBus. Debido a esta razón hay multitud de instalaciones que utilizan este tipo de dispositivos por lo que son unos candidatos apropiados a los que desarrollar el servidor OPC, utilizando el marco de trabajo estudio de esta tesis, para integrarlos dentro de los sistemas desarrollados.

Para el modelizado de este inversor ha sido necesario especializar la clase que implementa la lógica de un inversor solar genérico, según se muestra en la figura 4.13.

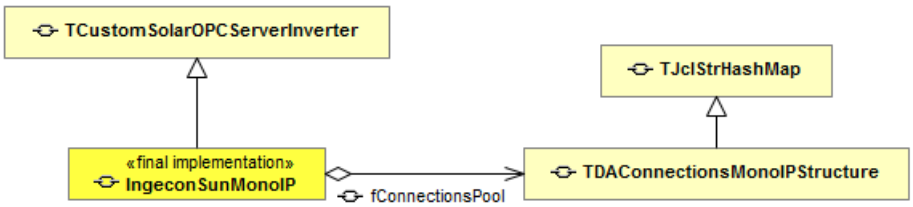

Figura 4.13: Implementación de un servidor OPC para un inversor monofásico

A continuación se muestran algunas implementaciones disponibles en el marco de trabajo para la comunicación con dispositivos basados en el protocolo ModBus pero para diferentes medios físicos de comunicación.

Servidor OPC-DA para inversor trifásico con protocolo Modbus y comunicación a través de GSM

El modelizado de un inversor trifásico con comunicaciones serie se ha realizado partiendo de la clase genérica que permite conectar con cualquier inversor solar utilizando el puerto serie y comunicación telefónica GSM, de acuerdo con la figura 4.14.

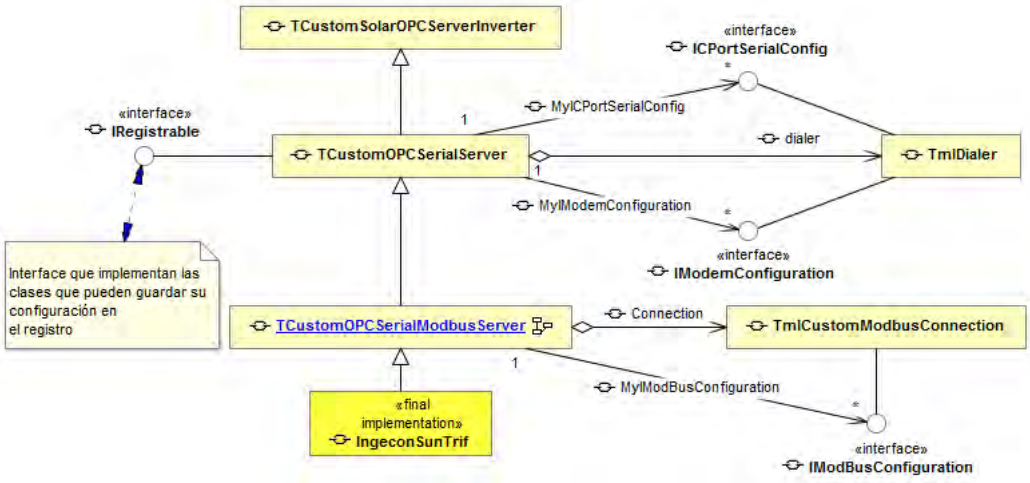

Figura 4.14: Implementación de un servidor OPC con comunicación GSM a través de ModBus

Puede observarse como la clase anfitriona TCustomOPCSerialServer dispone de una referencia a un componente denominado TDialer. Este componente implementa dos interfaces que el marco de trabajo utilizará para configurar dicho componente utilizando un interfaz de usuario apropiado. Por un lado ICPortSerialConfig que configura los parámetros del puerto serie, figura 4.15 (izquierda), y IModemConfiguration que configura los parámetros de la comunicación telefónica utilizando el módem, figura 4.15(derecha).

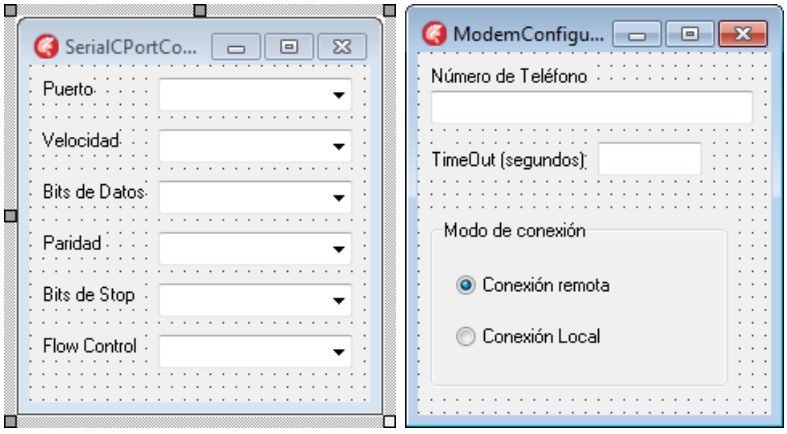

Figura 4.15: Implementación del interfaz para la configuración del puerto serie (izquierda) y módem (derecha)

Una vez declarada la estructura utilizando las clases que proporciona el marco de trabajo desarrollado se dispondrá de una comunicación telefónica con cualquier inversor trifásico utilizando el protocolo OPC.

Puede además apreciarse cómo la conexión lógica con el inversor, es decir, la implementación del protocolo, se ha encapsulado dentro de la clase TmlCustomModbusConnection donde se realizará el intercambio de información utilizando el protocolo Modbus.

Servidor OPC-DA para inversor trifásico con protocolo Modbus y a través de GPRS

Gracias al avance de las redes de datos y las comunicaciones móviles, es cada vez más común interconectar dispositivos utilizando la red de Internet. Para este tipo de comunicación, la manera más generalizada es la utilización de la capa de red IP. Existe por tanto la capacidad de conectar utilizando el servicio general de paquetes vía radio o GPRS cuya comunicación interna está basada en tecnología IP.

En este caso, y para la creación del servidor OPC para este tipo de dispositivos se ha utilizado la clase TCustomOPCGPRSServer que proporciona la conectividad GPRS y la clase TCustomOPCGPRSModbusServer que proporciona el funcionamiento del protocolo Modbus a través de GPRS, figura 4.16.

Puede apreciarse que la implementación se basa en la reutilización de la interfaz que implementa el protocolo ModBus y extiende el modelo para la conexión GPRS.

Para la configuración de este servidor OPC será necesario al menos indicar la dirección IP del dispositivo y el marco de trabajo permitirá indicarlo de manera gráfica haciendo uso de la interfaz IPlantWithGPRSConfiguration, figura 4.17.

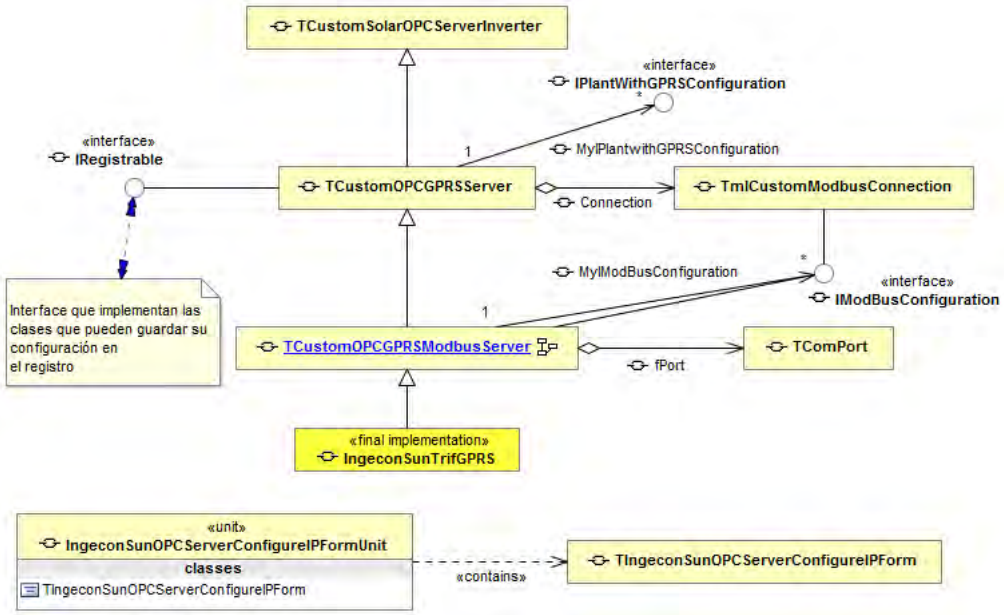

Figura 4.16: Implementación de un servidor OPC para inversor con comunicación GPRS

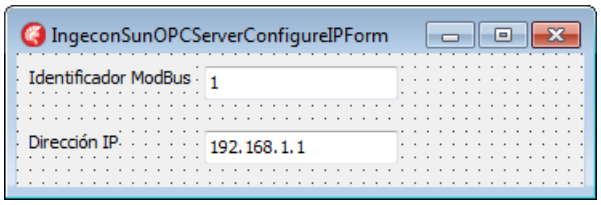

Figura 4.17: Implementación del interfaz para la configuración del puerto serie

4.3.3. Servidor OPC-DA para inversores con protocolo propietario

Para la implementación de un servidor OPC para inversores que tienen un protocolo propietario se ha seguido la misma filosofía de desarrollo para continuar demostrando la facilidad con la que se puede disponer de un servidor OPC para cualquier tipo de inversor.

Al ser un protocolo propietario, es decir, no se basa en ningún protocolo estándar de intercambio de información, la implementación del mismo se realizará en una clase específica y heredará el comportamiento de inversor genérico que disponga de un puerto serie, figura 4.18.

En este caso, la clase superior implementa el comportamiento de una servidor OPC para inversores con un puerto serie de comunicaciones así como el interfaz de usuario para configurar dicho puerto serie.

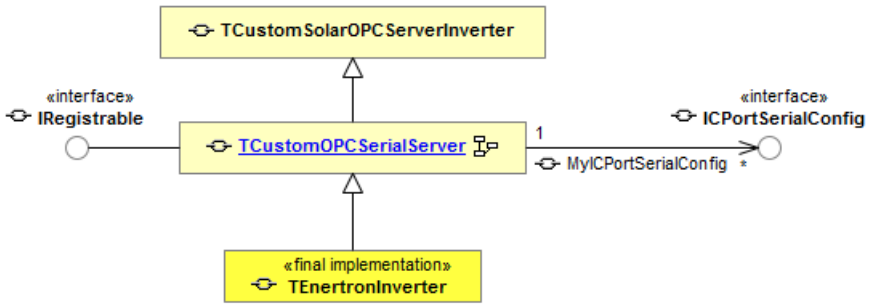

Figura 4.18: Implementación del interfaz para un protocolo propietario

4.3.4. Servidor OPC-DA para inversores con protocolo SNMP

Existen inversores cuyo protocolo de comunicación están basados en SNMP (Simple Network Management Protocol). En este caso es necesario instalar un agente que reporta la información a través de SNMP con el sistema de recogida de información. Es por ello, que en este caso, el servidor OPC deberá actuar como sistema escucha de la información que el agente emite sobre el funcionamiento del inversor utilizando tramas SNMP.

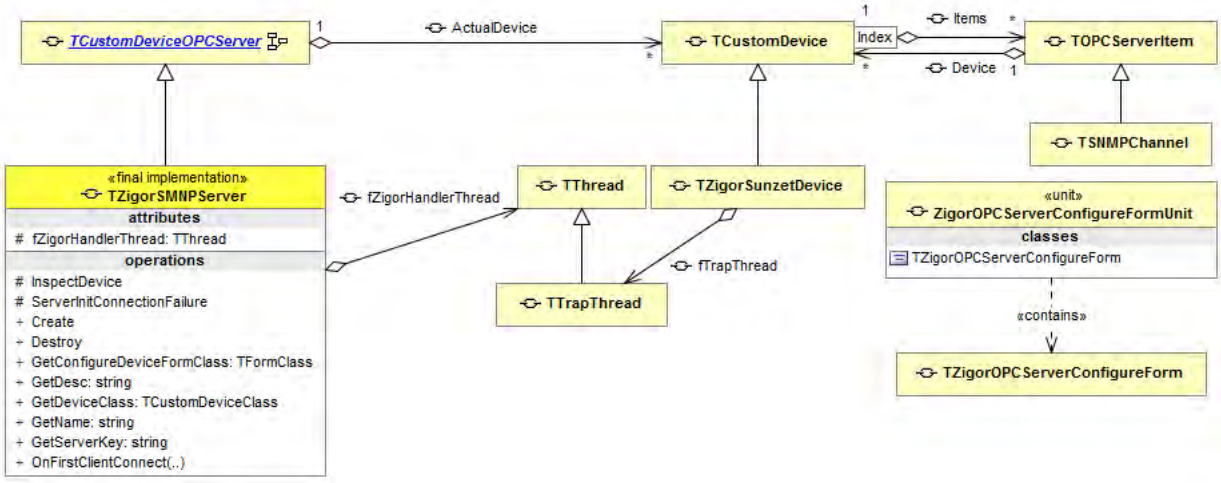

Figura 4.19: Implementación de un servidor OPC para inversores con protocolo basado en SNMP

La clase que implementa la particularización de un servidor OPC genérico en un servidor OPC, indicará además el formulario de configuración necesario para hacer funcionar el protocolo SNMP que implementa. Entre estos parámetros se encuentra la dirección IP donde se encuentra el agente que intercambia información con el inversor, la información SNMP y la identificación del dispositivo que se asocia con el inversor, figura 4.20.

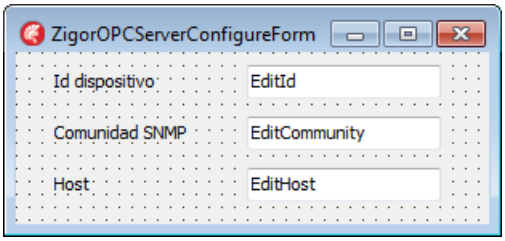

Figura 4.20: Formulario de configuración de los parámetros SNMP para servidor OPC

4.3.5. Servidor OPC-DA para Arduino

En los últimos años ha aparecido un nuevo paradigma en la fabricación y utilización de dispositivos basada en hardware abierto. Uno de sus máximos exponentes son los dispositivos basados en una placa con un microprocesador y entorno de desarrollo libre con aplicación en proyectos multidisciplinares.

El hardware está basado en un microcontrolador Atmel AVR y su fácil utilización y bajo coste ha supuesto una autentica revolución entre los aficionados y profesionales del mundo de la electrónica e incluso entornos educativos.

Aprovechando el desarrollo de este marco de trabajo y la total ausencia de ningún mecanismo de comunicaciones estándar basado en OPC, se ha desarrollado un servidor OPC que ha permitido que este dispositivo tenga esta funcionalidad y se ha liberado este producto que está siendo utilizado por miles de usuarios y ha permitido formar una comunidad alrededor de este producto.

Actualmente es el único producto de referencia que aúna la tecnología basada en *Hardware Libre* y la tecnología OPC y ha sido desarrollado con el marco de trabajo desarrollado en este tesis.

Los dispositivos de esta familia están representados por varios modelos con diferentes características y desde el principio se ha tenido en mente cubrir todos y cada uno de los modelos más emblemáticos. La multitud de variantes y formas de comunicación hacían de este un proyecto complejo pero apropiado para ser implementado siguiendo la filosofía y la arquitectura propuesta. Se han considerado los tres siguientes modelos:

- El modelo **Arduino UNO** es el modelo básico e implementa en su interior un microcontrolador Atmel AVR ATmega328 a 16MHZ funcionando a 5V con 14 entradas/salidas digitales y 6 entradas/salidas analógicas disponiendo de una memoria flash de 32Kb. La comunicación con el ordenador se realiza a través de comunicaciones serie materializadas sobre una línea USB.
- El modelo **Arduino Ethernet** es similar al Arduino UNO pero dispone un puerto de comunicaciones Ethernet proporcionando mayor potencia de comunicación al poder conectarlo a una red local y aumentado considerablemente el alcance de trabajo así como la velocidad de comunicaciones.

- El modelo **Arduino YÚN** es considerado de mayor gama al disponer en su interior de un sistema Linux basado en OpenWrt denominado OpenWrt-Yun. Dispone en la placa de un puerto Ethernet y soporte para Wifi, un puerto USB-A así como un conector para tarjeta micro-SD.

Implementación del servidor OPC-DA para Arduino

Uno de los requisitos fundamentales que se han buscado a la hora de desarrollar este servidor OPC es la de no desarrollarlo para un dispositivo determinado sino proporcionar la posibilidad de que desde un único servidor OPC poder controlar cualquier número y cualquier tipo de dispositivos Arduino. Así, sería posible con un único servidor OPC permitir que cualquier SCADA o sistema de monitorización controlase varios Arduinos, cada uno de un tipo diferente y por un protocolo físico de comunicaciones diferente.

Para ello se ha partido de la jerarquía de clases mostrada en la figura 4.21 y forma parte del marco de trabajo desarrollado en esta tesis.

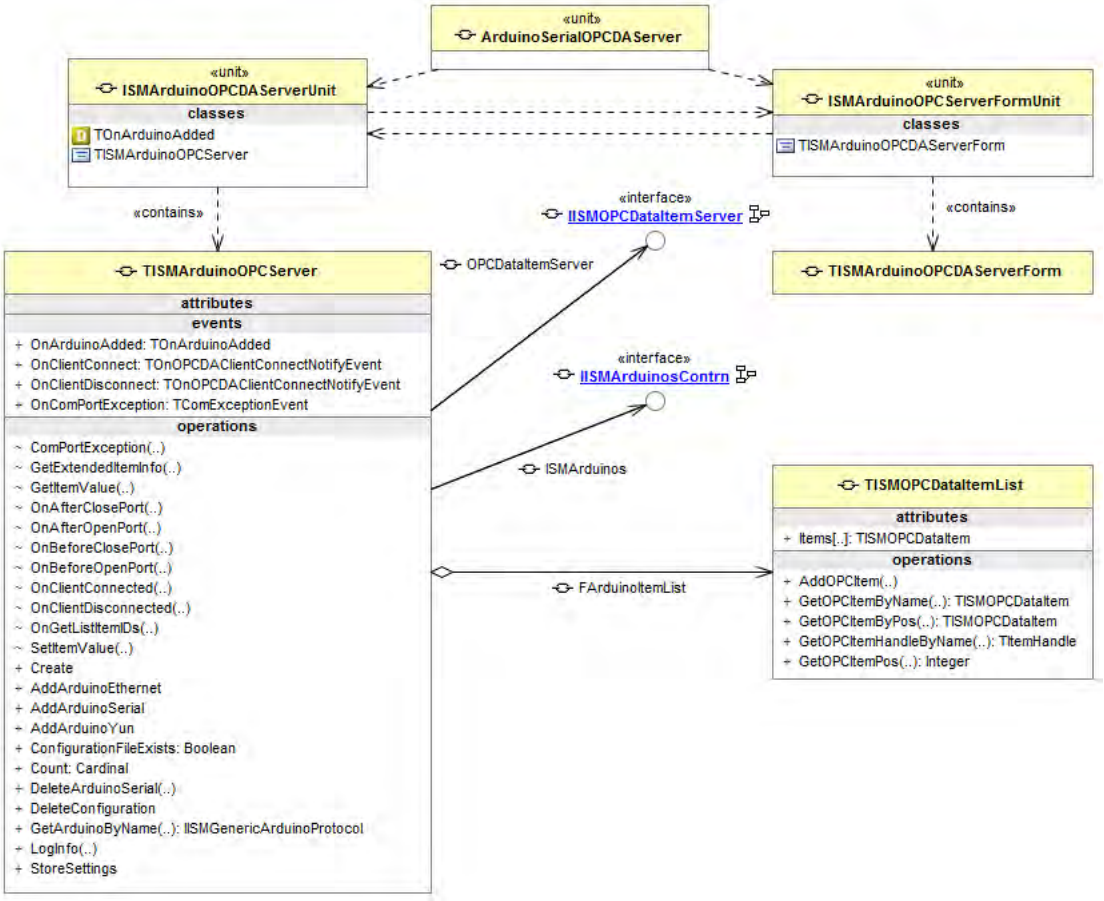

Figura 4.21: Jerarquía de implementación

La clase central será TISMArduinoOPCServer y tendrá una referencia al formulario

visual del servidor y a un contenedor con todos los Arduinos con los que se podrá conectar así como sus configuraciones. Hay que recordar que para el Arduino UNO se necesitan indicar los parámetros de configuración del puerto de comunicaciones serie mientras que, por ejemplo, para el Arduino Ethernet y Arduino YÚN, las direcciones IP de la red en la que están conectados.

La implementación del contenedor de Arduinos describe la gestión de las instancias que se deben crear para la comunicación con cada dispositivo (Fig. 4.22). Así se tendrán los métodos apropiados para añadir o eliminar Arduinos dentro del sistema así como la lista de instancias para poder comunicarse con cada uno de ellos.

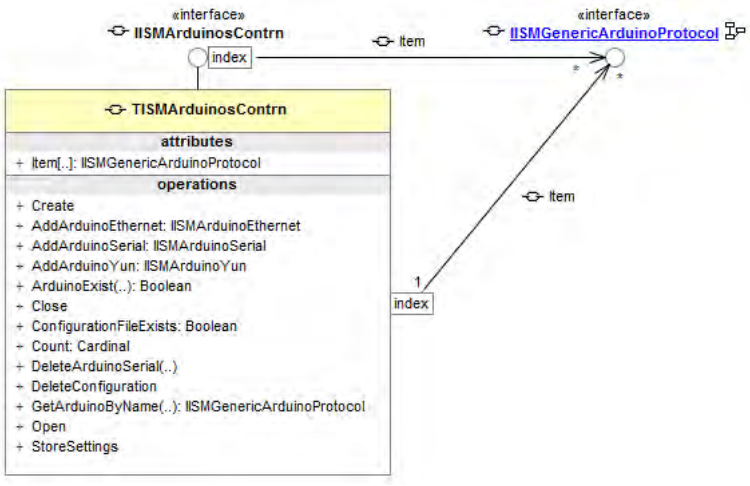

Figura 4.22: Contenedor de Arduinos

Esta lista se corresponde con una lista de objectos que deberá implementar un interfaz que corresponda con el protocolo apropiado para cada Arduino: el Arduino UNO con un protocolo de comunicaciones serie y los Arduino Ethernet y Arduino YÚN con protocolos de comuniaciones IP, figura 4.23.

Con esta propuesta se consigue encapsular cada particularidad de cada protocolo heredando e implementando de un protocolo genérico que será con el que interactuará la clase principal de la jerarquía.

El servidor OPC desarrollado necesita de un mecanismo de comunicación genérico que se encuentre implantado dentro del Arduino para poder comunicarse con él. Es por esto que se ha desarrollado una librería genérica que es compatible con cualquier modelo de Arduino y permite una comunicación hacia el exterior de manera unificada. De esta manera, el servidor OPC puede comunicarse con cualquier Arduino de una única manera y luego este servidor delegar esa comunicación al estándar OPC, figura 4.24.

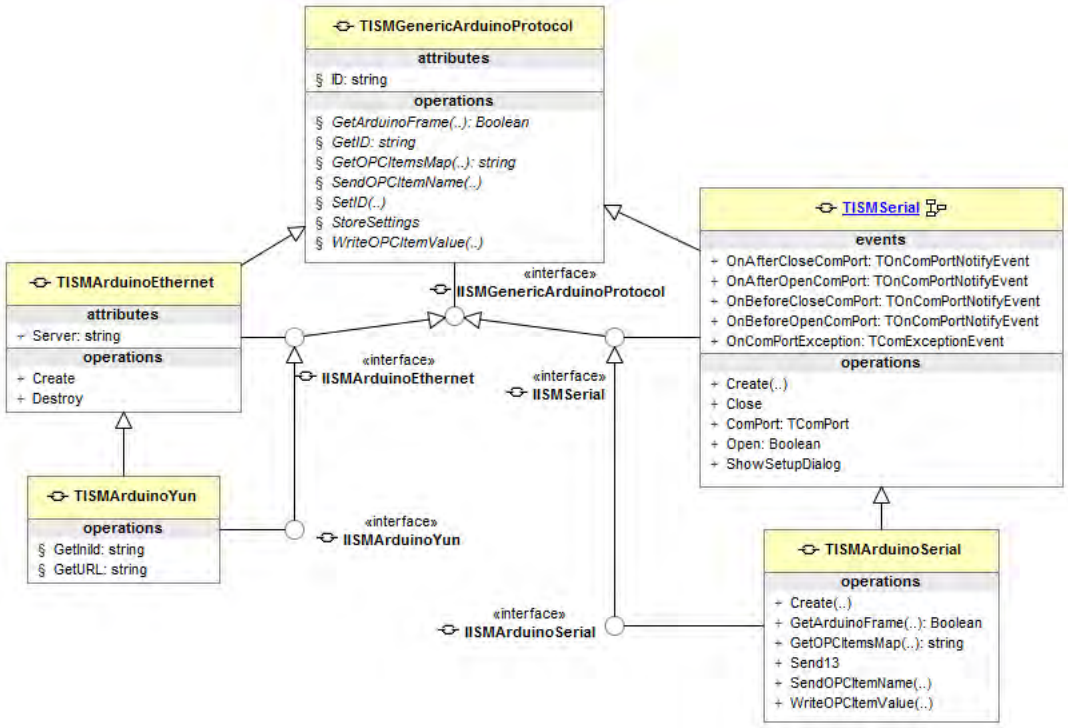

Figura 4.23: Jerarquía de protocolos para cada Arduino

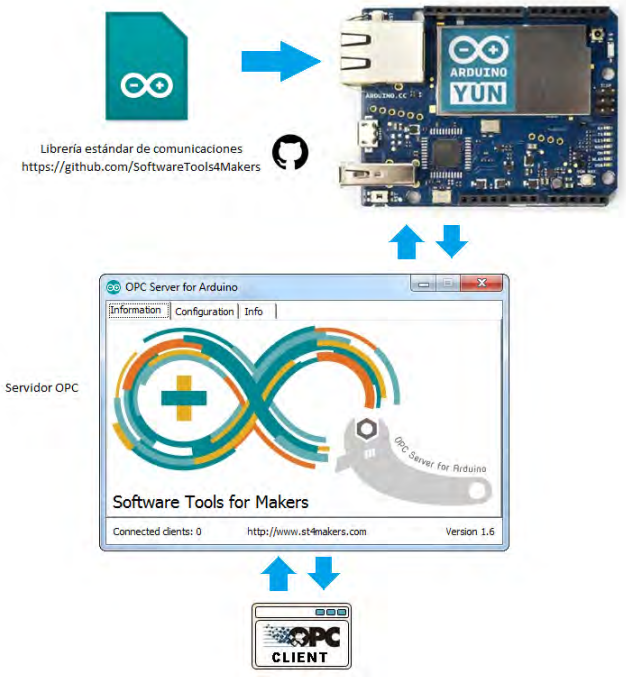

Figura 4.24: Esquema general del servidor OPC para Arduino

Una vez que el Arduino, a través de la librería genérica de comunicaciones y a su vez, a través del servidor OPC desarrollado, puede comunicarse de manera uniforme, este

servidor permite que cualquier sistema SCADA o cualquier sistema desarrollado con el software modelizado en este trabajo pueda intercambiar información con el mismo.

En la figura 4.25 puede verse el servidor OPC para Arduino desarrollado utilizando el marco de trabajo propuesto.

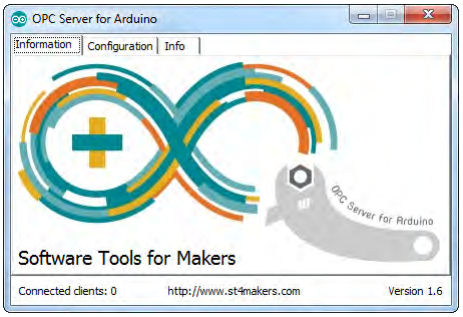

Figura 4.25: OPC Server para Arduino

Este servidor se instala en el sistema operativo cuando se ejecuta por primera vez anotando en el registro la información de su localización en disco y el resto de información para ser localizado (Fig. 4.26).

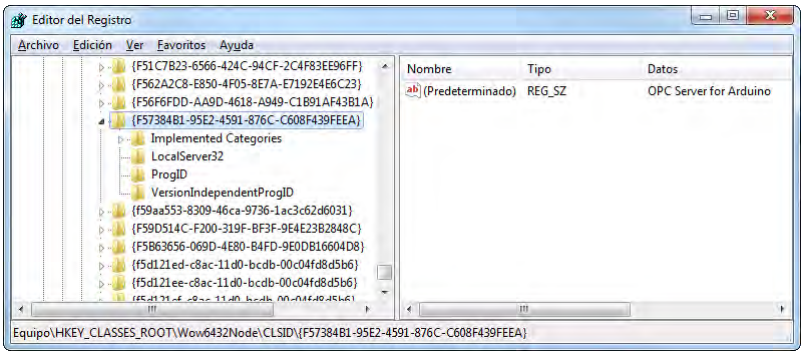

Figura 4.26: OPC Server para Arduino registrado en el sistema operativo

De esta manera, utilizando cualquier cliente OPC, se tendrá la posibilidad de preguntar por los servidores OPC instalados en el sistema y utilizarlos para intercambiar información con los dispositivos asociados a dicho servidor OPC, 4.27.

El simple hecho de utilizar un servidor OPC no implica disponer de manera directa de la información contenida en los dispositivos reales sin antes configurarlo. Es decir, el servidor OPC es simplemente un mecanismo de abstracción de comunicación por lo que será necesario indicarle con qué dispositivo y qué parámetros son necesarios para comunicarse con los dispositivos reales. En la figura 4.28 se muestra cómo se indican los parámetros de comunicación con un Arduino utilizando comunicaciones serie; se indican el puerto, la velocidad y diversos parámetros.

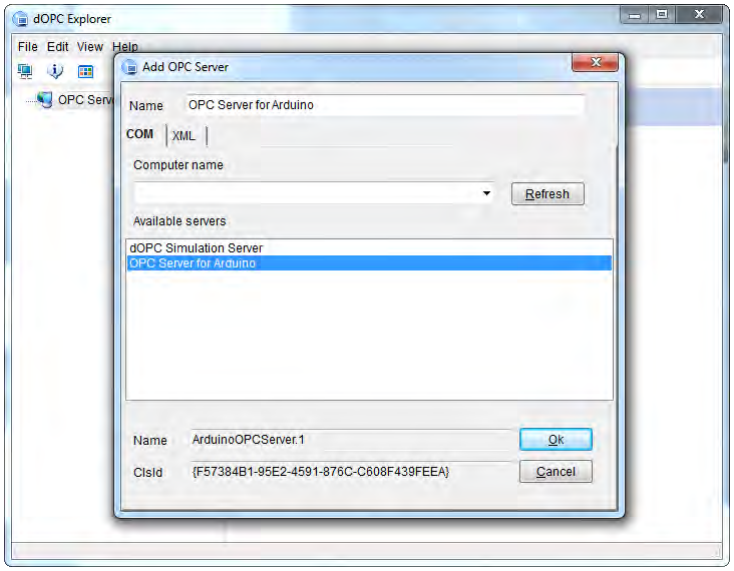

Figura 4.27: Cliente OPC utilizando un servidor OPC

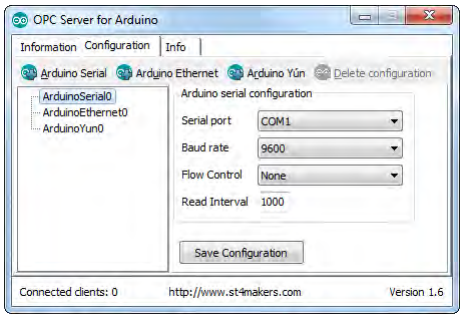

Figura 4.28: Configuración del OPC Server para Arduino Serie

En el caso de la comunicación con un Arduino y un medio de comunicaciones ethernet será necesario indicar su dirección IP, figura 4.29, de la misma manera que al utilizar un Arduino a través de una red Wifi (Fig. 4.30).

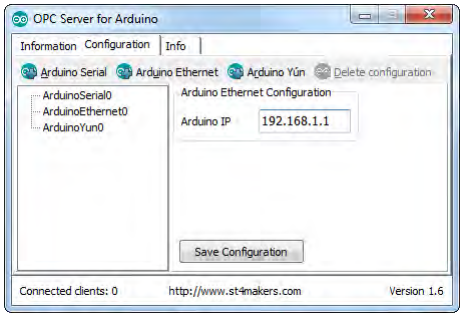

Figura 4.29: Configuración del OPC Server para Arduino Ethernet

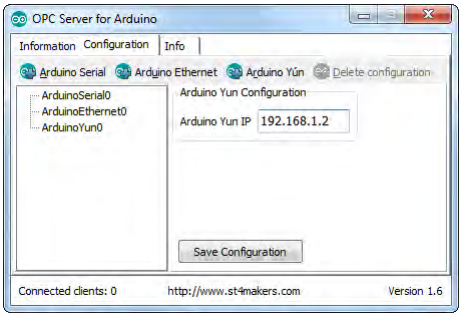

Figura 4.30: Configuración del OPC Server para Arduino YÚN

En la figura 4.31 se muestra el servidor OPC para Arduino, desarrollado con el marco de trabajo aquí descrito, interactuando con un sistema SCADA industrial comercial.

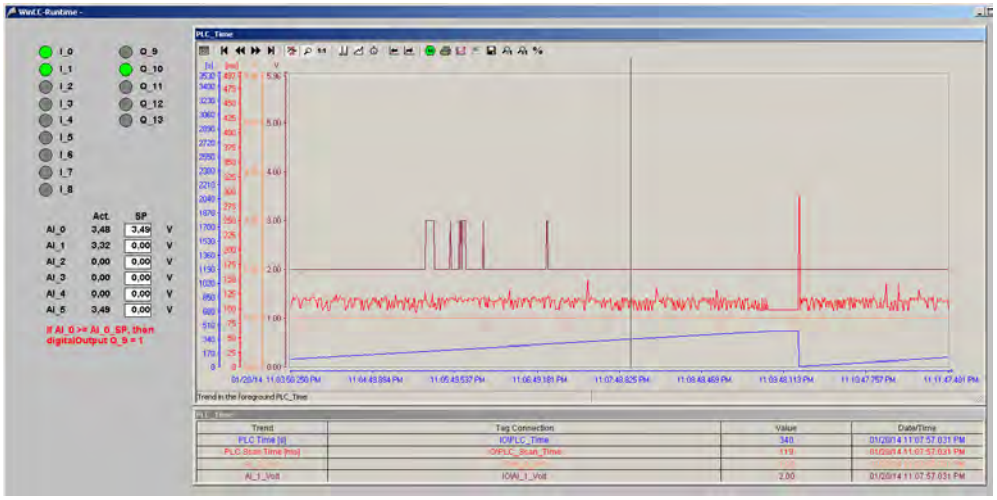

Figura 4.31: Utilización del servidor OPC para Arduino y el SCADA Wincc

4.4. Marco de trabajo para la generación de servidores OPC-HDA

En la sección anterior se ha mostrado cómo la tecnología OPC y la parte del marco de trabajo desarrollado, encargado de la construcción de servidores OPC-DA, permite desarrollar controladores compatibles con multitud de dispositivos en poco tiempo y de una manera sencilla. La información que se intercambia con estos dispositivos son señales instantáneas o en tiempo real. Sin embargo, en multitud de ocasiones, los dispositivos que forman parte de una instalación, almacenan en su interior valores denominados históricos. Estos dispositivos suelen tener una memoria interna y almacenan el valor de sus canales a una frecuencia determinada. Así, para acceder a ellos es necesario utilizar el protocolo determinado por el fabricante e inspeccionar la memoria de estos dispositivos para descargar la información.

En este caso la situación es la misma que cuando se necesita disponer de la información almacenada de diversos dispositivos en los que cada uno almacena la información de una manera determinada como puede verse en la figura 4.32.

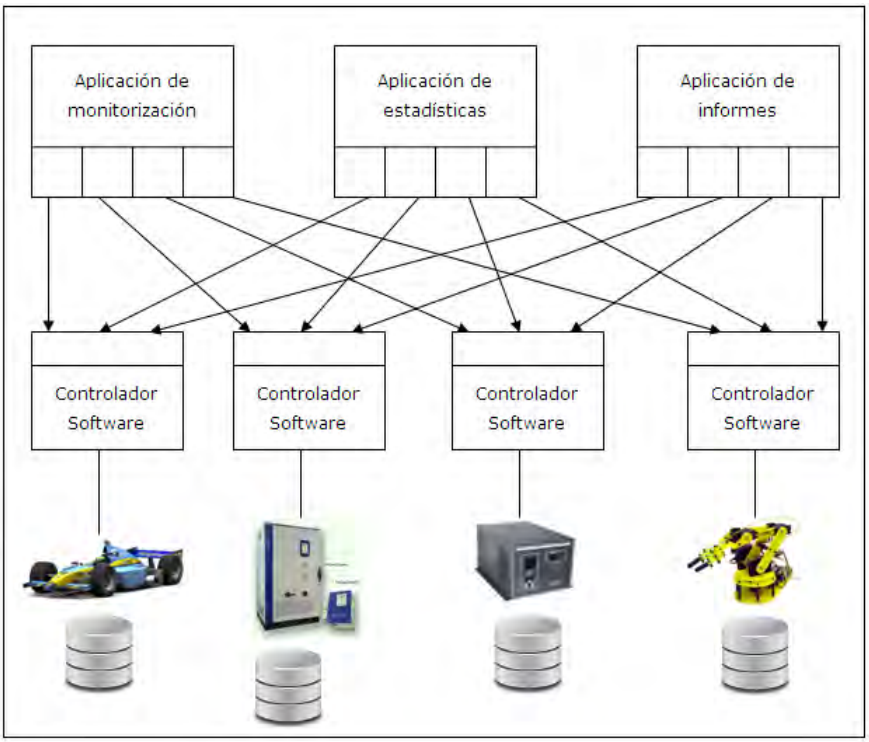

Figura 4.32: Problema de la interconexión entre multitud de dispositivos para acceder a información almacenada

La utilización de OPC-HDA permite disponer de un mecanismo de unificación de acceso a la información independientemente del formato en el que se encuentre almacenada la información dentro del dispositivo. Al igual que con la parte OPC-DA, solo será necesario conocer el protocolo para la descarga de la información y la exposi-

ción hacia cualquier cliente interesado en esos datos se realizará utilizando una forma homogénea basada en el protocolo OPC-HDA, figura 4.33.

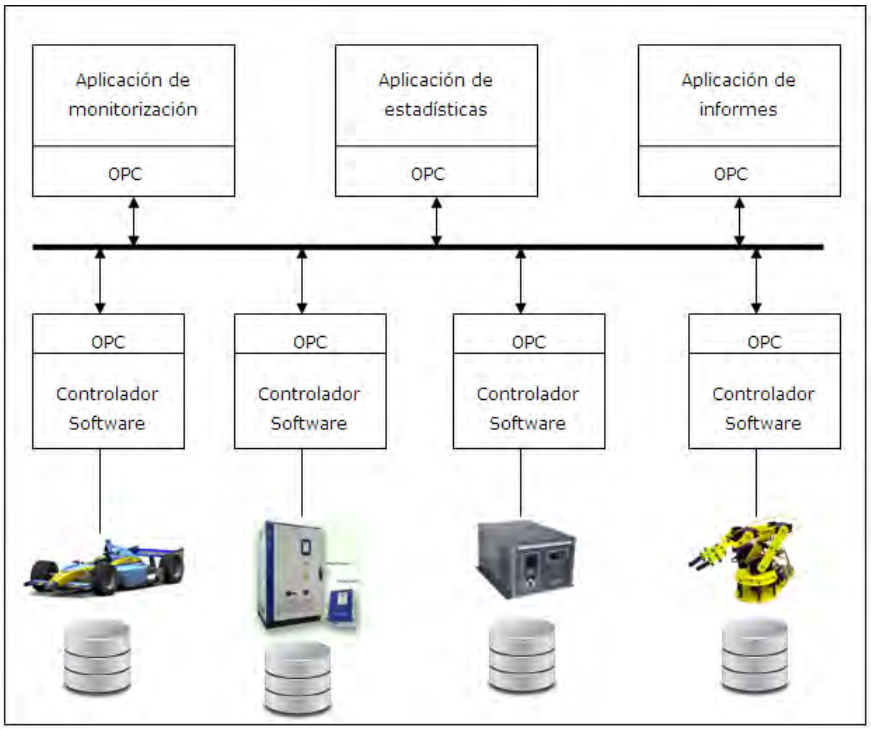

Figura 4.33: Utilización de OPC-HDA para unificar el acceso a la información almacenada

4.4.1. Servidores OPC-HDA

Un servidor OPC-HDA es un componente basado en COM/DCOM que implementa el interfaz OPC-HDA. Esta interfaz está basada en el modelo que se muestra en la figura 4.34.

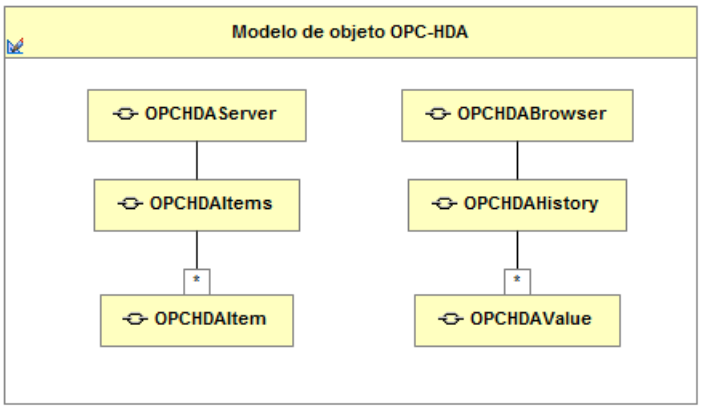

Figura 4.34: Modelo de objeto OPC-HDA

En la tabla 4.1 se muestra cada uno de los elementos que conforman el interfaz que implementa el modelo de objeto OPC-HDA.

Objeto	Descripción
OPHDAServer	Instancia de un servidor OPC-HDA.
OPCHDAItems	Colección de OPCHDAItem.
OPCHDAItem	Representa la definición de un item o canal de información.
OPCHDABrowser	Permite navegar por la estructura de canales disponibles.
OPCHDAHistory	Colección de valores de un canal determinado.
OPCHDAValue	Representa un valor histórico discreto de un canal determinado.

Tabla 4.1: Elementos que conforman el interfaz que implementa el modelo de objeto OPC-HDA

De esta manera, el componente que implemente esta interfaz servirá de clase base para la construcción de cualquier servidor OPC-HDA disponiendo así de un mecanismo autónomo para la recuperación de los datos de cualquier dispositivo que forme parte de una instalación de gestión energética.

Al igual que en el caso de la comunicación con cualquier dispositivo del que se quiera recuperar la información de sus canales instantáneos, para la recuperación de los valores históricos, la tecnología OPC establece el diagrama de secuencia genérico que se muestra en la figura 4.35.

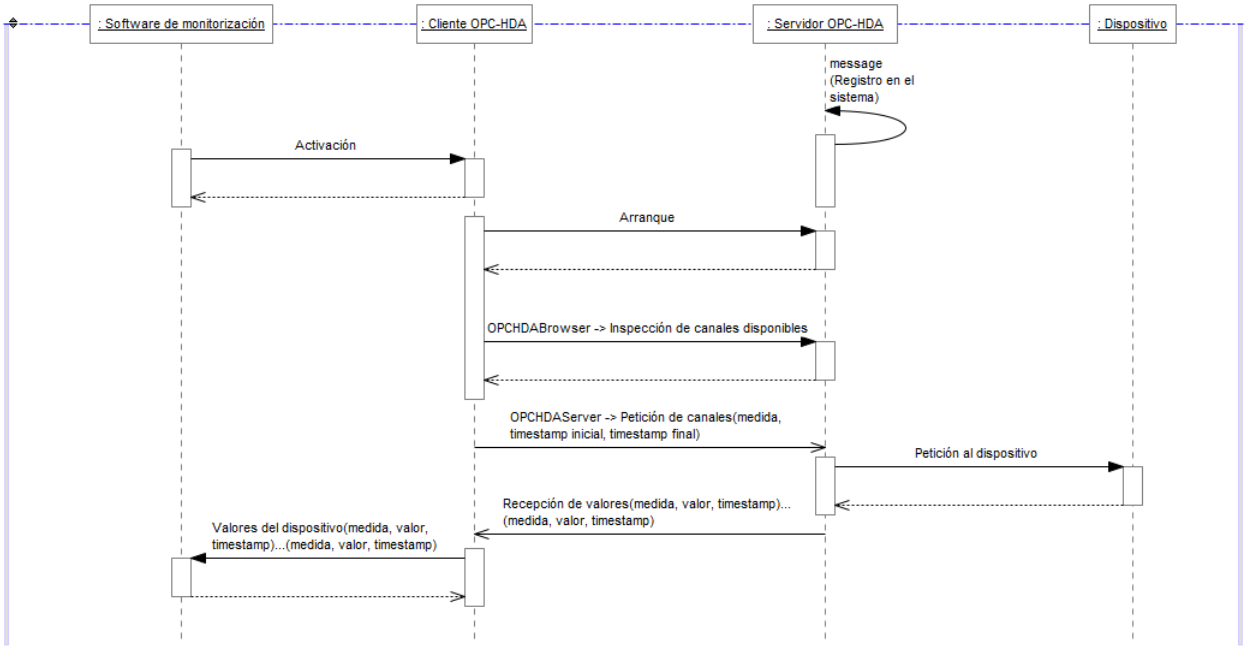

Figura 4.35: Diagrama de secuencia entre un sistema de monitorización y un dispositivo a través de un cliente OPC-HDA y un servidor OPC-HDA

En este diagrama puede verse cómo aparecen varios actores. Por un lado el software cliente que usa la tecnología OPC-HDA para descargarse los datos de un dispositivo,

el dispositivo a interrogar y el servidor OPC-HDA que conoce el protocolo y el procedimiento apropiado para descargar los datos de dicho dispositivo.

Antes de poder utilizar un servidor OPC-HDA, este debe estar disponible en el sistema operativo; para ello, su registro se realiza utilizando el mecanismo de registro de componentes COM/DCOM y haciendo uso del registro del sistema. Una vez registrado, paso que suele hacerse una única vez en el momento de su instalación, este servidor está disponible para cualquier sistema que quiera hacer uso del mismo.

En la figura 4.36 se muestra la clase base propuesta como punto de partida para creación de cualquier servidor OPC-HDA. Esta clase TCustomHDAServer implementa la lógica OPC-HDA con los componentes COM instalados en el sistema y dispone además de un interfaz de usuario para darle a todos los servidores un aspecto visual similar y homogéneo.

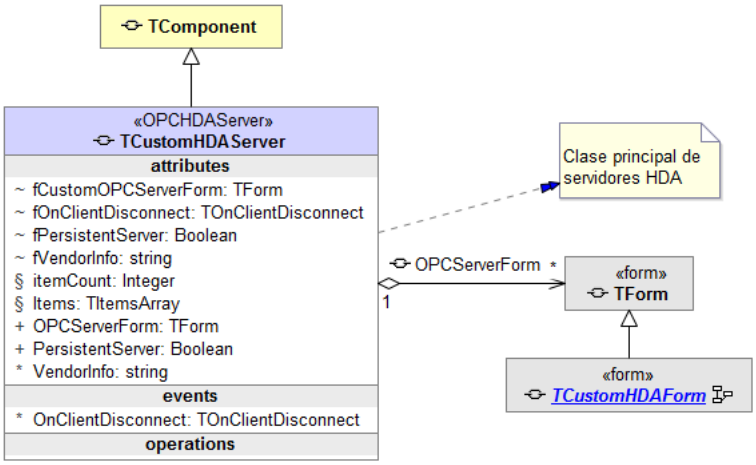

Figura 4.36: Implementación de la clase genérica para servidores OPC-HDA

En la figura 4.37 se detalla el interfaz de usuario único para cualquier servidor OPC-HDA. Existe una referencia bidireccional con la clase que implementa la lógica OPC-HDA y, además de un menú donde realizar las acciones más comunes, como puede ser registrar, cerrar, mostrar los eventos, se ha añadido una ventana de bienvenida que suele ser muy útil para identificar con qué dispositivos es compatible el servidor OPC-HDA que está siendo usado.

Siguiendo la estructura desde la clase más genérica hasta la clase más específica, que suele ser la que implementa las particularidades para cada dispositivo, se ha creído conveniente desarrollar una clase que implementa las funcionalidades para permitir a cada servidor guardar de manera homogénea toda la información que necesiten, relativa en la mayoría de las veces, a los parámetros de configuración y conexión con los dispositivos de los que van a extraer la información, figura 4.38.

En esta clase existen métodos virtuales que pueden ser sobreescritos por las clases inferiores para configurar el mecanismo de almacenamiento de los parámetros que sean necesarios.

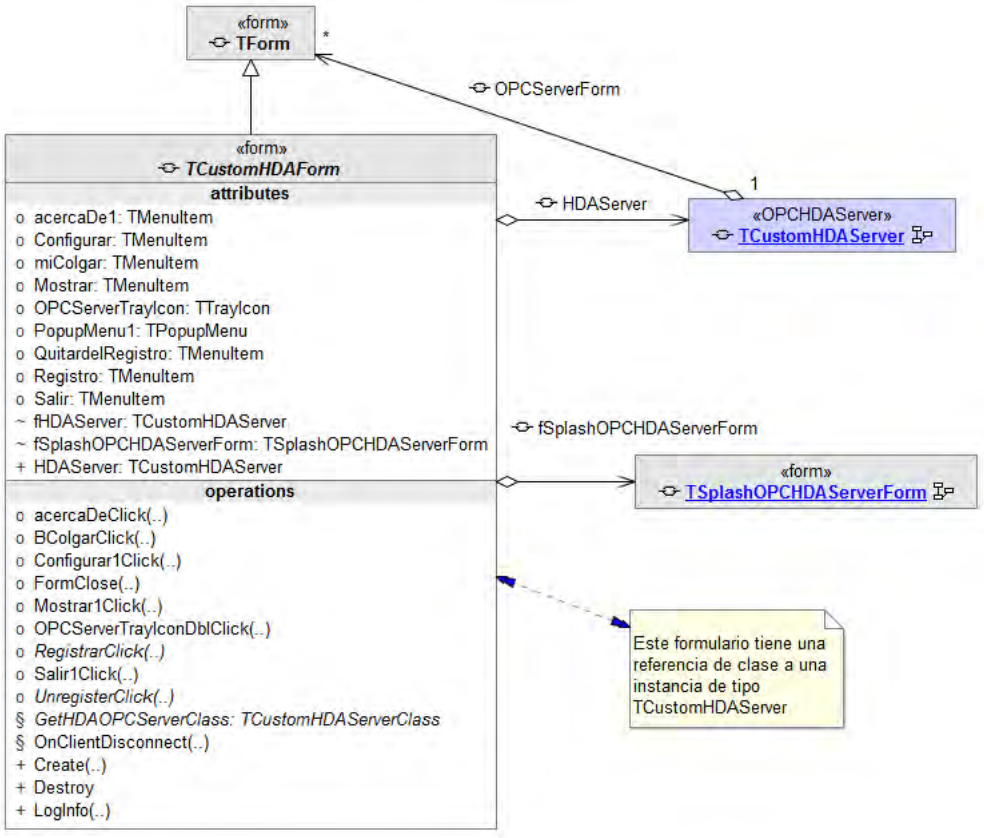

Figura 4.37: Implementación de la clase genérica para el interfaz de usuario de servidores OPC-HDA

Con esta jerarquía propuesta, en los siguientes apartados se detallan algunas implementaciones de servidores OPC-HDA para dispositivos reales muy utilizados en sistema de gestión energética.

4.4.2. Servidor OPC-HDA para inversores con protocolo Modbus

Uno de los protocolos más utilizados para la recuperación de información histórica de los dispositivos es el basado en ModBus. Es por esta razón por la que se propone la generalización de todos los dispositivos de los que disponen de este protocolo en una única clase que encapsulará la lógica y el algoritmo de comunicación que es común para todos los tipos de inversores que utilizan ModBus.

Partiendo de la clase base anteriormente descrita, se pone a la disposición del marco de trabajo la clase TCustomIngeconHDAServer, que implementa los interfaces de configuración de parámetros, la configuración del protocolo Modbus y demás clases de apoyo para representar a cualquier inversor de este tipo, figura 4.39.

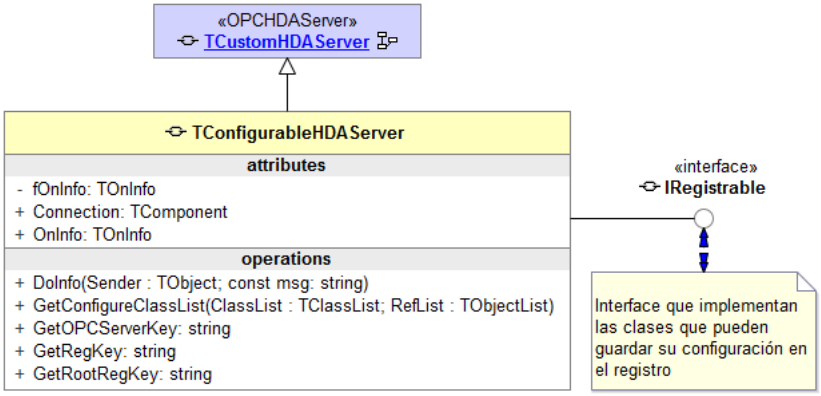

Figura 4.38: Implementación de la clase genérica para servidores OPC-HDA configurables

En la figura 4.40 puede verse cómo es sencillo implementar servidores OPC-HDA para cualquier inversor de este tipo, haciendo uso de la herencia de clases e interfaces ya que las clases anfitrionas son las encargadas de implementar la lógica común y solo las particularidades de los diferentes clases de inversores son realizadas en las clases particulares.

En el caso de aparecer dispositivos con algunas particularidades muy específicas, como en el caso de inversores que se conectan utilizando la tecnología GPRS, se pueden agrupar dichas funcionalidades e implementar las interfaces apropiadas para resolver el problema de la implementación de dispositivos con características comunes. En la figura 4.41 se puede observar el diseño de la clase TCustomIngeconHDAServerGPRS y el interfaz IPlantWithGPRSConfiguration para dar solución al problema sugerido.

Esta interfaz es una extensión de una interfaz disponible en el marco de trabajo para proporcionar la capacidad de configuraciones a las clases que la implementen o hagan uso de ella. Pueden verse que los atributos de esta interfaz pertenecen a parámetros de configuración de una conexión IP y en este caso además a parámetros de autenticación para acceder a la información del dispositivo.

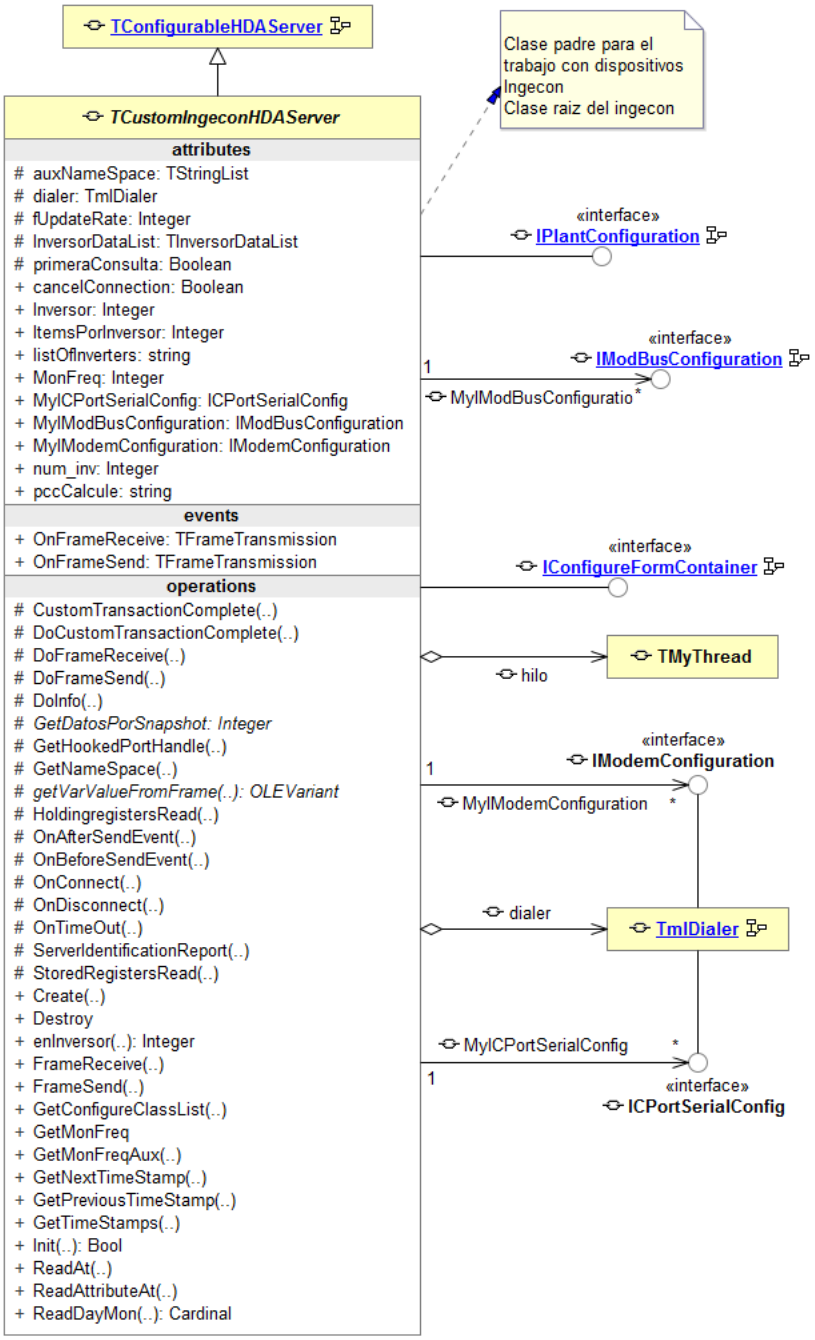

Figura 4.39: Implementación de la clase genérica para servidores OPC-HDA para inversores con protocolo Modbus.

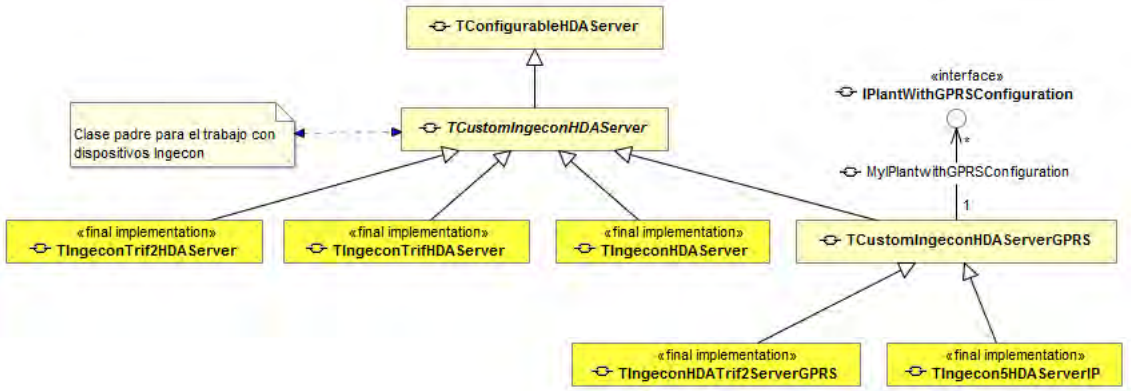

Figura 4.40: Implementación de las clases particulares de servidores OPC-HDA para inversores con protocolo Modbus

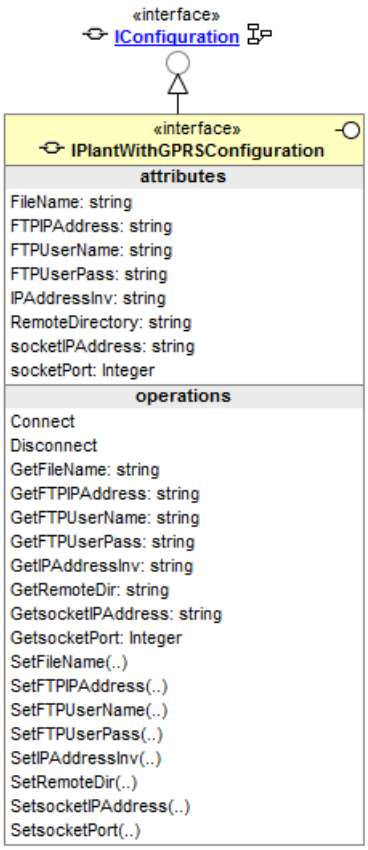

Figura 4.41: Implementación de la interfaz de conexión utilizando IP a través de GPRS

4.4.3. Servidor OPC-HDA para comunicación con grupos de inversores

Algunos sistemas cuentan con un dispositivo que recolecta la información de distintos inversores. Un ejemplo muy utilizado es el dispositivo WebBox que interroga a una serie de inversores y almacena la información de los mismos en su memoria interna. Por otro lado, ofrece una conexión HTTP que permite con un navegador web acceder a su interior y ver los datos almacenados en forma de página web.

Este tipo de dispositivos son interesantes ya que si bien, en caso de disponer de muchos inversores, podríamos conectarnos a todos, podemos dejar que sean estos dispositivos quienes se encarguen de recoger la información de los mismos y conectarnos a él para descargar toda la información de todos los inversores.

En este caso se tiene un dispositivo que tiene un comportamiento que podría ser común, ya que muchos otros dispositivos podrían almacenar la información en formato de ficheros, por lo que la solución a adoptar pasará por dar una solución genérica a la misma e incorporar al marco de trabajo dicha solución fuertemente reutilizable.

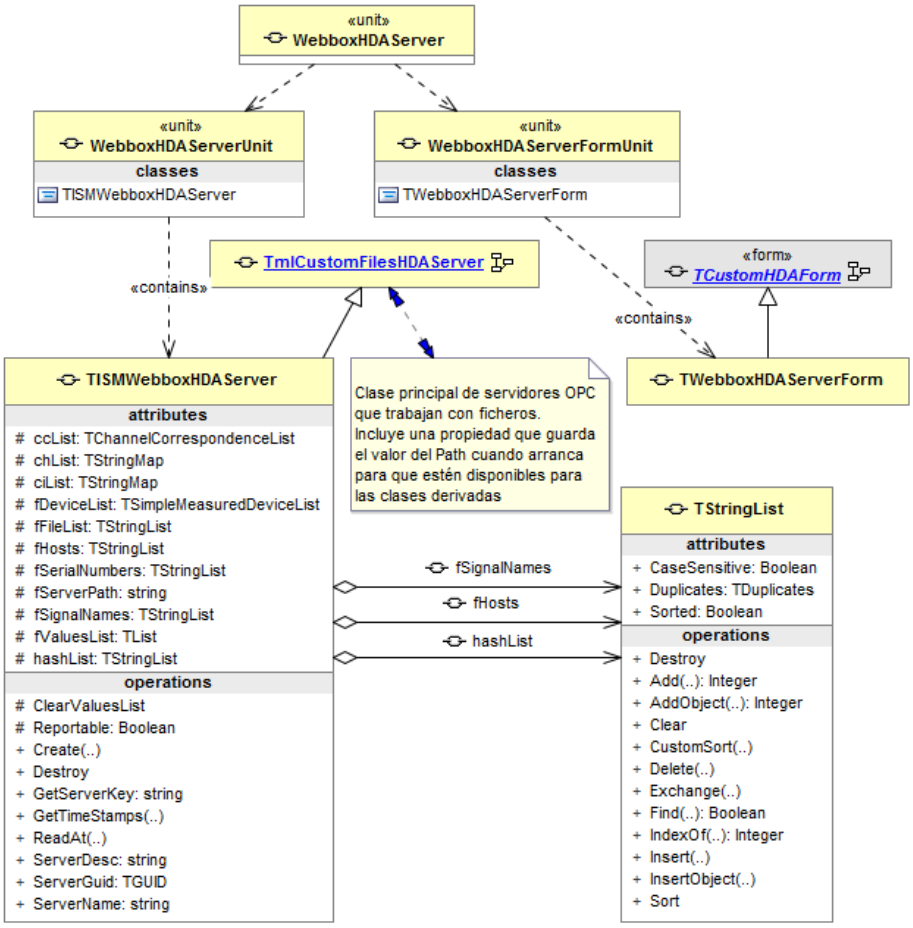

Figura 4.42: Implementación del servidor OPC-HDA para WebBox

Para ello la propuesta que se hace es crear una clase que dé soporte para el tratamiento de datos en formato de ficheros y se utilizará, haciendo uso de la herencia, como base de la implementación final. En este caso será la clase *TmlCustomFilesHDAServer*, figura 4.42.

En la figura 4.42 puede verse la clase desarrollada para el marco de trabajo fruto de su detección durante la fase de desarrollo de un nuevo servidor OPC-HDA. Esta situación hace patente que la recogida de requisitos al inicio de un proyecto ha llegado a no ser importante en estos últimos tiempos, donde las nuevas metodologías ágiles, como la que se ha utilizado para el desarrollo de este marco de trabajo, se centran en la capacidad de adaptación al cambio o incorporación eficiente de requisitos en un proyecto ya comenzado.

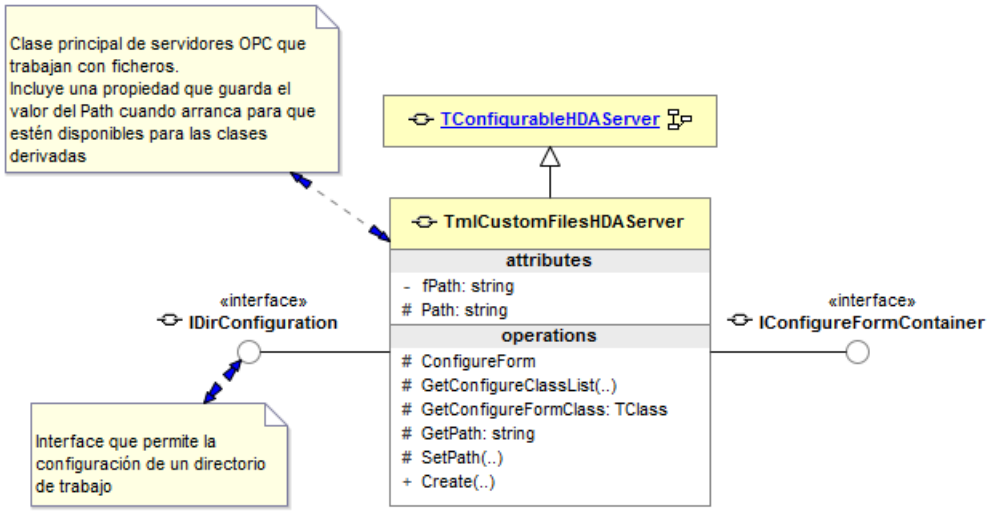

Figura 4.43: Implementación del servidor OPC descarga de ficheros de los inversores

4.4.4. Servidor OPC-HDA para inversores con servidor FTP

Algunos inversores disponen de un método bastante común de recuperación de la información ya que incorporan en su interior un servidor FTP. Debido a que este mecanismo es muy usual, durante el desarrollo de este servidor se ha visto apropiado generalizar la situación para dar solución a casos similares incorporando esta solución al marco de trabajo.

Una de las metodologías aplicadas en el desarrollo de este marco de trabajo ha sido siempre implementar contra interfaces para evitar de esta manera los problemas de la herencia múltiple y facilitar la reutilización. De esta manera se genera el interfaz que da soporte para el comportamiento de descarga de ficheros utilizando el protocolo FTP y se crean clases intermedias que lo implementan facilitando así la reutilización de dichas clases, figura 4.44.

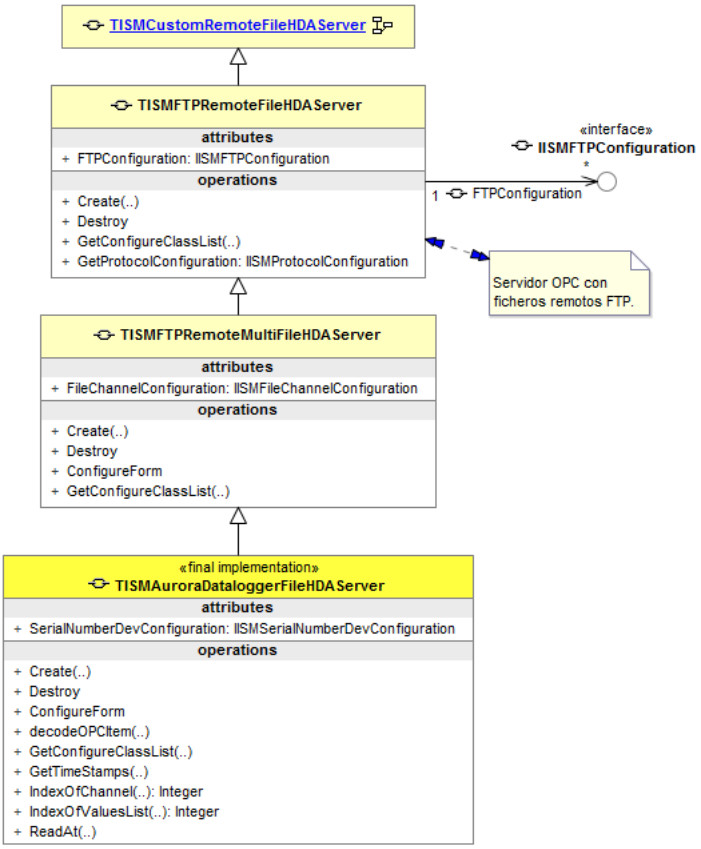

Figura 4.44: Implementación del servidor OPC para inversores con servidor FTP.

En la jerarquía desarrollada puede observarse que la posibilidad de implementar toda la lógica en la clase particular se ha abandonado en favor de distribuir la implementación en clases de apoyo que reutilizar. Así se dispone de las clases *TISMFTPRemoteFileHDAServer* y *TISMFTPRemoteMultiFileHDAServer* que forman parte del marco de trabajo y que son usadas para implementar el servidor OPC-HDA y su lógica de descarga por FTP. Todas estas clases parten en su origen de una clase base

desarrollada con el fin de implementar la lógica genérica de tratamiento de datos de dispositivos en forma de ficheros, figura 4.45.

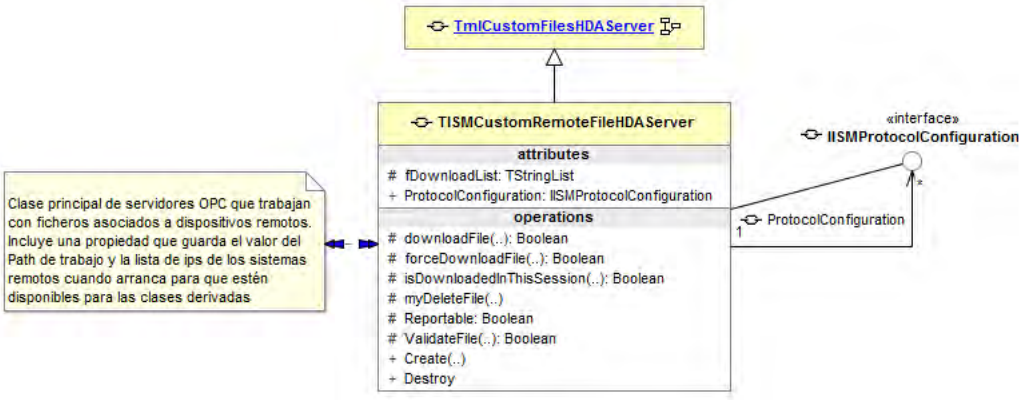

Figura 4.45: Implementación de la clase base para la descarga de ficheros remotos de dispositivos.

4.4.5. Servidor OPC-HDA para inversores con protocolo propietario

Algunos inversores disponen de un protocolo de comunicaciones propio por lo que en este caso no se puede reutilizar código generado anteriormente pero sí es posible partir de la clase que permite implementar todas las funcionalidades de un servidor OPC-HDA.

Durante la investigación y el estudio realizado, se detectó que para un mismo inversor puede haber diferentes protocolo dependiendo de la versión del firmware instalado en ese momento en el dispositivo, por lo que se decidió crear una clase genérica para el tratamiento de cualquier inversor con protocolo propietario y posteriormente una clase con las particularidades de cada inversor.

Así la clase *ISMSunwaysNTHDAServer* implementa el protocolo de comunicaciones propietario del fabricante y se ha construido implementando a su vez la interfaz *IConfigureFormContainer* para proporcionar el comportamiento de configuración, figura 4.46.

De esta manera es posible desarrollar los servidores OPC-HDA correspondientes a este tipo de inversores para cada versión del firmware usando como base la clase genérica suministrada por el marco de trabajo. En las figuras 4.47 y 4.48 se muestra la implementación del servidor OPC-HDA para un inversor comercial con protocolo propietario para dos versiones distintas del mismo inversor.

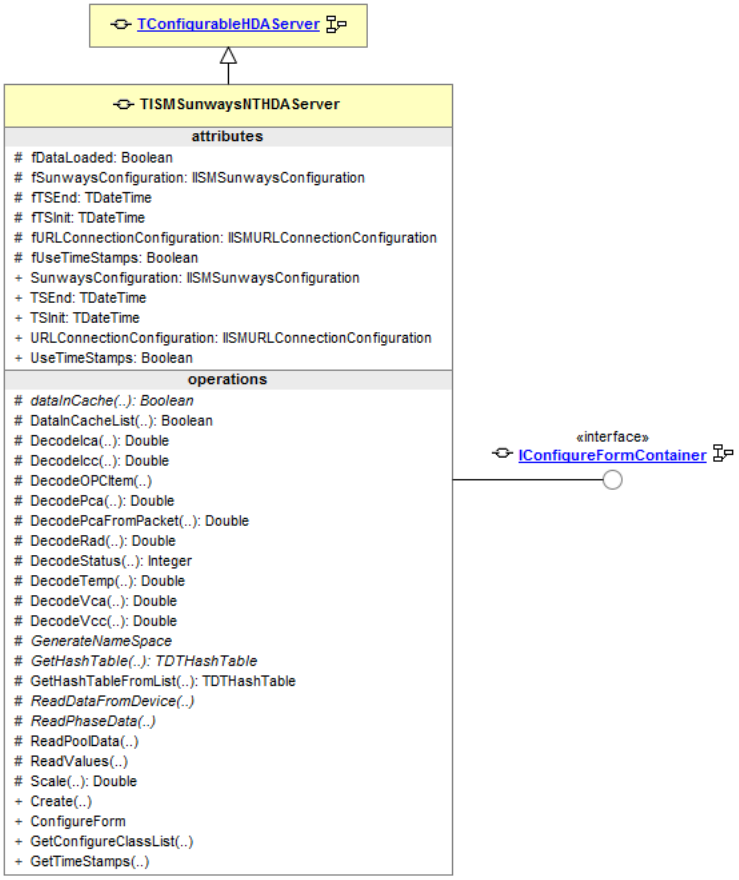

Figura 4.46: Clase que implementa el protocolo propietario.

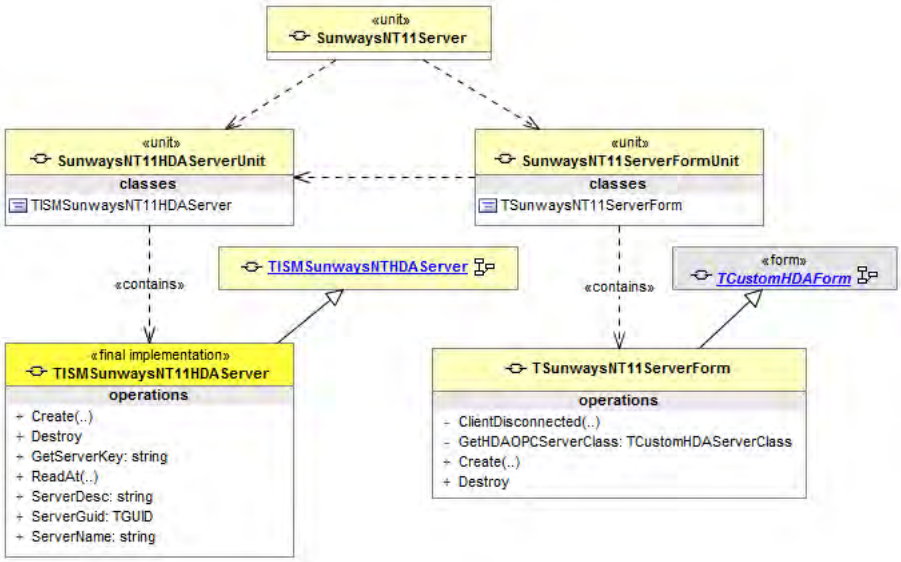

Figura 4.47: Implentación de servidor OPC-HDA para un inversor comercial con protocolo propietario, versión 1.1

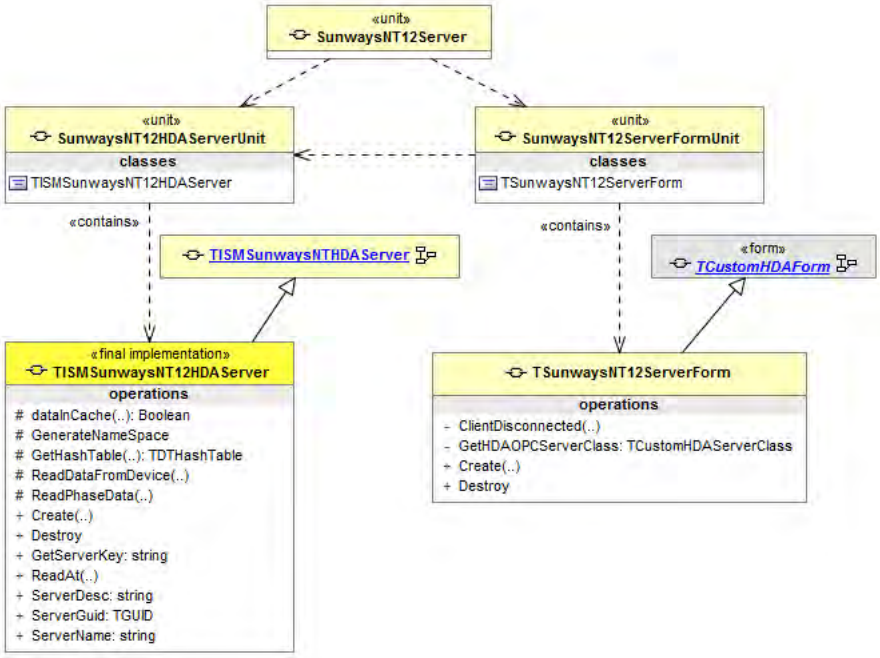

Figura 4.48: Implentación de servidor OPC-HDA para un inversor comercial con protocolo propietario, versión 1.2

4.5. Conclusiones

Cuando hay que realizar una generalización sobre sistemas heterogéneos surge el problema de que es difícil encontrar los puntos en común de dichos sistemas. En este caso, resolver el problema de comunicación de manera homogénea entre sistemas diferentes requería de una solución robusta y eficaz. Para dar una respuesta al mismo, en este trabajo se propone la utilización de la tecnología OPC y el desarrollo de una serie de clases e interfaces que proporcionen una manera rápida de desarrollar los componentes de comunicación.

Con esta propuesta se ha logrado disponer de una solución que permite la integración de los diferentes dispositivos dentro de una instalación energética y lo que es más importante, su gestión y capacidad de inclusión de nuevos componentes y modelos sin que afecte al resto del sistema.

Para validar esta capa desarrollada del marco de trabajo, se muestra la implementación, haciendo uso de los mecanismos que ofrece la programación orientada a objetos, de diferentes módulos para la comunicación con dispositivos reales que se encuentran de manera muy frecuente en instalaciones de energía solar. Así, la creación de nuevos módulos para la comunicación con diferentes dispositivos, resuelve el problema indicado al inicio de este capítulo disponiendo de una solución altamente desacoplada y muy eficiente a la hora de mantener, modificar y extender.

"Es más fácil cambiar las especificaciones para que encajen con el software que hacerlo al revés"
– Alan Perlis

5

Capa de caracterización y modelizado de sistemas energéticos

5.1. Introducción

En este capítulo se hace una propuesta de modelo que permita describir cualquier sistema genérico a partir de los componentes propios del dominio de instalaciones energéticas. En la figura 5.1 puede verse un esquema jerárquico de los diferentes elementos que componen el modelizado de una instalación energética dentro del marco de trabajo. Se hace esta propuesta de la jerarquía de estos elementos, para su modelizado en el marco de trabajo. Esta jerarquía representa la estructura de las configuraciones posibles que tiene las instalaciones que se modelizan en este trabajo. Permite la definición de todos los elementos, dispositivos e información incluidos en cualquier sistema de este tipo.

Para establecer esta jerarquía, el elemento base es la *medida* que corresponde a algún canal de información de cualquier dispositivo vinculado a una instalación. Las *medidas* pueden corresponder a medidas registradas por algún dispositivo de una planta, a parámetros estimados o calculados a partir de los valores registrados o incluso a valores constantes para almacenar información necesaria para el modelizado y gestión de cada sistema. En el primer caso se hará referencia a ellas como *medidas registradas* y las demás se denominarán *medidas virtuales o estimadas*. Por ejemplo, en un sistema de energía solar fotovoltaica, las *medidas registradas* son: voltaje del generador, potencia del generador, corriente continua y alterna del inversor, radiación, temperatura de los módulos, etc. Las *medidas virtuales* se utilizan para calcular algunos parámetros de

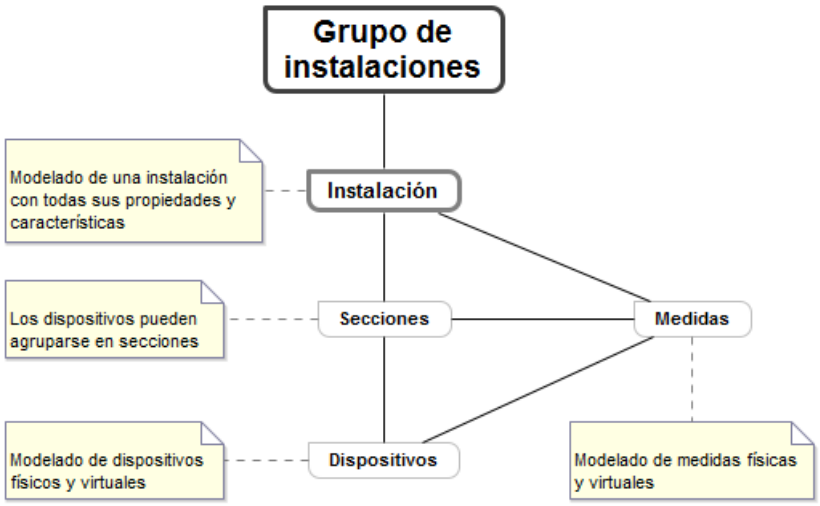

Figura 5.1: Esquema de los elementos que componen una instalación

la planta y para implementar los modelos de evaluación y predicción de los sistemas. Algunas de las *medidas virtuales* que se han definido a partir de los datos registrados para el citado sistema fotovoltaico podrían ser: eficiencia del inversor, energía diaria producida, rendimiento de la planta, etc.

En la jerarquía se puede apreciar que no solo los dispositivos disponen de canales o medidas de información, sino que también una instalación o una agrupación de dispositivos puede definir sus propios canales de información como medidas asociadas. Esto significa que cada uno de estos elementos de la jerarquía propuesta puede tener sus propias *medidas*, tanto *registradas* como *estimadas o virtuales*.

El elemento *dispositivo* tiene, por una parte, toda la información descriptiva del mismo, tal como nombre, tipo de dispositivo o planta en la que está instalado. Un *dispositivo* se modela como un elemento que tiene varias *medidas* que a su vez pueden tener simultáneamente una lista de elementos. De esta forma es posible modelizar un dispositivo utilizando la composición de dispositivos; y se puede incluso crear algún elemento abstracto partiendo de un conjunto de dispositivos (como puede ser, por ejemplo, una medida estimada sobre algún parámetro de evaluación del conjunto de dispositivos). Este tipo de elemento abstracto se utiliza para almacenar los modelos de evaluación y predicción de las instalaciones al igual que si se tratase de un dispositivo real, lo que facilita los resultados de la evaluación a través de sus canales de información, reduciendo así la complejidad del código necesario para implementar esta funcionalidad.

Los *dispositivos* pueden ser agrupados en *secciones*. Varias *secciones* conforman una *instalación* y, finalmente, una o más *instalaciones* dan lugar a lo que se ha denominado como *grupo de instalaciones*. Esta última agrupación es muy útil cuando un mismo *usuario* tiene más de una instalación, de manera que el marco de trabajo permite generar aplicaciones para mantener toda la información de distintas plantas con la misma aplicación y asociados a la misma cuenta de usuario.

Esta organización de los elementos de una planta proporciona una correspondencia entre una planta real y el modelo de planta que se construye en el marco de trabajo.

La representación de un sistema se puede realizar mediante las siguientes reglas:

Grupo de instalaciones	→	*Instalación*
	|	*Grupo de instalaciones Instalación*
	;	
Instalación	→	*Sección*
	|	*Instalación Sección*
	;	
Sección	→	*Dispositivo*
	|	*Sección Dispositivo*
	;	
Dispositivo	→	*Medida*
	|	*Dispositivo Medida*
	;	

Para cada uno de esos elementos se pueden especificar las medidas asociadas. Así por ejemplo, para el elemento *Instalación*, las medidas asociadas se podrían representar de la siguiente forma (r: medidas registradas, v: medidas estimadas)

- Medidas registradas a nivel de Instalación, $Instalacion.r_i$, corresponden a las i medidas que pueden registrarse para una Instalación.
- Medidas virtuales estimadas para una Instalación, $Instalacion.v_j$, son las j medidas que se calculan a partir de otras medidas de la Instalación, y que pueden ser función de una o más de las siguientes medidas:

  $Seccion.r_k, Seccion.v_l, Dispositivo.r_m, Dispositivo.v_n, medida.r_p, medida.v_q$

Como ejemplo de posibles medidas asociadas a una *instalación* podrían ser aquellas donde un dispositivo devolviese un valor que afectase a la instalación por completo como *temperatura_ambiente*, *E_contador* (si hubiera un contador general que registrara toda la energía generada por la Instalación), etc.; las posibles medidas estimadas para una instalación podrían ser: *Potencia_total*, que se calcularía como suma de las potencia de todos los generadores que haya en la instalación, *Rendimiento_diario*, calculado a partir del balance energético diario de la instalación, etc. También se pueden definir medidas virtuales para almacenar información de la instalación necesaria para su correcta gestión, como puede ser *latitud* y *longitud* del emplazamiento, etc.

Así, de los elementos descritos, los dispositivos suelen ser los que registran los datos físicos de la instalación (medidas físicas de parámetros reales, como puede ser potencia, intensidad, etc.). Pero todos ellos pueden tener asociada información que no corresponda con las medidas físicas pero que sea útil para describir características de los mismos. Esta información se almacena en forma de lo que se ha denominado *medida virtual o*

calculada. En un apartado posterior se definen detalladamente todos los tipos de medida que se han contemplado en el marco de trabajo. Cada medida (real o virtual) irá siempre asociada a uno de los elementos enumerados anteriormente.

5.2. Modelizado y gestión de usuarios

En el momento que existe la posibilidad de desarrollar un sistema que puede ser usado por diferentes personas aparece la necesidad de establecer criterios o restricciones para el acceso de la información que dependa del rol de la persona que está accediendo al mismo.

La inclusión de la funcionalidad que permita modelizar diferentes roles con el marco de trabajo proporciona que un determinado usuario disponga de diferentes privilegios frente a diferentes sistemas a gestionar. Cada usuario será único en el sistema y se autenticará con un nombre de usuario y una contraseña. Un usuario dispondrá de capacidad de asociación con un grupo de instalaciones por diferentes roles y así podrá disponer de diferentes roles dependiendo de las instalaciones o sistemas a las que tenga acceso.

Se han definido los siguientes tipos de roles o perfiles:

- Administrador: los usuarios con este perfil tienen acceso a todos los sistemas y puede realizar cualquier tarea dentro del marco de trabajo.

- Gestor: los usuarios con este perfil dispondrán de acceso a un grupo de instalaciones, permitiendo de esta manera, que un mismo usuario tenga acceso y control con sus mismos datos de acceso a varias instalaciones.

El marco de trabajo proporciona las clases e interfaces apropiadas para modelizar el almacenamiento de los roles al mismo, siendo además posible extenderlas proporcionando nuevas funcionalidades a la hora de utilizarlas. Primeramente se ha creado una clase ISMDBUsers que contendrá la lista de usuarios y permitirá gestionar y proporcionar a los usuarios del marco de trabajo la gestión con usuarios, figura 5.2. Esta clase soporta el interfaz IUsers que implementa el modelizado de dicho comportamiento. Además, dispone de los mecanismos de persistencia para poder almacenar y recuperar la información de la base de datos asociada al marco de trabajo siguiendo un modelo relacional mostrado en el capítulo 6.

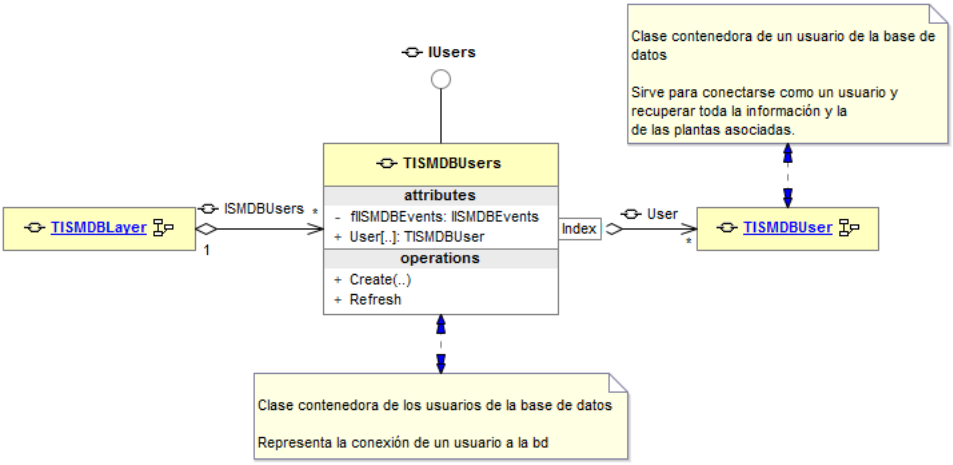

Figura 5.2: Clases para el modelizado de usuarios

Cada usuario dispone de propiedades para almacenar la información más relevante. Esta clase puede extenderse siempre que se implemente las funcionalidades del interfaz correspondiente. En la figura 5.3 se muestran las clases para el caso IISMDBUser.

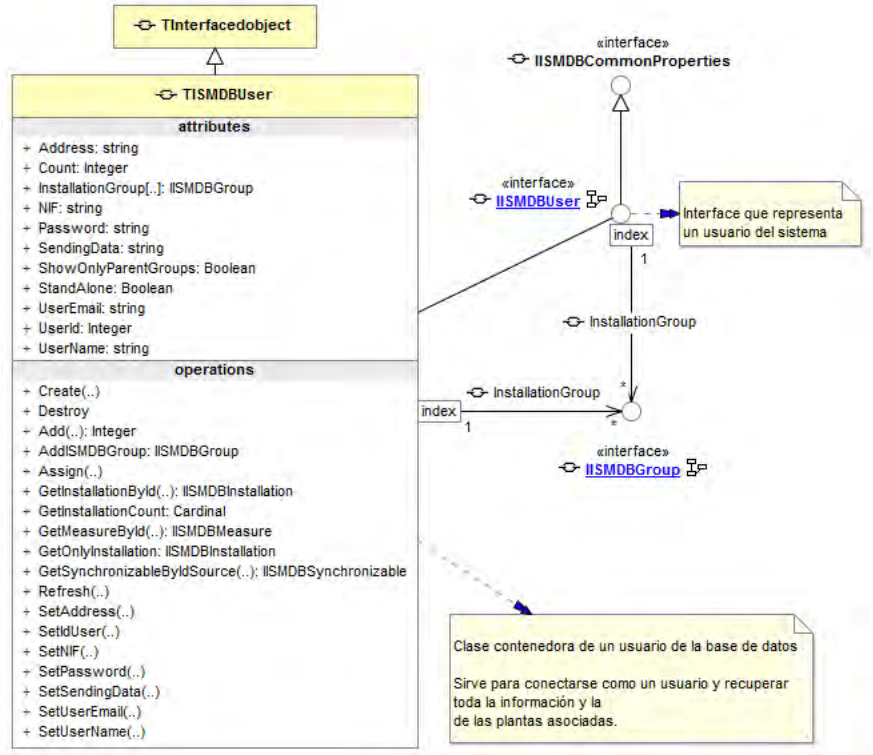

Figura 5.3: Clases para el modelizado de un usuario

Puede verse que un usuario puede relacionarse con un grupo de instalaciones sobre las que tendrá permisos para acceder y gestionar a través de la propiedad *InstallationGroup*

5.3. Modelizado de una instalación energética

Para la utilización del marco de trabajo en la gestión de sistemas energéticos, una de las tareas fundamentales que se deben poder realizar de manera sencilla es el modelizado de cada sistema que se vaya a gestionar. Este modelizado debe permitir definir todos los dispositivos que lo integran y la asociación del sistema a uno o más usuarios con su rol determinado. Para ello se parte de una jerarquía genérica para más adelante ver las posibilidades de extensión, figura 5.4.

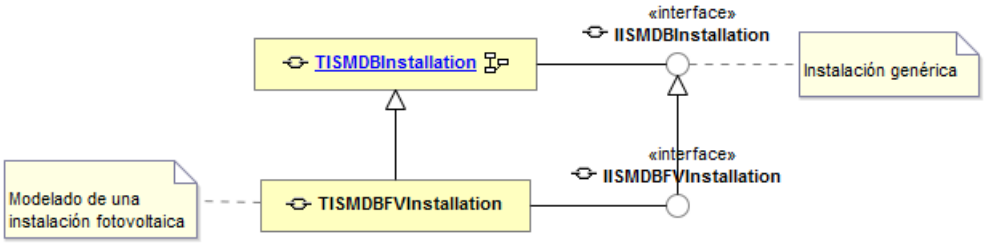

Figura 5.4: Modelizado de una instalación genérica basada en interfaces

Cada sistema energético, al igual que el grupo de sistemas, dispondrá de la capacidad de modelizar medidas asociadas a este independientemente de las medidas de los dispositivos.

Una vez que se ha modelizado un sistema o instalación, es necesario modelizar cada uno de los subsistemas que lo integran. Un sistema energético o instalación se puede considerar que está formado por *elementos* cada uno con una capacidad de generación, adquisición o procesamiento de datos. Estos *elementos* aunque dispersos físicamente forman la *instalación* o *agrupación de instalaciones*.

También se ha implementado un mecanismo de asociación para permitir que un sistema disponga de asociaciones transversales. De esta manera, se pueden asociar diferentes dispositivos en diferentes secciones para reseñar que existe una relación física u organizativa entre ambos; por ejemplo, en el caso de una instalación de energía solar fotovoltaica, un inversor y su contador de energía o una placa fotovoltaica y el inversor que recoge la energía que produce.

Una vez se ha descrito el sistema que se propone para modelizar todo el sistema conceptual de una instalación real, se analizan ahora los elementos físicos de los sistemas energéticos. El elemento físico básico que se va a modelizar es el *dispositivo*, que es el componente por excelencia dentro de un sistema energético. En los sistemas energéticos existen multitud de clases diferentes de dispositivos y con objetivos diferentes como pueden ser registradores de datos, inversores, módems, sistemas de adquisición de datos, conversores, sensores de temperatura, células calibradas o piranómetro para medir radiación, etc. Por otra parte, el dispositivo mantiene toda la información descriptiva del elemento que describe como pueden ser el nombre, la clase de dispositivo, descripción y instalación a la que pertenece, entre otras.

En la figura 5.5 se muestran la clase e interfaces que modelan una instalación.

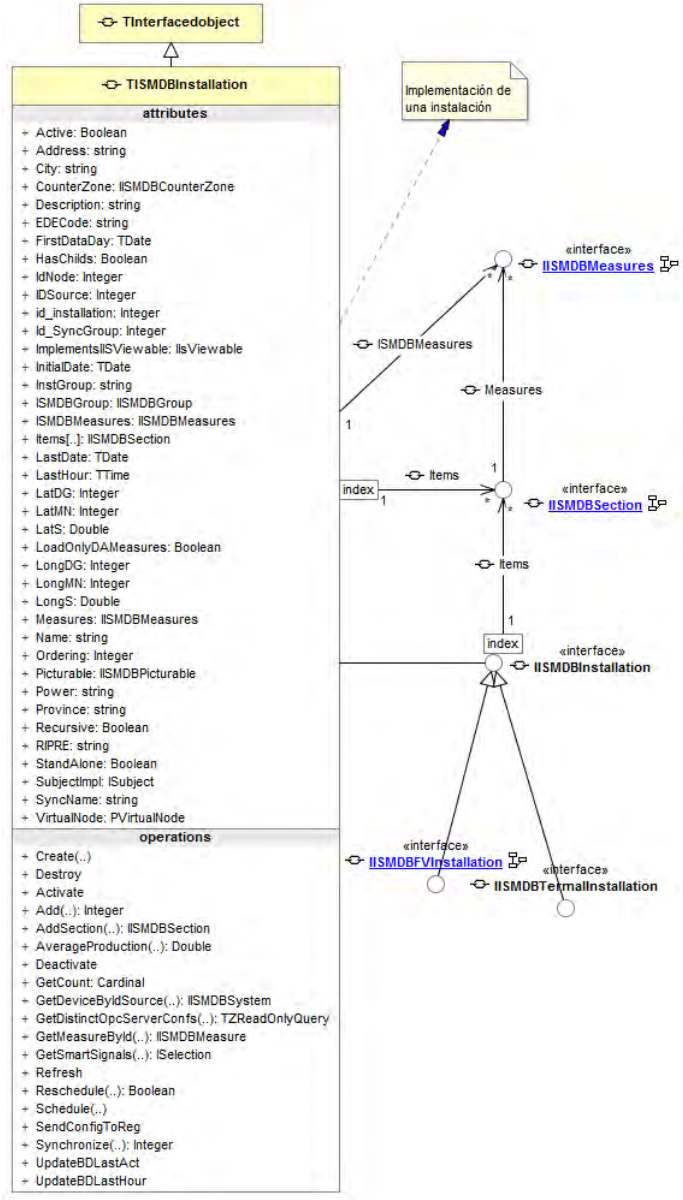

Figura 5.5: Modelo de una instalación

Siguiendo las directrices y metodología de desarrollo expuestas en el capítulo 3, se ha definido el marco de trabajo basando su estructura en interfaces y diseñando las clases que los implementan. De esta manera es sencillo añadir nuevas clases a la estructura siempre y cuando se implementen el comportamiento que define la interfaz. Así, en la figura 5.6 se muestra la capacidad de particularización al poder hacer uso de la herencia y adaptar las clases para instalaciones térmicas o solares *ISMDBFVInstallation* o *IISMDBTermalInstallation*.

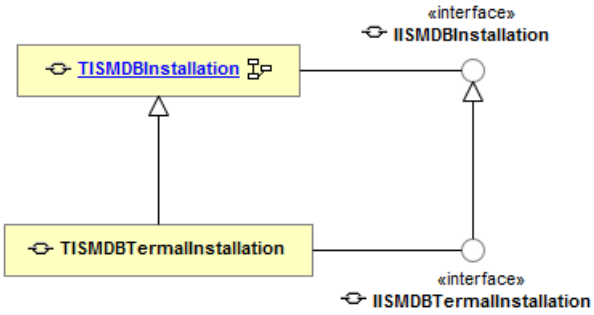

Figura 5.6: Creación de nuevos tipos de instalaciones haciendo uso de la herencia de interfaces

Observando la implementación, figura 5.7 se comprueba cómo la propia instalación puede disponer de sus propias medidas o canales de información *IISDBMeasures* definidas por lo general como medidas virtuales o calculadas a partir de medidas de los dispositivos.

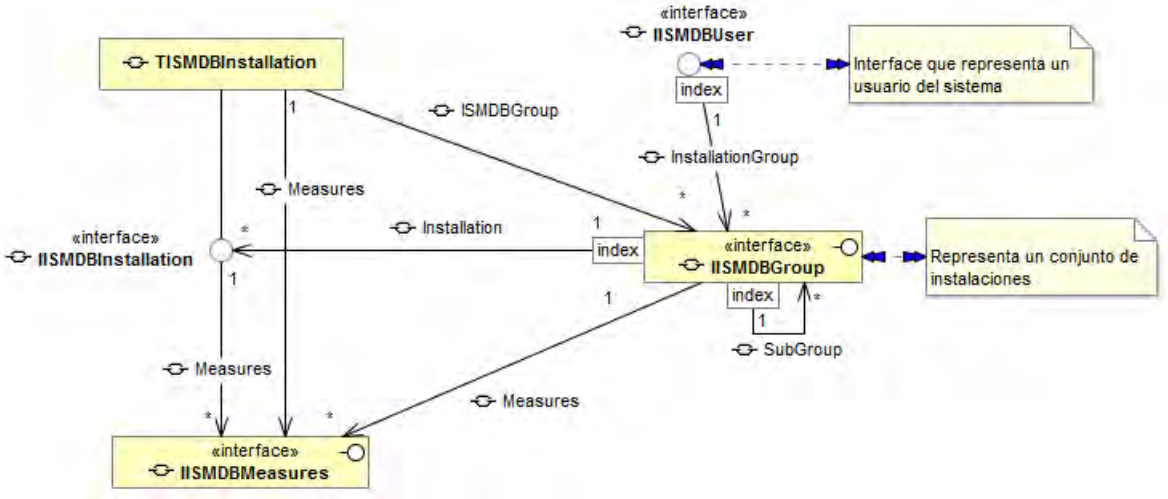

Figura 5.7: Grupos de instalaciones y relación con los usuarios del marco de trabajo

También se observa cómo este modelizado permite agrupar instalaciones a través del interfaz *IISMDBGroup* y cómo su relación con el interfaz *IISMDBUser* establece la asociación entre usuario-instalaciones. Se aprecia la referencia circular de la interfaz que modela la agrupación de instalaciones permitiendo tener tantos niveles de grupos como sean necesarios.

5.4. Modelizado y caracterización de grupos de dispositivos

Para muchos sistemas es necesario disponer de la facultad de poder organizar o agrupar los dispositivos que forman parte de los mismos, ya sea porqué están físicamente organizados de una manera determinada, ya sea porqué es interesante disponer de medidas o canales en las que se vean involucrados una serie determinada de dispositivos.

Para dar respuesta a esta necesidad, a una asociación de dispositivos se la denomina *sección* y representará a un conjunto de dispositivos y a un conjunto de posibles medidas o canales que por su naturaleza pueden representar información sobre los dispositivos que asocia.

En el diagrama de clases asociado que se muestra en la figura 5.8 puede verse la clase TISMDBSection que implementa el interfaz IISMDSection. Éste a su vez dispone de un conjunto de dispositivos o sistemas *IISMDBSystem* y una referencia a un grupo de medidas *ISMDBMeasures*.

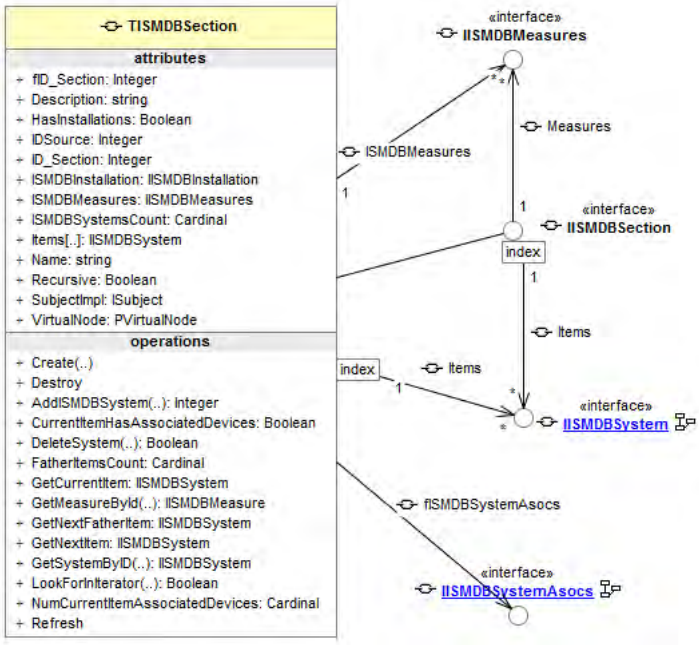

Figura 5.8: Modelo de un grupo de dispositivos

Se ha establecido además la posibilidad de enlazar por medio de la referencia a *ISMDBSystemAsocs* de asociaciones entre grupos de dispositivos o secciones, según se muestra en la figura 5.9.

Esta funcionalidad ha sido necesaria introducirla para establecer una relación entre dispositivos que por su función deben trabajar juntos pero forman parte de una sección donde existen muchos más dispositivos.

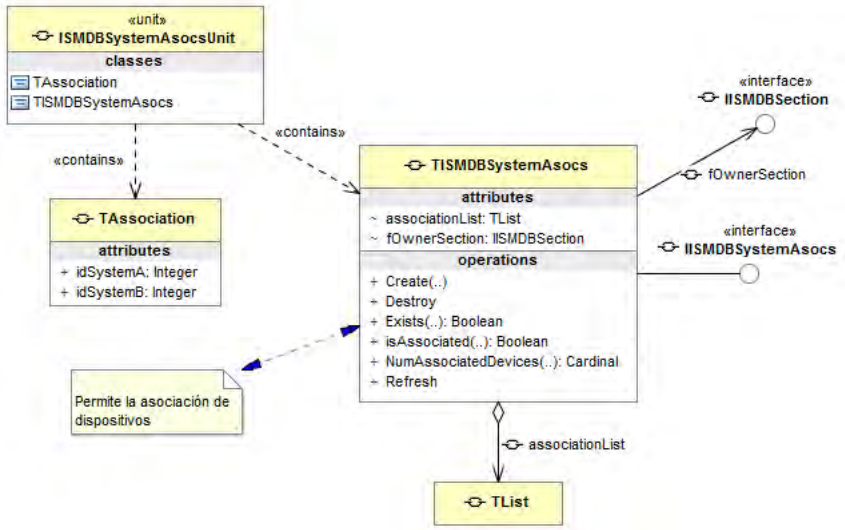

Figura 5.9: Asociación de dispositivos

5.5. Modelizado y caracterización de dispositivos

Los elementos que se han tratado son conceptos abstractos y organizativos de los dispositivos que finalmente forman parte de una instalación. Estos últimos son los que al final se trasladan en elementos físicos y reales que procesan y generan información. Los dispositivos o sistemas que forman parte de una instalación pueden ser de diferente índole y disponer de diferentes canales de información o medidas. Teniendo en cuenta la heterogeneidad que existe respecto a los dispositivos que forman parte de una instalación se ha definido la estructura de clases e interfaces que se muestra en la figura 5.10

Cada dispositivo o sistema mantiene una referencia con la instalación a la que pertenece. Puede verse en la declaración del interfaz la referencia circular a otros posibles dispositivos para tratar el caso que se ha reseñado en el apartado anterior.

El mecanismo de herencia que proporciona la programación orientada a objetos permite particularizar el interfaz de dispositivo para adaptarlo a dispositivos particulares que no puedan ser tratados, por su naturaleza y características, de manera genérica. Así, es posible declarar, partiendo de la interfaz genérica, diferentes dispositivos con sus propias particularidades. En este caso y por ejemplo células de radiación, generadores, inversores o contadores eléctricos.

En la figura 5.11 se muestra, a modo de ejemplo, la utilización de clases e interfaces declaradas en el marco de trabajo para la definición de un dispositivo de tipo inversor.

Simplemente heredando (y de esta manera implementando el interfaz apropiado), se puede extender la clase sistema, que representa a un dispositivo, para definir un inversor con los métodos y propiedades que sean necesarios.

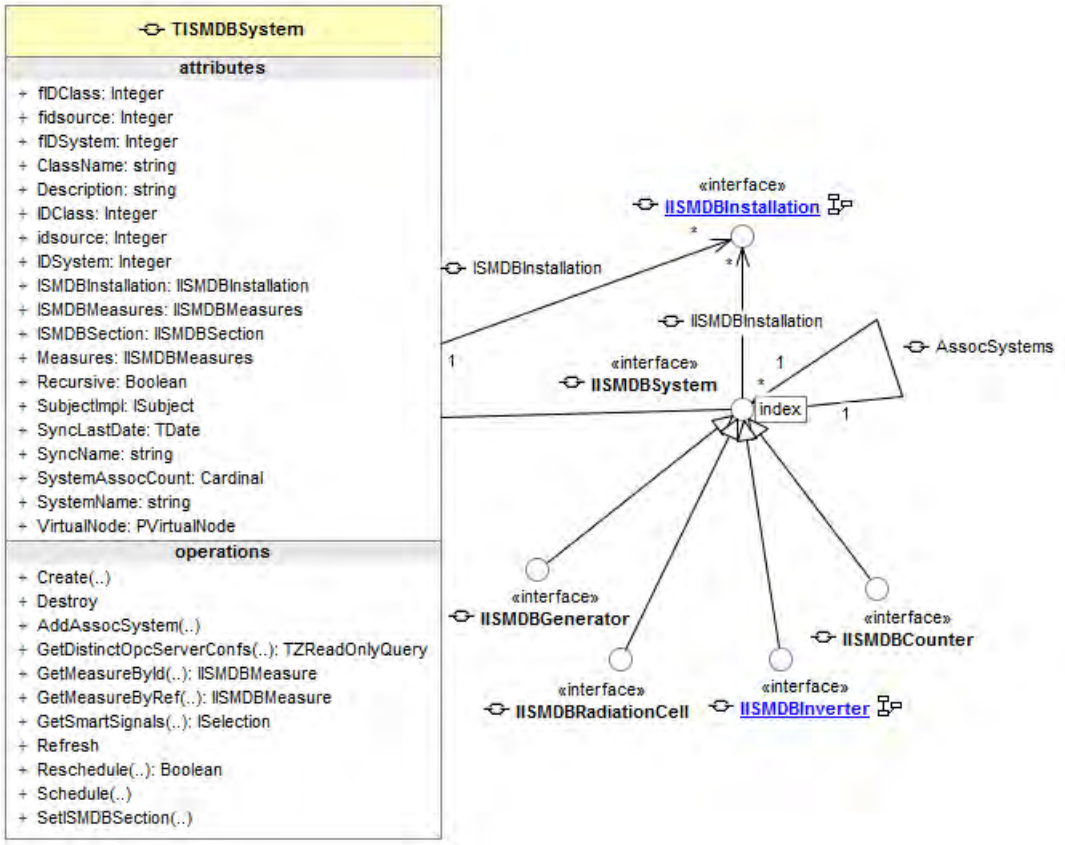

Figura 5.10: Modelado de un dispositivo o sistema

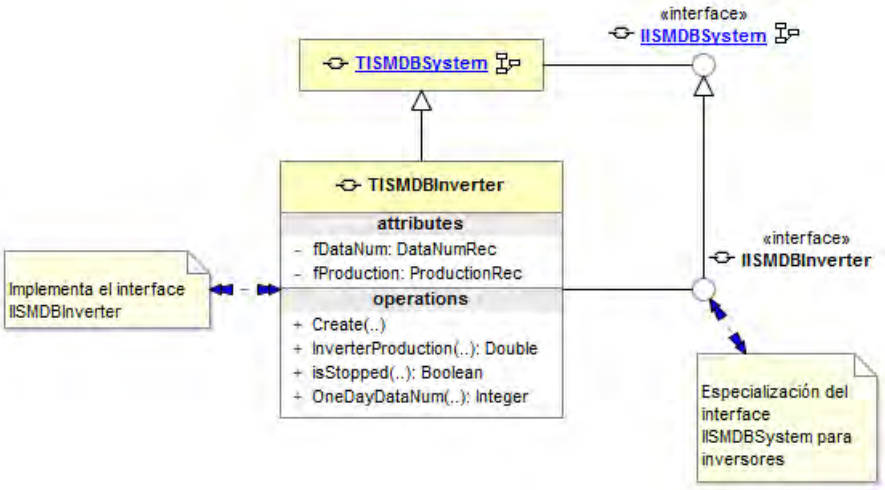

Figura 5.11: Modelado de un dispositivo tipo invesor

5.6. Modelizado de medidas

La unidad mínima de información que se trata dentro del marco de trabajo desarrollado es el valor de una **medida**. La medida representa los valores de un canal físico o virtual y suele venir representado por la tupla `(id, valor, timestamp)`. Para un determinado instante de tiempo se dispondrá de un valor para cada medida identificada por un identificador único suministrada esta por un determinado dispositivo.

En el marco de trabajo se ha contemplado que las medidas puedan ser de distintos tipos, para poder integrar en el mismo los distintos tipos de señales físicas que se registran en un sistema energético. Además, en el marco de trabajo se pueden también modelizar medidas virtuales, lo que da mucha potencia al mismo para las tareas de gestión. Los tipos de medida que se han modelizado y desarrollado en el marco de trabajo para dar respuesta a todos los tipos de magnitudes físicas o calculadas que intervienen en la gestión de un sistema energético se muestran en la figura 5.12, en el que se puede observar la jerarquía del modelizado propuesto.

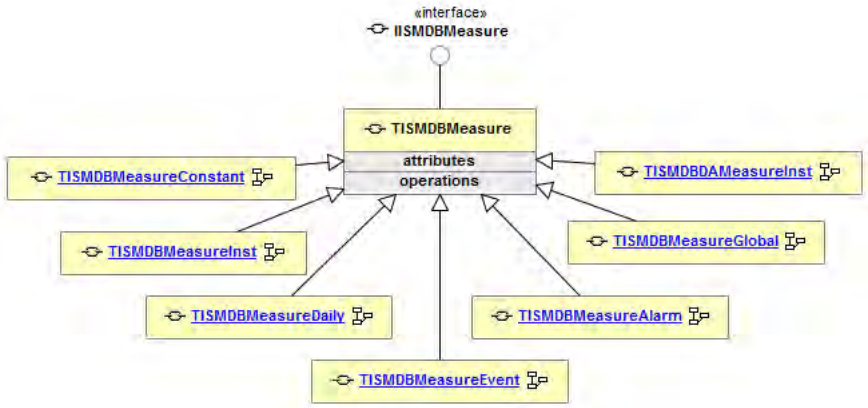

Figura 5.12: Jerarquía de modelizado medidas

Se describen ahora en detalle cada una de estas medidas:

- Medida constante: En ocasiones es necesario disponer de una manera de representar un valor constante en un atributo, como puede ser un número de serie de un dispositivo o un factor de calibración de un determinado canal. Para esto se puede usar una medida de tipo constante asociada a un determinado dispositivo, grupo o instalación, figura 5.13.

- Medida instantánea: Una medida instantánea se describe como una medida que almacena valores de canales que facilitan datos en forma continua; es, por tanto, una medida registrada. Cada valor está descrito por una tupla `(valor,timestamp)` siendo `valor` el valor de la magnitud y el `timestamp` el momento en que esa variable se establece. El `timestamp` puede venir dado por el dispositivo o por el momento en que es adquirida por el sistema. Ya que estos tiempos no tienen por

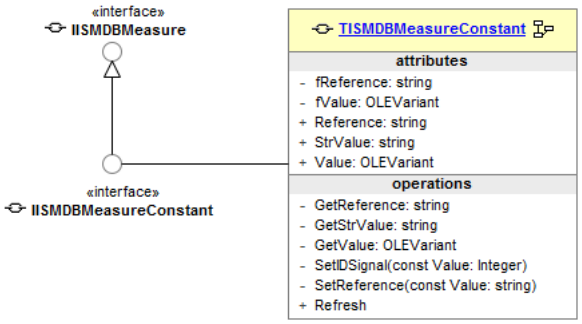

Figura 5.13: Modelado de medidas constantes

qué ser iguales siempre se facilitará el más preciso de ambos o en su defecto el que esté disponible. En la figura 5.14 se muestra el modelo conceptual del sistema de adquisición de tiempos de medidas en el marco de trabajo.

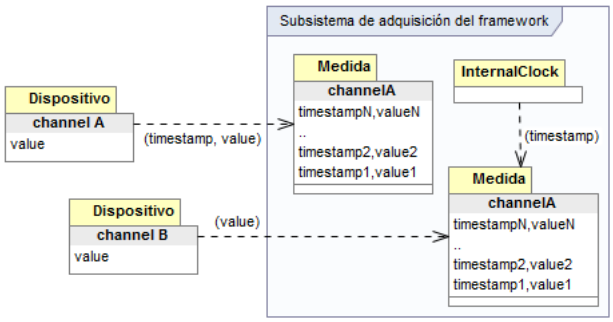

Figura 5.14: Sistema de adquisición de tiempos de medidas

- Medida diaria: Una medida diaria es aquella que su valor representa el comportamiento de una variable para el periodo de un día, figura 5.15. Esta variable viene dada por `(valor,timestamp)`, donde el `timestamp` es la fecha de un determinado día. Es una medida virtual o estimada. El valor suele calcularse de una de estas dos formas:
  - por medio de alguna función sobre las variables instantáneas durante el periodo de un día
  - el valor es facilitado directamente por el dispositivo indicando el día al que corresponde dicho valor.

- Medida para eventos y alarmas: Un dispositivo físico puede proporcionar medidas sobre las magnitudes que está adquiriendo o midiendo pero también proporcionar información sobre el funcionamiento de dicho dispositivo o comportamiento de las variables que genera. Ejemplo de este tipo de información son: alarmas por apagado, sobrecalentamiento, tensiones fuera de rango, etc. Este tipo de información se conoce como alarmas y eventos y es fundamental que cualquier sistema de gestión energética sea capaz de gestionarlos, esto es, de capturar, almacenar y reportar. Se trata de medidas registradas, figura 5.16.

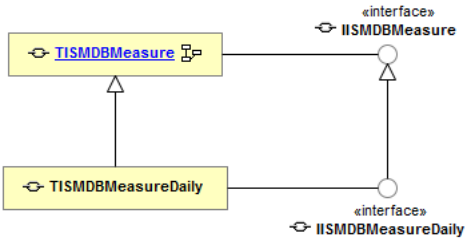

Figura 5.15: Modelado de medidas diarias

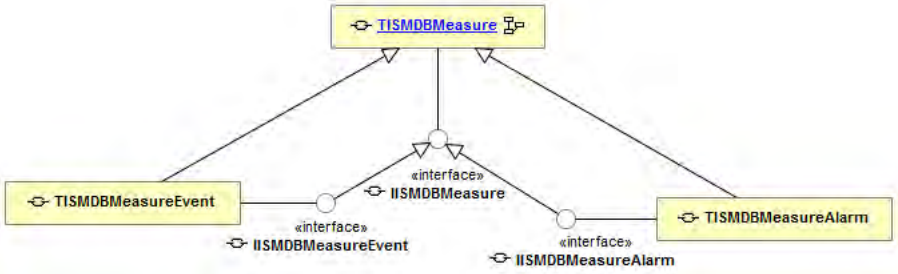

Figura 5.16: Modelado de alarmas y eventos

- Medida global: Una medida global es aquella que dispone de un único valor y por lo tanto dicho valor representa el significado de una variable hasta un momento determinado, figura 5.17. Cada variable de tipo global almacena tanto su valor como su timestamp permitiendo de esta manera disponer de un valor global para cualquier momento dado hasta dicho timestamp. Así, se puede disponer de un histórico de valores globales. El valor de esta variable suele ser función de otros valores diarios, instantáneos y constantes y es una manera muy potente para inferir información sobre conceptos de instalación, dispositivo o canal.

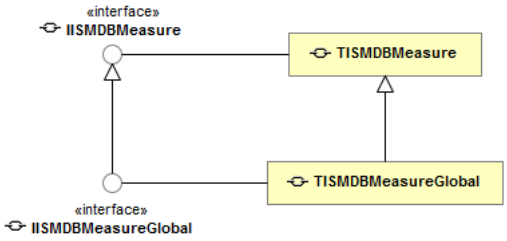

Figura 5.17: Modelado de medidas globales

- Medida instantánea en tiempo real: El marco de trabajo desarrollado además de permitir la adquisición de datos que disponen los dispositivos utilizando la tecnología OPC HDA puede establecer comunicación con un dispositivo e intercambiar información de manera continua utilizando OPC para tiempo real, es decir OPC DA. Para mantener el concepto de medida lo más desacoplado posible y no separar la implementación en sistemas diferentes, estas medidas de tiempo real se han implementando a partir de la clase genérica inicial, figura 5.18. Esto permite abstraer el concepto de medida y reutilizarla en cualquier parte del sistema. Son medidas registradas.

Así, a partir de una clase base que implementa la interfaz apropiada se puede definir los diferentes tipos de medidas anteriormente citadas.

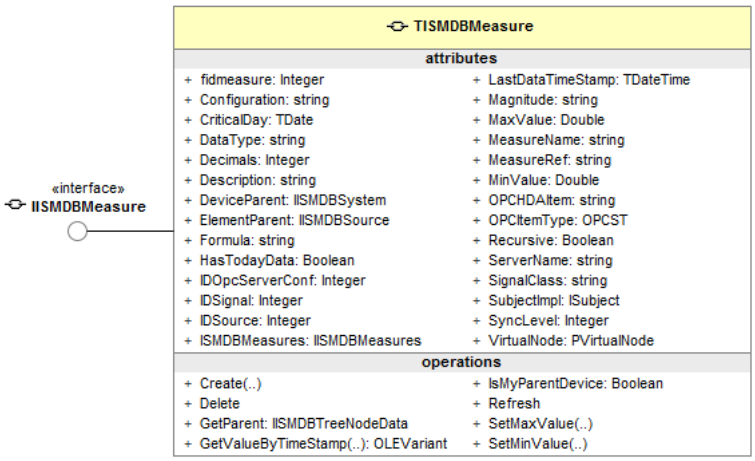

Figura 5.18: Clase para el modelizado de medidas

Aunque los tipos de medidas modelizadas suelen ser las que se necesitarán en la mayoría de los casos, puede verse que es sencillo añadir nuevos tipos de medidas simplemente aplicando los mecanismos de implementación y herencia sobre la estructura de clases e interfaces proporcionada por el marco de trabajo.

5.6.1. Fuente de datos de una medida

El marco de trabajo propuesto permite la asociación real de diferentes tipos de elementos físicos o virtuales, de acuerdo con el esquema que se muestra en la figura 5.19.

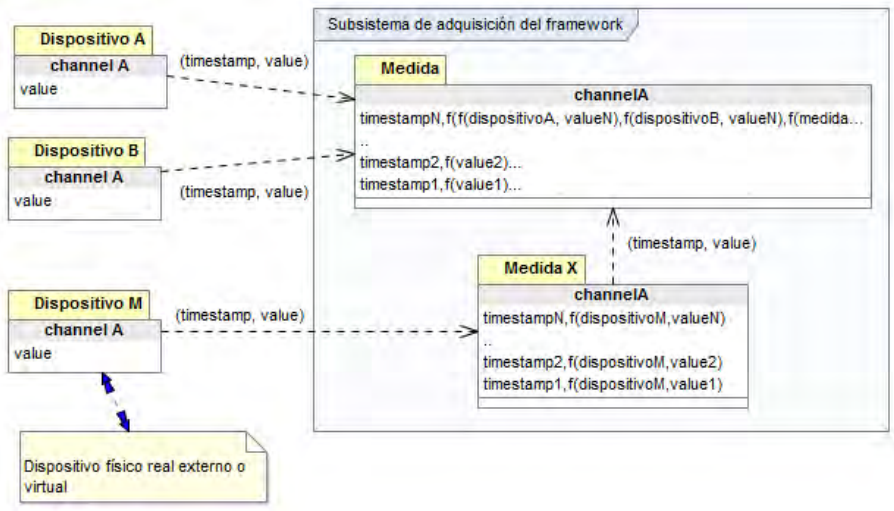

Figura 5.19: Representación real de un canal físico

En este caso la medida es una representación real del canal físico determinado. El subsistema, disponiendo del valor de dicha medida podrá almacenarlo, gestionarlo, publicarlo, etc. Para poder realizar estas acciones, en el marco de trabajo se propone que los valores virtuales asociados a los distintos elementos de un sistema puedan ser generados a partir de scripts lo que permitirá que se puedan utilizar distintas expresiones, aritméticas y/o lógicas, para su definición. El sistema dispone de un intérprete de scripts en varios lenguajes (python, visual basis script, java script y delphi script) y como fuente de datos de una medida se permite cualquier función desarrollada en estos scripts por lo que la función puede incluir expresiones de tipo aritmético y/o lógico y lógica, también es posible usar otras variables obtenidas por un OPC ya que el script las vería como propias. Además se pueden definir nuevas funciones externas (es decir, funciones que están en el marco de trabajo programadas internamente hacerlas visibles desde el script). Esto dará una mayor potencia a este tipo de medidas. Estos scripts pueden ser escritos en cualquier lenguaje que permita la escritura de scripts, ya que serán compilados para su incorporación al modelizado de una planta.

Las siguientes reglas gramaticales especifican cómo pueden ser creados estos scripts para incorporar medidas virtuales en el modelizado de un sistema y cómo se construyen las expresiones para calcular sus valores:

Script → *Sentencia*
| *Script Sentencia*
;

Sentencia → *Asignación*
| if *Condicion Sentencia* [else *Sentencia*]
| *Sentencia Sentencia*
;

Expresión → `medida.v =` *Expresión*
;

Expresión → `medida.r`
| *Expresion Op Expresion*
| - *Expresion*
;

Op → + | - | * | /
;

Condición → `Expresion Oprel Expresion`
| *Condición* AND *Condición*
| *Condición* OR *Condición*
| NOT *Condición*
;

Oprel → < | <= | > | >= | == | <>
;

El sistema de scripts tiene en cuenta la prioridad de operadores del lenguaje en el que se escriba el script. Así, se puede resumir que el valor de una medida virtual se puede calcular utilizando los valores tanto de medidas registradas, como los de

Sistema	<n_sis> sep <n_med>
Instalación	<n_sis> sep <n_ins><n_med>
Grupo	<n_sis> sep <n_ins> sep <n_grupo> sep <n_med>
Sección	<n_sis> sep <n_ins> <n_grupo> sep <n_sec> sep <n_med>
Dispositivo	<n_sis> sep <n_ins> <n_grp> sep <n_sec> sep <n_dis> sep <n_med>

Tabla 5.2: Identificadores de los OPC Items (n: nombre, sis: sistema, med: medida, ins: instalación, grp: grupo, sec: sección, dis: dispositivo)

medidas virtuales previamente calculadas gracias al sistema de evaluación de funciones incorporado en el marco de trabajo.

Cada medida tiene siempre asociado un item OPC que es el que se utiliza para obtener los datos de los dispositivos; estos items son los que se solicitan al servidor cuya configuración OPC tiene asignada cada medida. Los items OPC que usa un servidor tienen una identificación única: están formados por el nombre de la variable que almacena (real o virtual) y una ruta que determina su ubicación dentro del modelo de planta. Como se explicó anteriormente, una planta está formada por secciones que contienen uno o varios dispositivos. Las plantas, a su vez, están contenidas en grupos de instalaciones, que pueden contener también otros grupos de manera recursiva. Todas estas estructuras actúan en la base de datos como contenedores de medidas, cuyo OPC Item depende de su localización, de acuerdo con las reglas que se describen en la tabla 5.2.

En la figura 5.20 se propone una arquitectura que permite modelizar una medida instantánea proporcionando las funcionalidades anteriormente descritas.

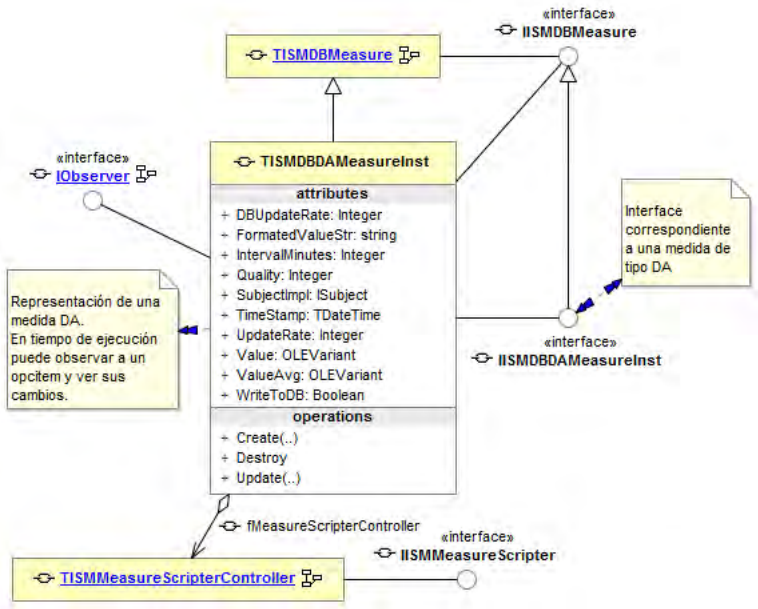

Figura 5.20: Modelizado de medidas calculadas

Esta medida dispone de un controlador asociado a un script que observa a través

del interfaz *IObserver* [7.3] los cambios producidos en el valor de la medida o de su timestamp. En el caso de detectar una cambio lanzará la ejecución del script que actualizará el valor de la medida teniendo en cuenta los valores de las diferentes medidas reales o virtuales asociadas a dicho script.

5.7. Conclusiones

En este capítulo se ha propuesto un modelizado basado en clases e interfaces para un sistema energético completo que permite efectuar consideraciones sobre el modelo tal y como se haría sobre el sistema real. Este modelizado hace posible disponer de una representación lógica de un sistema físico real y da respuesta a la complejidad de detalles que hay que tener en cuenta para este tipo de modelizado.

Se ha hecho también una propuesta para modelizar los elementos que conforman un sistema energético, desde la instalación de manera general hasta las medidas o puntos de información pasando por los diferentes dispositivos así como los posibles niveles de acceso o usuarios que tienen alguna capacidad de gestión sobre la planta.

Se ha propuesto, finalmente, un modelizado adecuado para las medidas, que son la última información y, en definitiva, más importante, ya que son las responsables de disponer de los parámetros de funcionamiento de las instalaciones, y por lo tanto su gestión y su capacidad de configuración son quizás la parte más importante al basarse toda la información y toma de decisiones en el contenido de las mismas.

"La belleza es más importante en informática que en ninguna otra tecnología debido a la gran complejidad del software. La belleza es la defensa definitiva contra la complejidad"

– David Gelernter

6

Capa de almacenamiento y persistencia de la información

6.1. Introducción

Para cada sistema que se quiera desarrollar es necesario almacenar información, no solo de los datos registrados sino también de la configuración de la planta que se gestiona, dispositivos que incorpora, agrupación de instalaciones, perfiles de usuarios, etc. Por ello, es necesario utilizar una base de datos para mantener esa información, tal y como se detallaba en la arquitectura propuesta en el capítulo 3. En concreto, será necesario almacenar:

- Datos: contendrá los datos instantáneos almacenados por los dispositivos y los datos calculados en forma de datos diarios y datos globales.
- Características de los elementos del sistema: incorporará todos los datos referentes a las plantas, los grupos, los inversores.
- Eventos: almacenará los diferentes eventos y alarmas que se produzcan en los diferentes dispositivos.
- Roles de usuario y permisos, para controlar los accesos a los distintos sistemas. Para ello, una posible organización de esta información es la siguiente:
    - Datos de cada usuario (nombre, contraseña, rol asociado a cada planta, etc..)

- Tipos de roles: consulta de una o más plantas y administrador. El rol de consulta servirá para dar acceso a la información de las plantas (eventos, datos instantáneos, diarios, información de las plantas, etc.). El rol de administrador permitirá ver la información de todas las plantas y gestionar los usuarios.

Para hacer persistente el modelizado de toda esta información se propone la utilización de sistemas gestores de bases de datos compatibles con el estándar SQL92. De esta manera y al hacerlo de manera genérica, la base de datos de almacenamiento puede ser elegida en cada caso para permitir una mayor flexibilidad. En este caso, todo el desarrollo se ha realizado para su utilización con el SGBD de fuentes abiertas PostgreSQL.

Desde un comienzo se pensó en la utilización e incorporación de marcos de trabajo para la persistencia de clases disponibles de manera libre pero el rendimiento no era el apropiado ya que la frecuencia de almacenamiento de la información podría llegar a ser muy exigente debido, en muchos casos, a la cantidad de datos que provienen de los dispositivos. Por esta razón se ha generado un modelo entidad-relación propio y asociado a cada clase. Esta modificación no entra en conflicto con la facilidad y flexibilidad de poder extender las clases con nuevos atributos persistentes al primar un alto grado de exigencia en la organización del código fuente del marco de trabajo desarrollado.

6.2. Modelo de persistencia del modelizado de usuarios y perfiles de acceso

La información correspondiente a los usuarios, los roles y la correspondencia con cada instalación dispone de una persistencia en base de datos a través del modelo entidad-relación mostrado en la figura 6.1.

Hay que destacar que en este modelo propuesto cada usuario puede disponer de un rol determinado para cada instalación y a su vez cada rol tendrá disponible una serie de servicios que estarán disponibles dentro del marco de trabajo. Con este modelizado, el sistema puede almacenar dicha información y recuperarla de manera sencilla cuando se invoca cada instancia de cada clase.

En la figura 6.2 puede verse una implementación con la utilización de la clase de definición de usuarios donde dicha información se almacena y recupera de la base de datos.

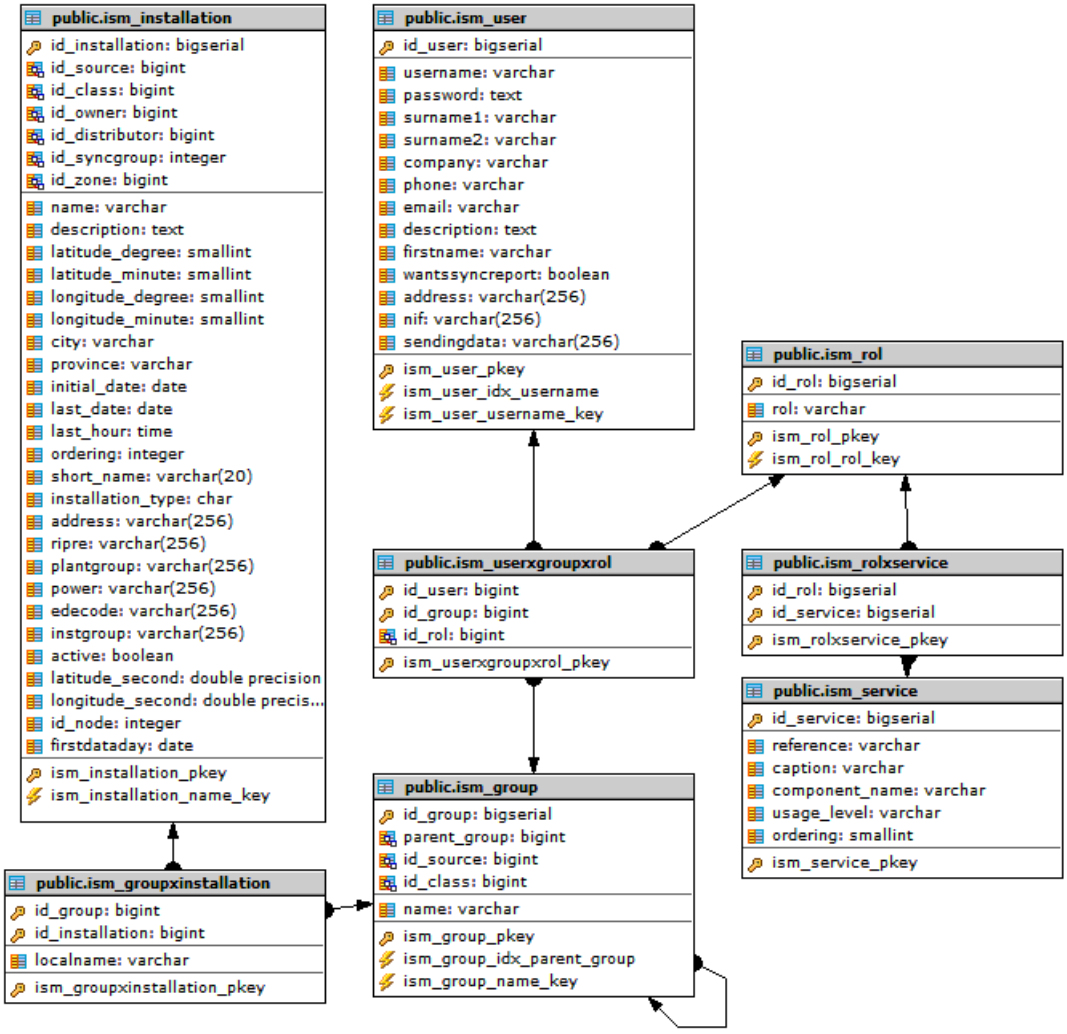

Figura 6.1: Modelo de almacenamiento en base de datos de usuarios

6.3. Modelo de persistencia del modelizado de instalaciones

Para proporcionar persistencia de almacenamiento de las instalaciones se ha desarrollado el modelo que se muestra en la figura 6.3.

En este caso, es necesario almacenar la información de cada instalación y la forma de agrupación en grupos y secciones de dispositivos. El modelo soporta las relaciones para establecer una integridad referencial y así permitir operaciones y modificaciones en cascada.

En la figura 6.4 se muestra la aplicación de la instanciación de grupos y en la figura 6.5 su asociación con instalaciones utilizando las clases proporcionadas por el marco

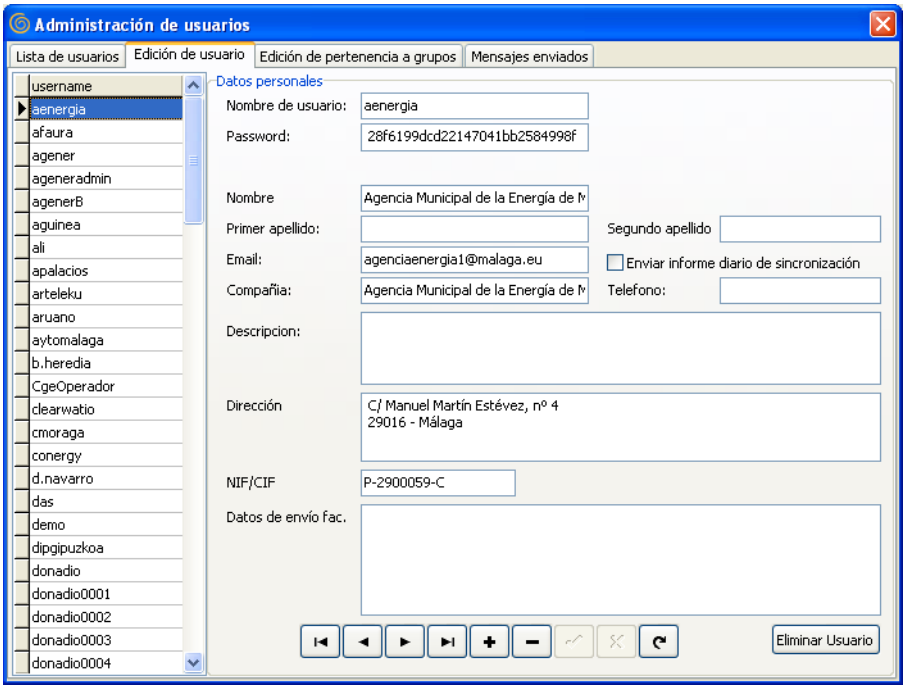

Figura 6.2: Interfaz para la administración de usuarios

de trabajo y el modelo entidad relación para su almacenamiento y recuperación de la base de datos utilizada.

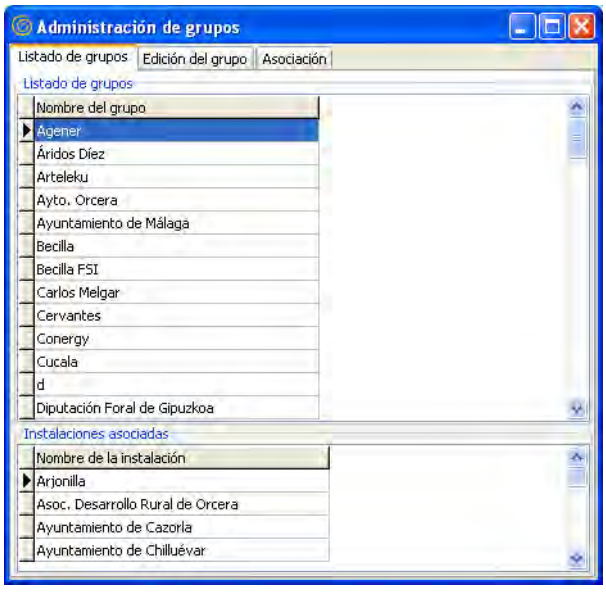

Figura 6.4: Implementación de grupos de instalaciones

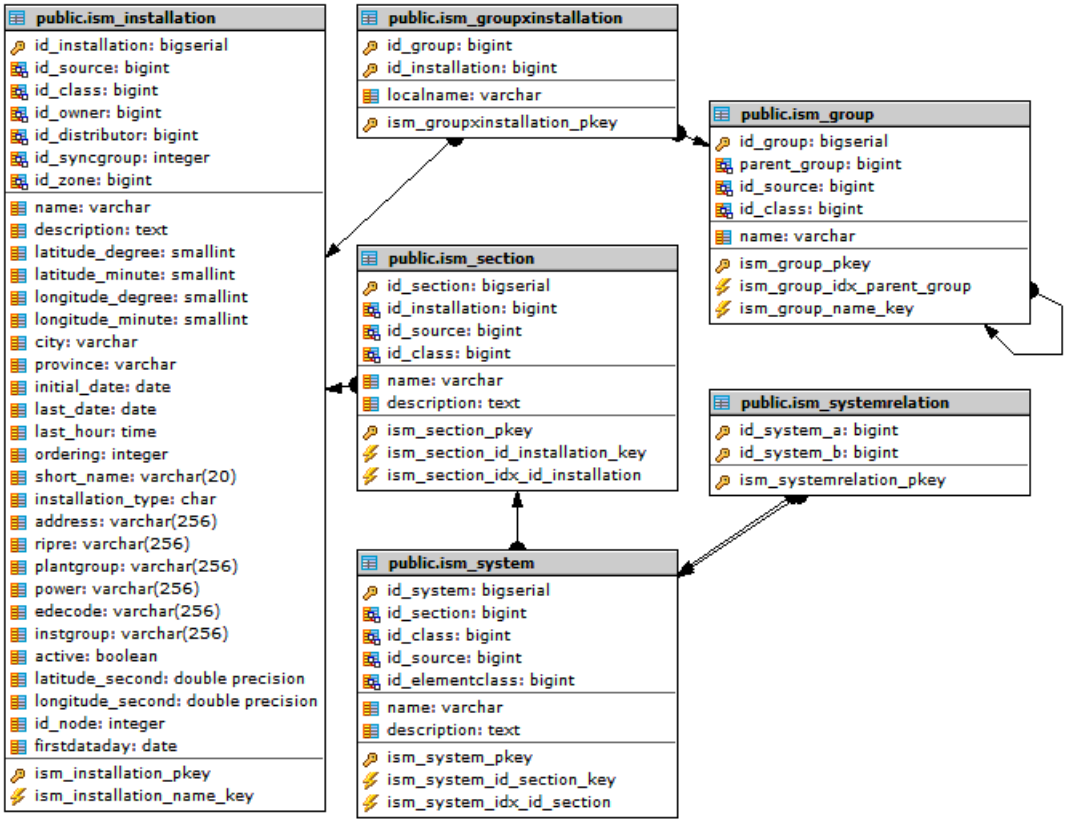

Figura 6.3: Modelo de almacenamiento en base de datos de instalaciones

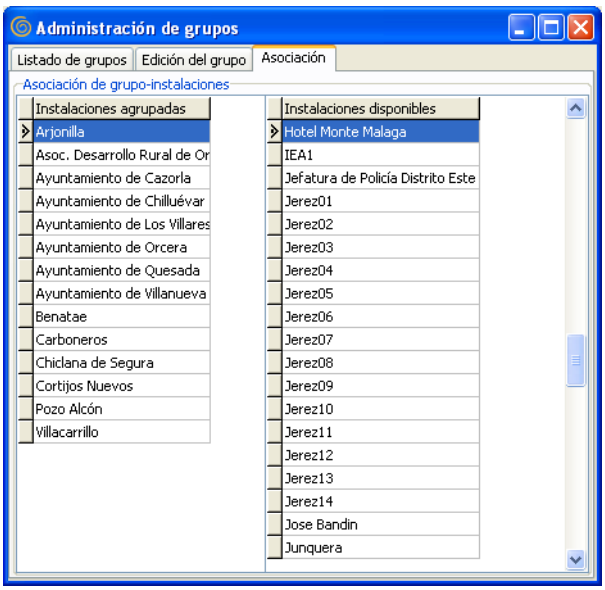

Figura 6.5: Agrupación de instalaciones

6.4. Modelo de persistencia del modelizado de dispositivos

Para poder almacenar la información referente a los dispositivos modelizados de una planta determinada se propone el uso del modelo entidad relación que se muestra en la figura 6.6.

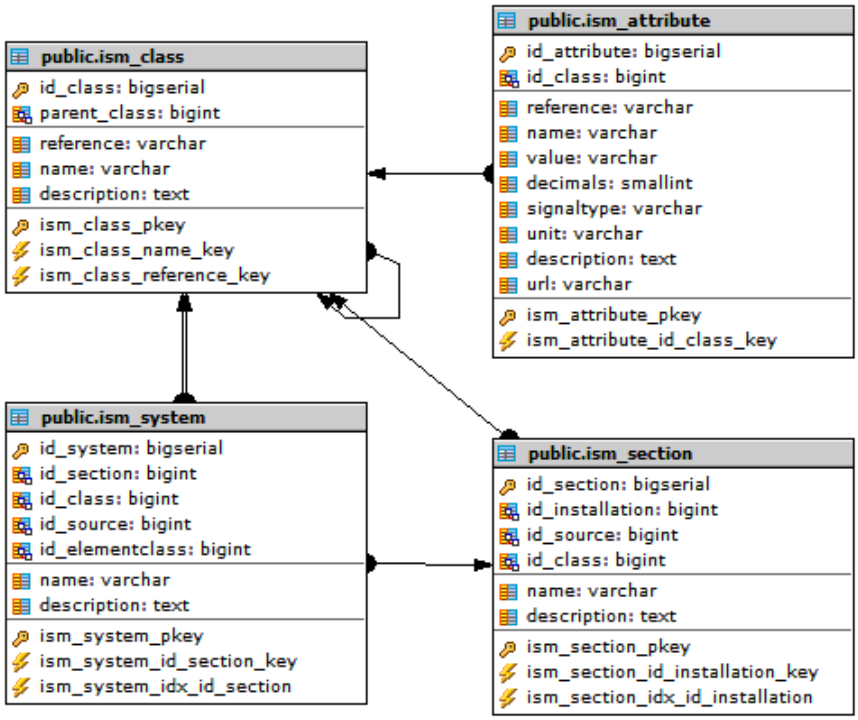

Figura 6.6: Modelo de almacenamiento en base de datos del modelizado de dispositivos

Según esta propuesta, un dispositivo estará asociado a un grupo de dispositivos o sección y el dispositivo será de un tipo o clase determinada. El asociar los atributos o medidas de un dispositivo a una clase de dispositivo en lugar de una instancia permite que se pueda reutilizar el modelizado de un dispositivo tantas veces como apariciones reales de dicho disposito ocurran en una planta real sin tener que volver a definir sus canales.

Las *medidas* y los *dispositivos* asociados son elementos que es necesario modelizar muy frecuentemente. Por ello, el marco de trabajo proporciona un repositorio de *medidas* y *dispositivos* perfectamente modelizados. Este repositorio se puede utilizar cuando se definen y añaden nuevos dispositivos durante el modelizado de una planta, sin necesidad de volver a definirlos cada vez. Además, es fácil crear nuevos dispositivos a partir de los que ya existan. Los dispositivos que se pueden incluir son de diferentes clases que, por ejemplo, en dispositivos para sistemas de energía solar podrían ser: inversores, generadores, contadores de energía, células para medir radiación, piranóme-

tros, sensores de temperatura y sistemas de adquisición de datos locales, entre otros. Este repositorio permitirá la inclusión de nuevos dispositivos y medidas relacionados con los predefinidos lo que supondrá que el diseño de una nueva aplicación de gestión energética se pueda hacer de forma más rápida a partir de ese repositorio.

En la figura 6.7 puede verse una librería de dispositivos modelizada en base a clases de dispositivos y con canales ya definidos. Al encontrarse este repositorio en la propia base de datos, se pueden reutilizar los dispositivos definidos desde cualquier aplicación que haga uso de este marco de trabajo para la construcción de aplicaciones de gestión energética.

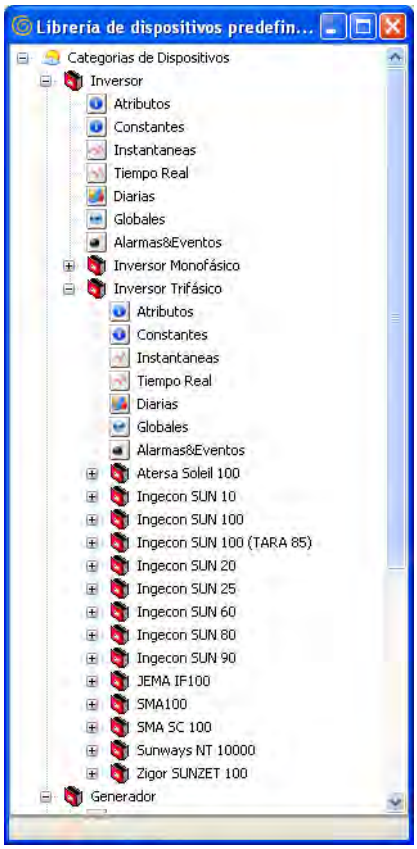

Figura 6.7: Librería de dispositivos modelizados

6.5. Almacenamiento de la información de los canales de los dispositivos

Uno de los retos a la hora de desarrollar el marco de trabajo ha sido la forma de almacenar gran cantidad de datos, en algunos casos, a alta frecuencia. La información que proviene de los dispositivos puede ser información histórica o información en tiempo real. En el caso de información histórica, el almacenamiento puede realizarse a baja frecuencia debido a que se dispone de la información en el dispositivo durante un

relativo largo periodo de tiempo. Sin embargo en casos de información en tiempo real, los dispositivos generan información que es necesario almacenar cuanto antes. Además es necesario que tener en cuenta que la información a almacenar puede ser de diferente tipo por lo que no es posible tratarla toda de la misma manera.

De manera formal se puede ver la información a almacenar de la manera que se describe a continuación.

Si se denominan todos los dispositivos de una instalación dada como $d_1, ..., d_k$ se tendrá que:

$$\forall d_i \exists (d_i, t_i, v_i) \tag{6.1}$$

donde t_i representa un instante determinado en el tiempo y v_i un valor asociado a dicho d_i en ese instante t_i.

Cada valor puede ser de un tipo determinado como números enteros, valores flotantes, valores booleanos o incluso cadenas de caracteres, por lo que su almacenamiento tendrá que realizarse de manera diferente sin sacrificar rendimiento. Por ello, la solución adoptada ha sido la disposición al marco de trabajo de tablas específicas para el almacenamiento de dicha información, figura 6.8.

De esta manera, cada canal conoce el tipo de información y almacenará y recuperará dichos valores de la tabla apropiada logrando una alta frecuencia de almacenamiento en la base de datos.

6.6. Mecanismo de sincronización para el almacenamiento de los canales de los dispositivos

El poder disponer del estado de una instalación en cualquier momento dado es una ventaja importante para estudiar su funcionamiento actual e incluso para poder inferir algunos parámetros sobre el comportamiento futuro. En muchas ocasiones es sencillo que se disponga de toda la información pero ocurren situaciones que en las que no se puede controlar cuando se quiere recuperar la información de todos los dispositivos y por alguna razón esto no es posible. Es entonces cuando se hace necesario disponer de mecanismos para tener la capacidad de recuperar la información de cualquier estado en el que ha estado la planta.

Existen por supuesto, algunas limitaciones como que ocurran problemas o cortes de comunicación entre los dispositivos que proporcionan valores en tiempo real, pero si el dispositivo dispone de una memoria interna o dichos dispositivos son registradores de datos es posible implementar algún mecanismo inteligente para disponer de la información de la forma más coherente y precisa posible.

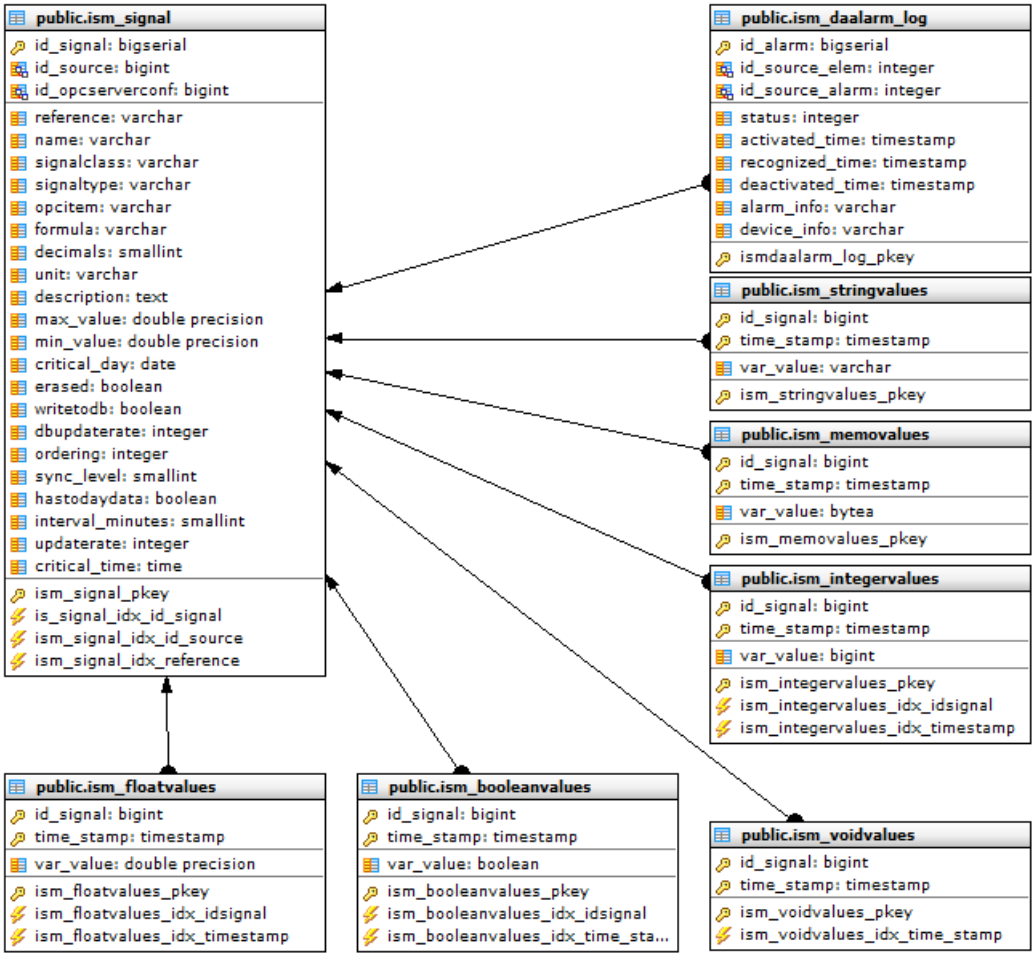

Figura 6.8: Almacenamiento de datos de diferentes tipos medidas

En el marco de trabajo desarrollado se ha implementado de manera transparente, un algoritmo de sincronización lo suficientemente versátil para poder reanudar la descarga y mantener la información lo más fiable posible ante causas que en algún momento no permitan descargar la aplicación. Estos casos pueden llegar a ser muy comunes cuando se realizan conexiones a dispositivos a través de tecnologías móviles o dichos dispositivos se encuentran en puntos geográficamente complicados causando problemas en las comunicaciones.

El proceso de sincronización se divide en dos niveles, partiendo de la cola de sincronizaciones, que es el nivel más externo, hasta llegar a la sincronizacion de un día para una medida concreta, que corresponde al nivel mas interno. Así, el proceso se puede desglosar de la siguiente manera:

- A un nivel superior se utiliza el algoritmo del sincronizador. Este es el algoritmo de funcionamiento del programa encargado de atender las peticiones de sincronización de las instalaciones según una cola de planificación (Fig. 6.9).

- En el siguiente nivel estará el algoritmo de sincronización de una instalación. Este es el algoritmo al que llama el sincronizador por cada instalación que encuentra en la cola de sincronizaciones.
- A continuación se encuentra el algoritmo que sincroniza una medida para el intervalo de tiempo especificado.
- Finalmente, en el último nivel, se encuentra la sincronización de un día concreto para una medida concreta, donde se realizaran las peticiones a los servidores OPC responsables de descargar la información de cada dispositivo.

El algoritmo del sincronizador se corresponde con el bucle principal del programa encargado de la sincronización de las instalaciones que haya en el sistema. El algoritmo que se propone para la tarea de sincronización principal se describe en la figura 6.9.

Este algoritmo se basa en la cola de petición de sincronizaciones, y puede estar ejecutándose en segundo plano continuamente, con idea de atender las peticiones que se vayan planificando manualmente o de forma automática desde el planificador. Así, cuenta con un temporizador, el cuál le indicará de forma periódica al sincronizador que realice una búsqueda en la tabla de sincronizaciones pendientes. En el momento que encuentre instalaciones pendientes, la sincronización se realiza según el siguiente orden:

1. Prioridad asignada
2. Fecha y hora de la petición

Una vez encuentra instalaciones pendientes, continúa buscando en la tabla y sincronizando hasta que no queden más instalaciones pendientes, momento en el que esperará a que se cumpla el temporizador para realizar una nueva búsqueda.

Para las instalaciones pendientes, el proceso de la sincronizacion que se propone es el siguiente:

1. Marcar la hora en la que se empieza a atender la petición.
2. Sincronizar la instalación según el algoritmo correspondiente a la *Sincronización de una instalación*, que se explicará más adelante.
3. Marcar la hora de finalización de la sincronización.
4. Aplicar a la instalación los filtros integrados en el sistema para la detectar e inferir incidencias y comportamientos extraños.
5. Volver a buscar una nueva instalación por sincronizar en la cola. En caso de que no queden instalaciones pendientes, se repite la búsqueda cada vez que se cumpla el temporizador.

La cola de sincronizaciones sirve tanto para introducir peticiones de sincronización como para llevar un historial de las sincronizaciones de cada planta. en esta cola se ordenan las peticiones por prioridad y a igual prioridad se toma la fecha de petición menos reciente. Cada vez que se hace una sincronización de una instalación se obtiene además el resultado de la sincronización, de acuerdo con los tres siguientes posibles resultados:

1. La instalación se sincronizó por completo
2. Faltaron algunos datos por sincronizar
3. No se pudo sincronizar ninguna medida

Por otra parte, para la sincronización de una instalación se propone el algoritmo que se muestra en el diagrama de flujos de la figura 6.10.

Con este algoritmo se realizan las sincronizaciones de todas las medidas asociadas a cada planta en cualquiera de sus niveles (planta, sección, sistema). Para esta sincronización es necesario establecer algún orden de manera que se minimicen los accesos a los dispositivos que se van a sincronizar.

En el siguiente nivel, para sincronizar las medidas instantáneas se propone un algoritmo que se encarga de recuperar todas las medidas que correspondan a un mismo día en un único paquete; el diagrama de flujo de este algoritmo se muestra en la figura 6.11.

La idea es dividir las medidas en los diferentes servidores OPC que se puedan encontrar con medidas instantáneas para cada instalación. Las inserciones en la base de datos de los datos recuperados se realizan mediante hebras, por lo que la sincronización no finaliza hasta que se acaban todas las hebras. Para ello el algoritmo implementa un contador de hebras al que se accede mediante una sección crítica y se espera hasta que ese contador llega a 0 o se supera un tiempo máximo de espera también predefinido en el algoritmo.

Por último, para la sincronización de un día para una medida se propone el algoritmo que se muestra en el diagrama de flujos de la figura 6.12.

Con este algoritmo se puede conseguir realizar la sincronización de una medida los días que se especifiquen. En caso de que no se hubiesen podido descargar anteriormente los datos de un dispositivo, la utilización del concepto de día crítico hará que se descarguen dichos datos en el momento que estén disponibles teniendo seguridad que dada una fecha, o no será posible descargar los datos porque se han perdido debido a que se haya llenado la memoria o se descargarán cuando haya comunicación con el dispositivo.

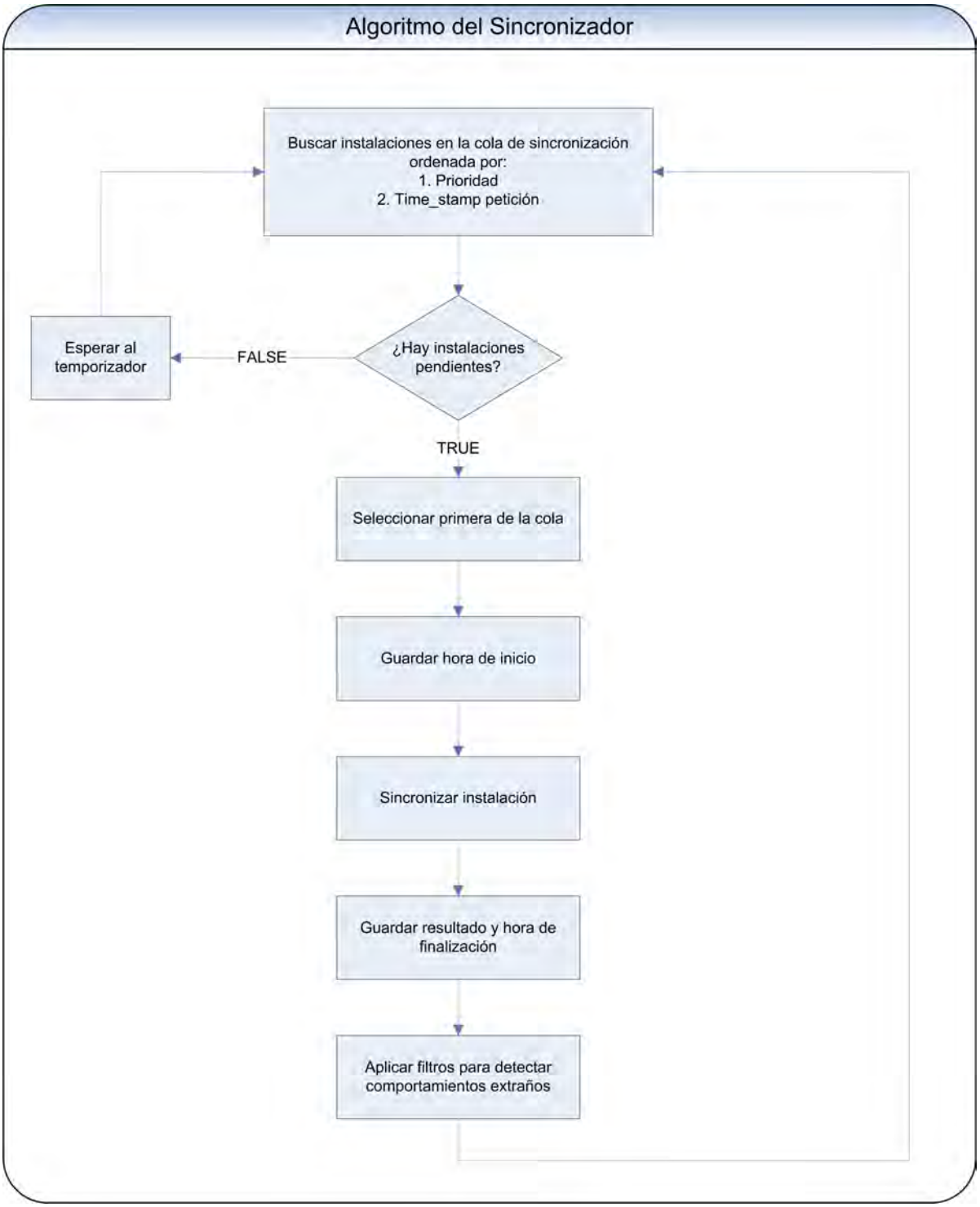

Figura 6.9: Algoritmo para la sincronización con las plantas

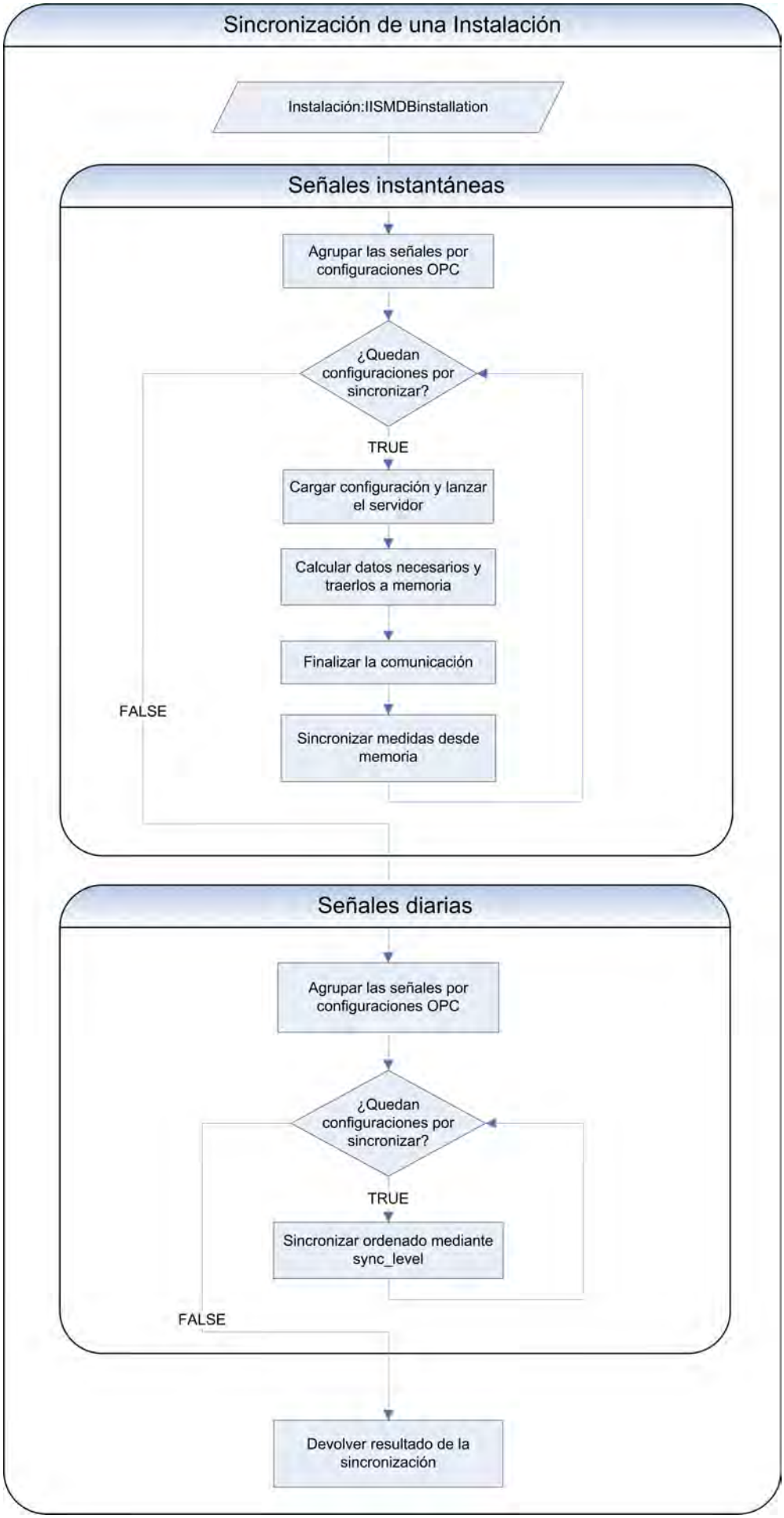

Figura 6.10: Algoritmo para la sincronización de una instalación

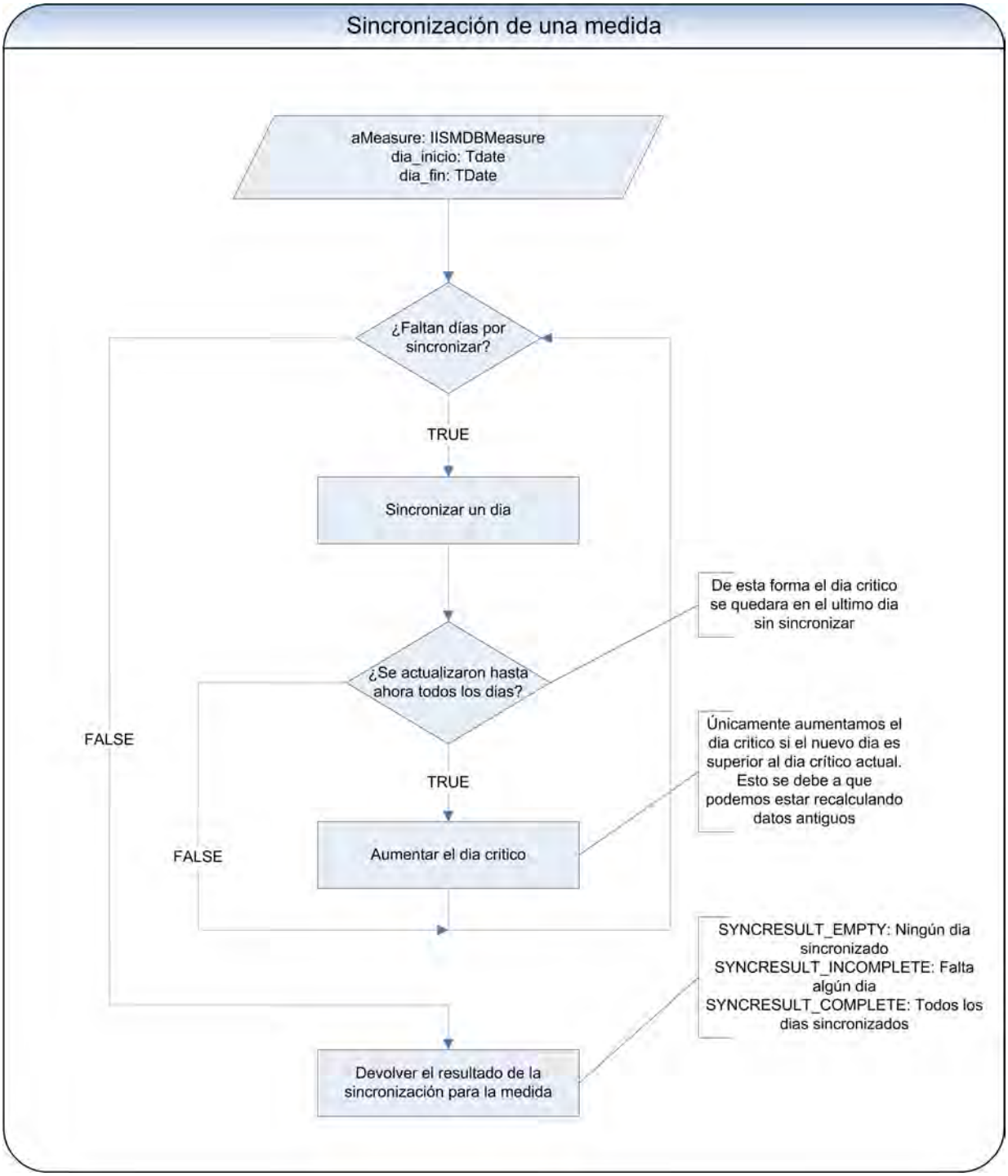

Figura 6.11: Algoritmo para la sincronización de una media

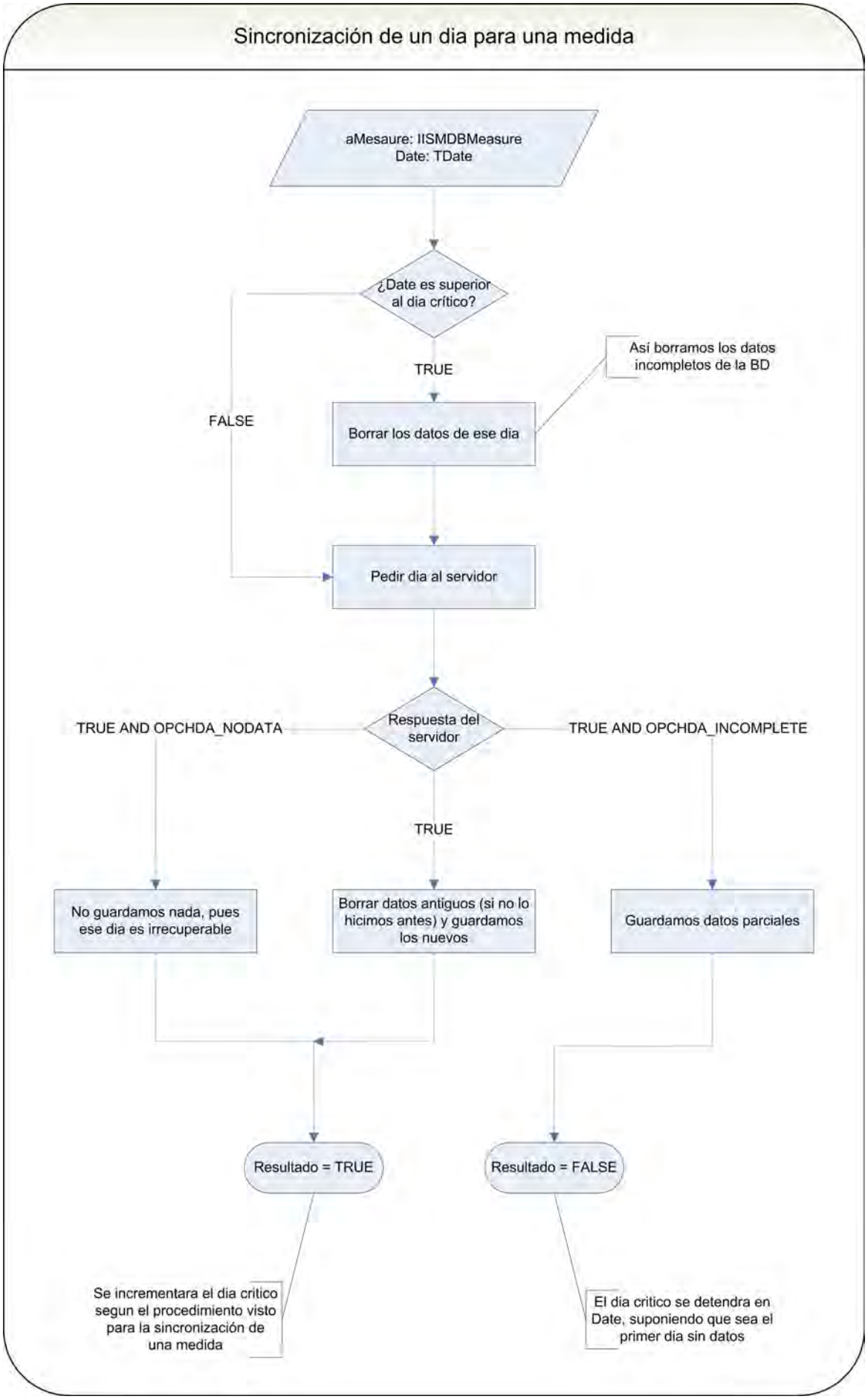

Figura 6.12: Algoritmo para la sincronización de una media en un día

6.7. Conclusiones

Una vez modelizada la instalación, la información de la misma, tanto de su modelo como del contenido de los datos que representan su funcionamiento, debe poder ser almacenada y recuperada de manera eficiente. La utilización en este caso de una capa específica para dotar de persistencia a la información en bases de datos y la utilización de sistemas genéricos para realizarla permite disponer de múltiples opciones para la utilización de los sistemas que más convengan en cada caso. Opciones como la utilización de sistemas gestores de bases de datos basados en fuentes abiertas, como PostgreSQL o MariaDB, o incluso licenciados como Oracle, abren un abanico de opciones sin tener una dependencia a priori con el sistema a utilizar.

Por otra parte, se han expuesto una serie de algoritmos que garantizan disponer en todo momento de la máxima información posible de los dispositivos, lo que se hace independientemente de que en los casos reales, multitud de problemas relacionados con protocolos, fallos de comunicación, sistemas de comunicación ineficientes o con ruido pueden ser despreciados, garantizando una coherencia en la información y en los datos. Esto es realmente importante si se quiere tener una idea de cómo se comporta el sistema o incluso a la hora de inferir información o realizar evaluaciones o predicciones de las instalaciones que se están gestionando.

"Cualquier bug lo suficientemente avanzado es indistinguible de una funcionalidad"

– Rich Kulawiec

7

Capa de soporte a la implementación

7.1. Introducción

Dentro de un marco de trabajo aparecen multitud de relaciones y referencias entre objetos que pueden hacer que la complejidad e interrelación entre los mismos se haga muy complicada, causando además un problema al disponer de un sistema altamente acoplado. Además, en los lenguajes de programación de mayor rendimiento, los sistemas de tipado fuerte hacen que haya que estar continuamente asociando los tipos de los objetos y la inclusión de nuevas referencias han de pasar por fuerza porque dichos tipos sean compatibles.

7.2. Sistema de selección genérica

Para evitar estos casos, en este trabajo se propone que la comunicación entre clases se realice traspasando una interfaz genérica, figura 7.1. De esta manera, solo en destino es necesario comprobar si dicha instancia implementa la interfaz necesaria y evita pasar y definir tipos que sean compatibles para conseguir con éxito la compilación del sistema.

Así, cualquier clase que quiera disponer de la funcionalidad de poder facilitar a otra una referencia a cualquier otra clase podrá hacerlo implementando el interfaz ISelection. Se proporciona además una clase que implementa esa interfaz para que cualquier otra clase pueda implementar por delegación de interfaz el interfaz ISelection.

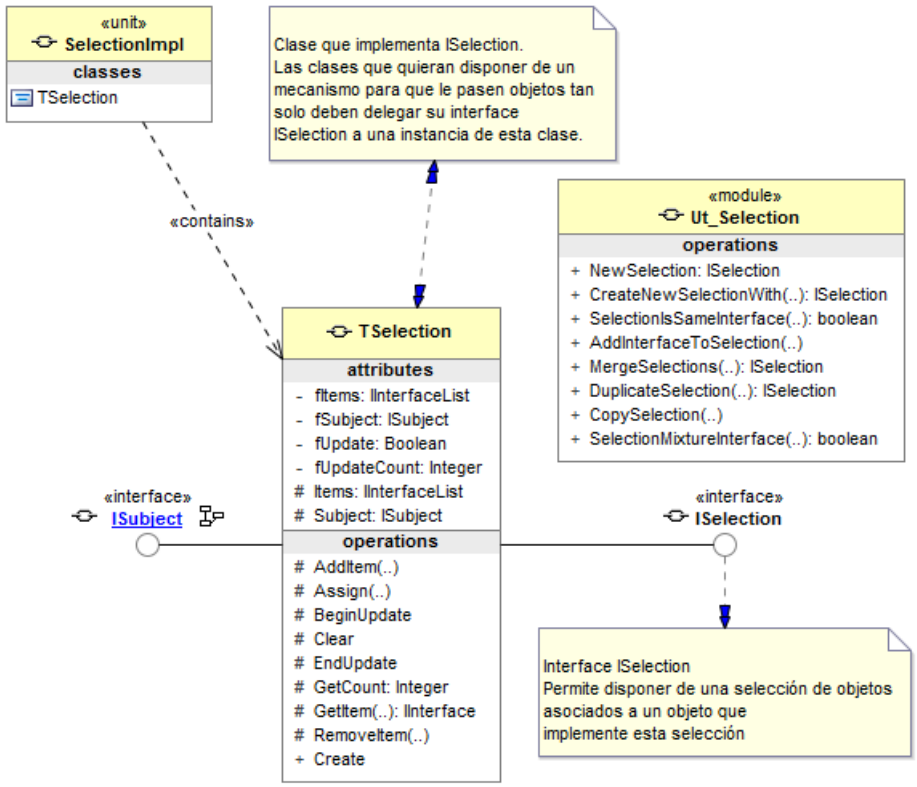

Figura 7.1: Selección genérica

Una instancia que, por ejemplo, tenga un conjunto de objetos de tipo dispositivo o un conjunto de usuarios no necesitarán declarar esa lista de objetos de manera diferente, sino que utilizando la misma implementación guardará los interfaces de esas clases y cuando otro objeto tenga la necesidad de utilizarlos simplemente mirará si los objetos implementan la interfaz deseada y si es así tendrá los objetos que desea.

Esta solución se puede realizar en los lenguajes de programación modernos haciendo uso de los *templates* pero tienen como inconveniente que en tiempo de preprocesamiento, antes de la compilación, el código se duplica para cada tipo diferente por lo que hace que los programas crezcan y no sean igual de eficientes que la solución que se plantea en este trabajo.

7.3. Sistema Sujeto-Observador

En marco de trabajo desarrollado, la comunicación entre las clases se realiza bajo eventos. Con los eventos es posible eliminar los algoritmos iterativos ya que la ejecución de los métodos de las clases se ejecutan bajo ciertas circunstancias. Realizar una implementación de mecanismos de eventos en cada clase puede llegar a ser muy costosa y compleja por lo que el marco desarrollado facilita la interacción entre clases usando eventos, proporcionando clases e interfaces reutilizables para esta tarea.

Para implementar esta funcionalidad se ha realizado una modificación del patrón sujeto-observador utilizando interfaces. Por un lado se cuenta con el interfaz *ISubject* proporcionando, a la clase que lo implemente, poder ser observado y avisar sobre cualquier evento a la clases que lo observen. Se proporciona además una clase que implementa dicha interfaz (Fig. 7.2) para facilitar su inclusión por agregación y delegación de la implementación de dicha interfaz siguiendo los mecanismos explicados en el apartado 3.4.4..

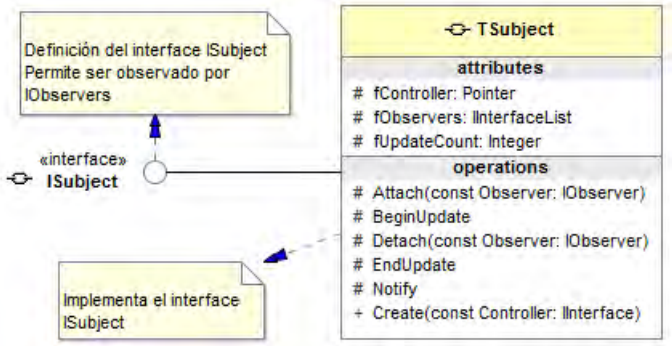

Figura 7.2: Interfaz sujeto observable ISubject

Por otro lado se tiene el interfaz *IObserver* que permite, a las clases que lo implementan, observar las clases que implementan el interfaz *ISubject* y recibir notificaciones de las mismas, figura 7.3.

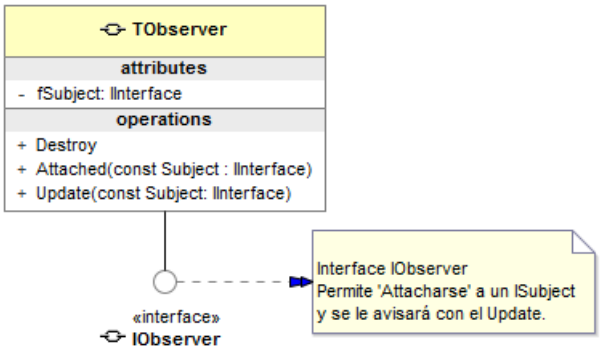

Figura 7.3: Interfaz IObserver

De esta manera, no es necesario estar comprobando variables ni estados de las clases de manera síncrona. En el momento que se crean las diferentes instancias de las clases, se establecen los enlaces entre sujetos y observadores y los cambios producidos en ciertas instancias producirán eventos sobre las instancias que los observan disponiendo así de un mecanismo de notificación de eventos totalmente asíncrono. Con esta propuesta se consigue reducir mucho la carga del marco de trabajo y se proporciona una manera elegante y clara de producirse flujos de información y eventos.

7.4. Sistema de acciones y comandos

La interacción con el usuario es siempre una funcionalidad deseable en cualquier sistema por muy automatizado que se encuentre. Existen multitud de soluciones para separar la parte de los datos, la lógica de negocio y la interfaz del usuario como los patrones basados en Modelo-Vista-Controlador que buscan la separación de conceptos y reutilización de código en la arquitectura del software.

Debido a que el marco de trabajo desarrollado está basado en interfaces software, se ha visto la oportunidad de poder conectar dichas definiciones de procedimientos o propiedades con el interfaz gráfico de una manera sencilla registrando la asociación de definición de interfaz con acciones del interfaz de usuario. De esta manera es sencillo incorporar a las aplicaciones desarrolladas con este marco de trabajo acciones asociándolas simplemente a los objetos a los que se quiere dotar de acciones o comandos determinados, figura 7.4.

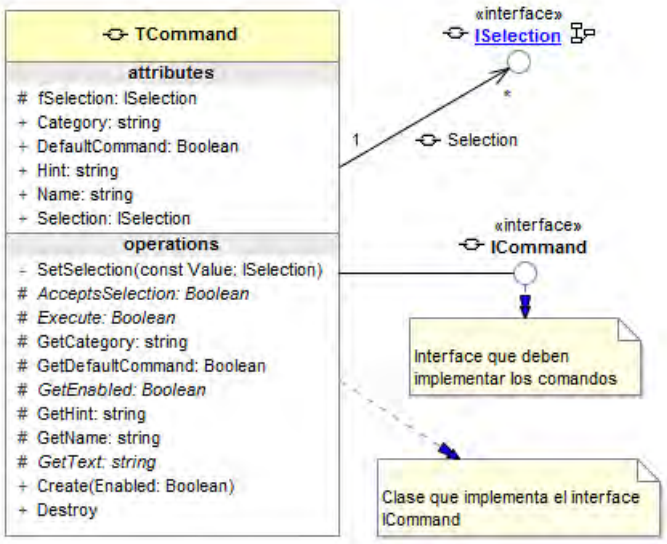

Figura 7.4: Modelado de comandos

El interfaz *ICommand* facilita la creación dentro del sistema de comandos que podemos asociar a cualquier instancia como se muestra en la figura 7.5.

A su vez, cada comando puede estar agrupado en diferentes categorías permitiendo al interfaz de usuario mostrar los comandos agrupados para una mejor experiencia del usuario, figura 7.6.

En esta figura puede verse cómo integrar un menu contextual con el sistema de comandos para que los muestre de manera gráfica. Así, dado un objeto que soporte una selección de instancias de objetos, se le aplicará la lista de comandos que son compatibles con las interfaces que soporte.

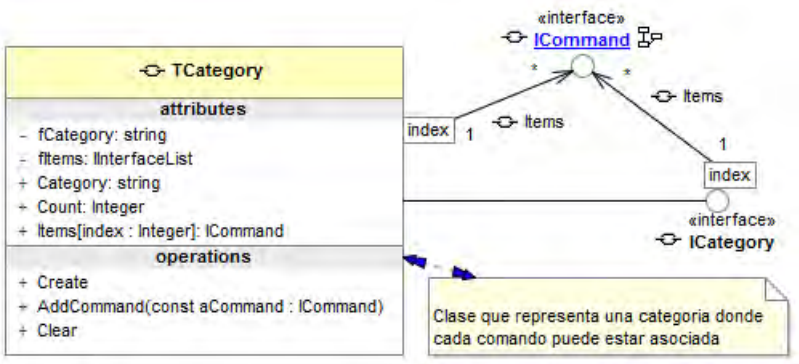

Figura 7.5: Modelado de categorías de comandos

En la figura 7.7 puede verse cómo el marco de trabajo aplica automáticamente, sobre los objetos seleccionados, los comandos disponibles en el sistema, permitiendo incluso diferentes representaciones visuales de los mismos. En este caso un menú contextual y un panel de acciones para representar las mismas acciones disponibles.

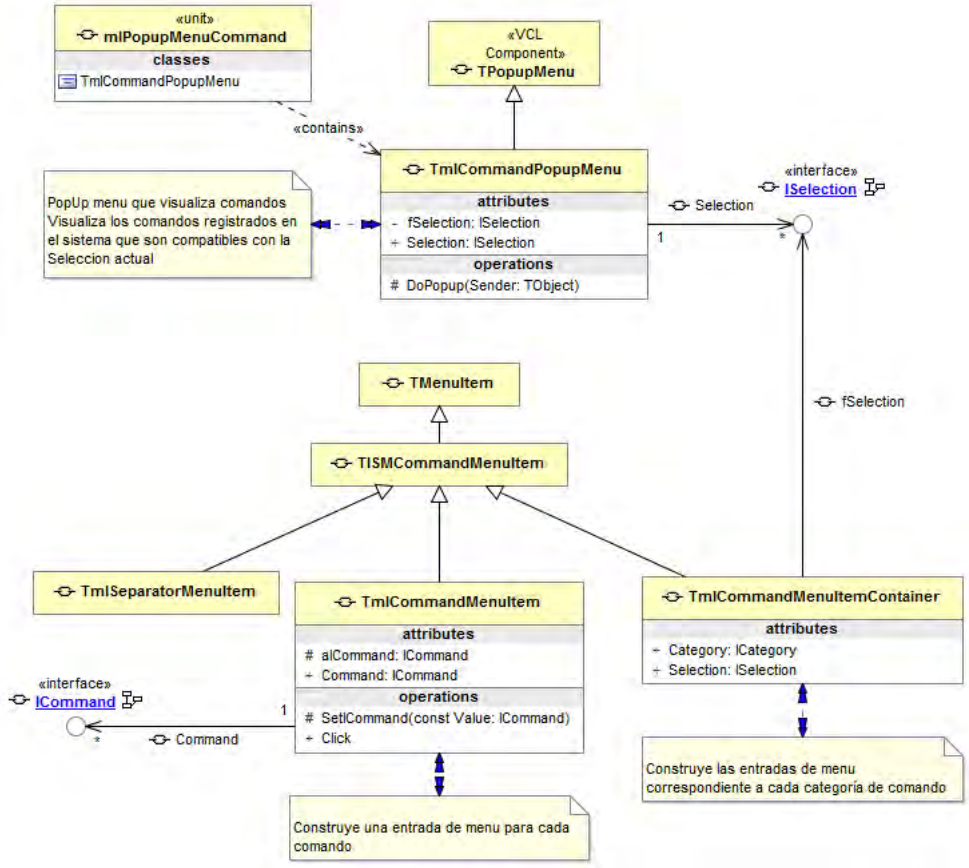

Figura 7.6: Sistema de comandos

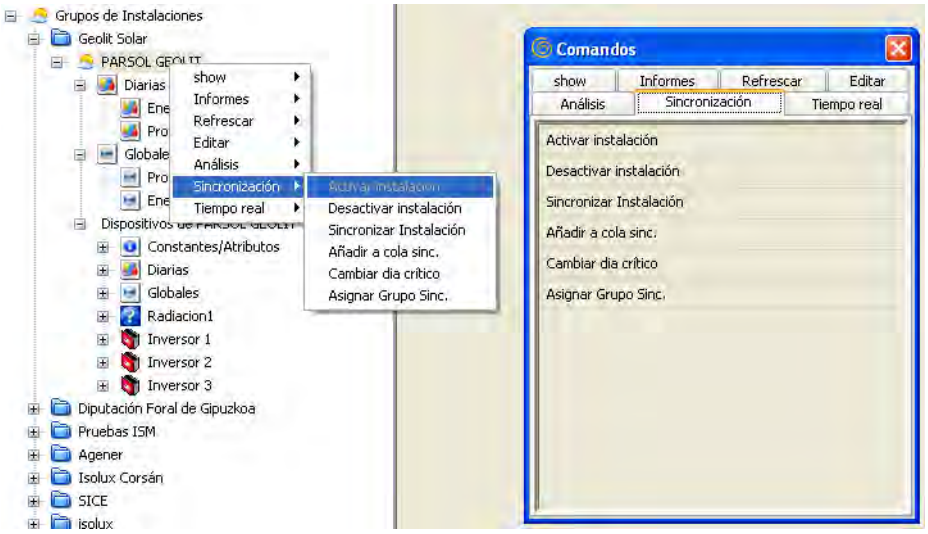

Figura 7.7: Sistema de comandos

7.5. Sistema dinámico de propiedades

Cada vez que se define una clase con programación orientada a objectos o incluso en otros paradigmas de programación, es necesario conocer la estructura de la misma y sus atributos. Esto hace que en todo momento se disponga de una dependencia entre los tipos, atributos y métodos y las demás clases que se relacionan con ellas y que conozcan esta estructura en tiempo de compilación. Una desventaja evidente es la poca flexibilidad que ofrece esta condición ya que la inclusión de nuevos atributos en las clases ya definidas pasa por modificar en tiempo de diseño los desarrollos.

Para dar respuesta a esta situación, en este trabajo se proporciona al marco de trabajo de un mecanismo de extensión en tiempo de ejecución, lo que es una ventaja y mejora añadida al permitir que las clases se construyan y modifiquen en el momento de su utilización.

En la figura 7.8 pueden verse los interfaces que pone a disposición el marco de trabajo para que las clases que se utilicen puedan se extendidas en tiempo de ejecución. Cada clase puede hacer uso de una lista de propiedades *IProps* y definir nuevas propiedades o atributos con diferentes niveles de acceso (solo lectura, solo escritura o de lectura/escritura).

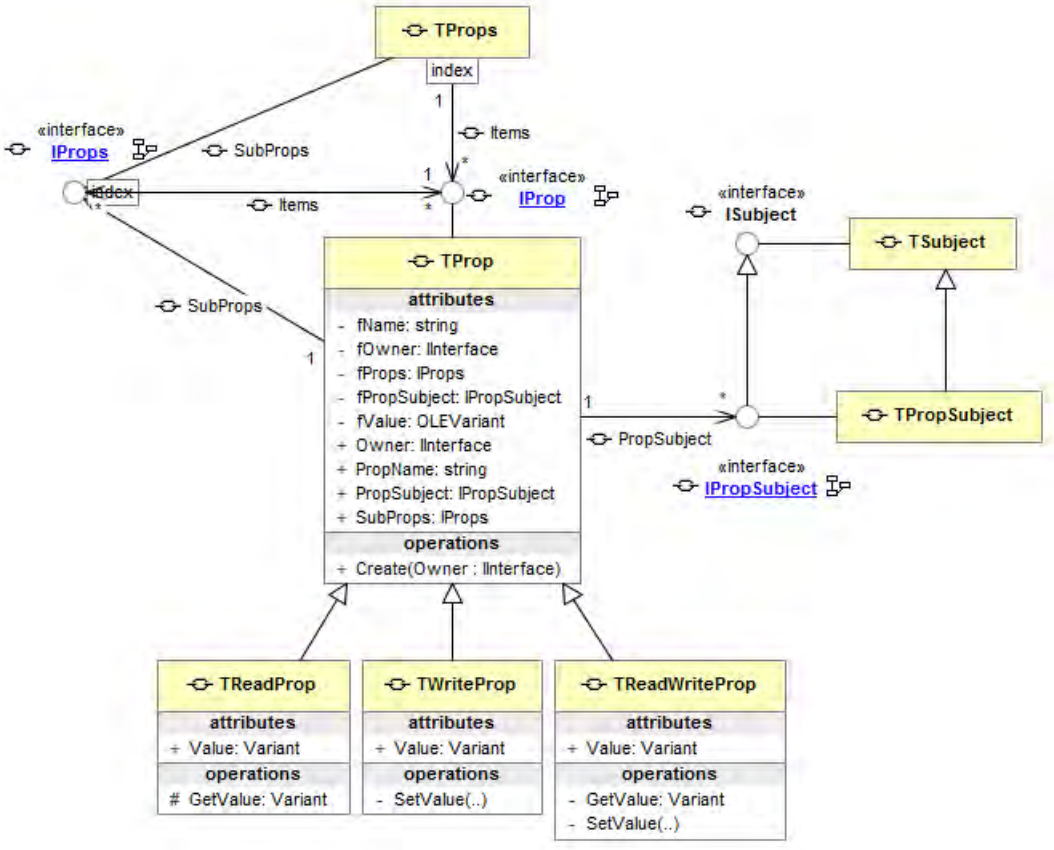

Figura 7.8: Sistema dinámico de propiedades

Disponer de esta funcionalidad proporciona una flexibilidad que elimina la limitación de tener que modificar las clases codificadas, ya que permite añadir nuevos atributos a las mismas en tiempo de ejecución y las clases relacionadas o dependientes pueden consultar dichos atributos en el momento apropiado.

Cada propiedad dispone a su vez de la capacidad de ser observada (véase 7.3) por lo que se añade además la funcionalidad de ser notificado por el cambio de los valores de dichos atributos.

7.6. Sistema dinámico de configuración

La arquitectura de clases anteriormente presentada permite modelizar un servidor OPC para cada dispositivo con la condición de conocer el protocolo de intercambio de información con el mismo. Sin embargo existe todavía una funcionalidad importante que debe ser implementada y está relacionado con los parámetros de configuración del protocolo y el dispositivo. Será, por ejemplo, necesario facilitar al sistema los parámetros de conexión físicos, si la comunicación es con un dispositivo por medio de un puerto serie, o el número de teléfono, si es a través de una línea telefónica.

Para facilitar esta funcionalidad, se ha tenido en cuenta que el marco de trabajo proporcione una serie de clases de ayuda así como un flujo de comportamiento para indicar y facilitar los parámetros de configuración de cada servidor OPC que se desea desarrollar, figura 7.9.

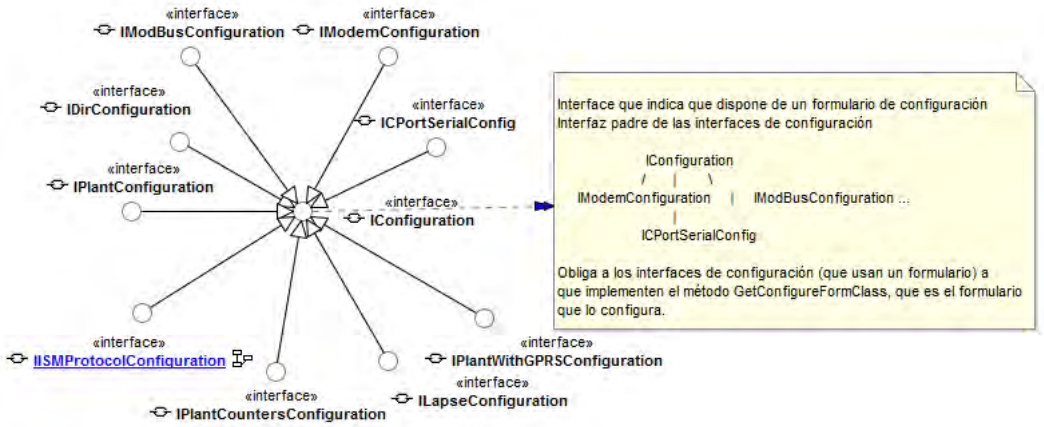

Figura 7.9: interfaz para la implementación del sistema de configuraciones dinámico

Así, para cada caso que se deba disponer de la capacidad de proporcionar parámetros de configuración, se heredará del interfaz IConfiguration. En la figura 7.10 se muestra cómo proporcionar los parámetros de configuración de un dispositivo que necesita configurar su puerto de comunicaciones serie.Como es una situación muy común, se ha creado un servidor OPC genérico que permite la comunicación con dispositivos por el puerto de comunicaciones serie, al que se ha denominado TCustomOPCSerialServer.

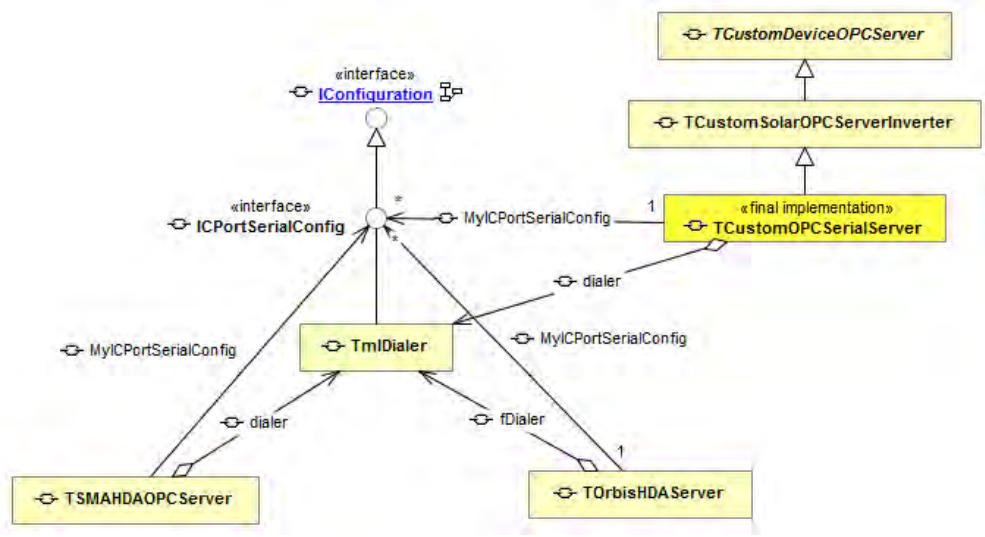

Figura 7.10: Configuración del puerto de comunicaciones serie

Este clase tendrá un componente que se encargará de la gestión directamente con el puerto de comunicaciones, en este caso TmlDialer, e implementará además el interfaz IConfiguration, que es un componente que puede ser externamente configurado.

Cuando el sistema encuentra una clase que implementa esta interfaz buscará el formulario existente que permite configurar dicha clase y lo lanzará bajo petición del usuario, figura 7.11.

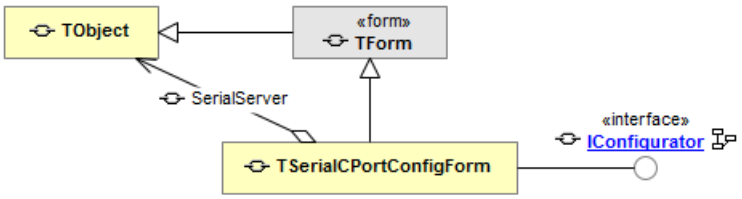

Figura 7.11: Configuración del puerto de comunicaciones serie

En este caso un SerialCPortConfigForm, figura 7.12, permitirá configurar dicha instancia y se encargará a su vez de proporcionar las funcionalidades de persistencia en disco para almacenar o recuperar la configuración proporcionada.

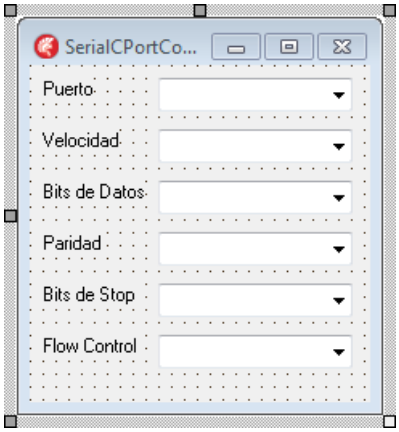

Figura 7.12: Formulario para la configuración del puerto de comunicaciones serie

De esta manera, para cada caso particular de cada dispositivo que deba permitir incorporar parámetros de configuración, existirá una manera sencilla y unificada de proporcionar un formulario de configuración y un interfaz que realice el trabajo de configuración de la clase. Por ejemplo para facilitar la configuración de un dispositivo cuya conexión se realice a través de una línea telefónica como puede verse en 7.13.

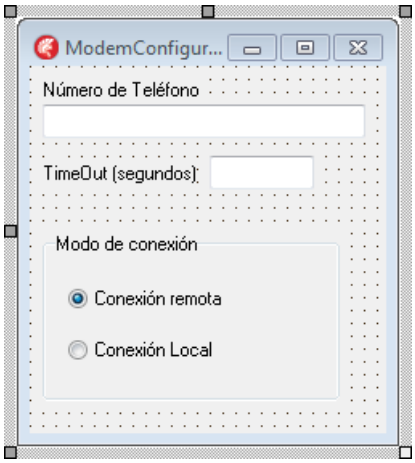

Figura 7.13: Formulario para la configuración del model telefónico

Una vez se disponga de los formularios de configuración de cada clase que intervenga en el servidor OPC y dado que la clase superior de la jerarquía propuesta implementa el comportamiento de un contenedor de interfaces de usuario de configuración, solo quedará indicar desde las clases específicas que el marco de trabajo incluya dichos formularios en el interfaz de usuario (Fig. 7.14).

Las clases anfitrionas de la jerarquía disponen de un interfaz de usuario que servirá como contenedor de los intefaces de configuración que deseemos añadir como puede verse en la figura 7.15.

En la figura 7.16 se ha configurado un servidor OPC que dispone de una clase para el control del puerto serie y otra para las comunicaciones por IP. Será necesario por

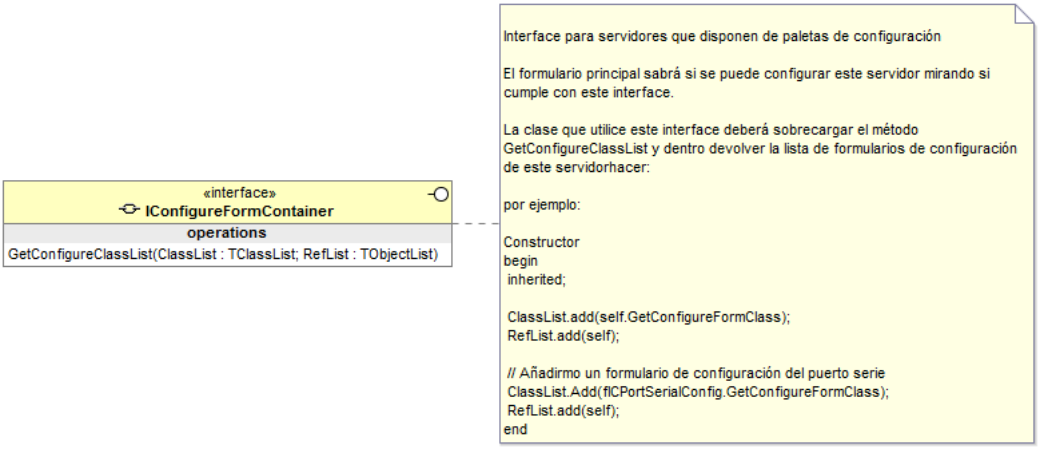

Figura 7.14: interfaz para contenedor de formularios de configuración

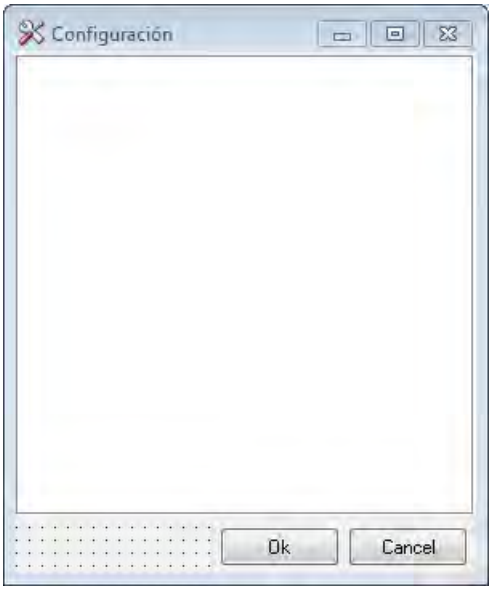

Figura 7.15: Formulario contenedor de formulario de configuración

tanto disponer de los formularios de configuración de cada una de esas clases y una vez añadidas puede observarse como el contenedor albergará los interfaces de usuario de ambas clases de manera automática.

Toda la información de configuración se almacena en el registro del sistema o en ficheros locales a la aplicación que lo está utilizando y por lo tanto es posible arrancar cualquier aplicación e inyectarle la configuración que que se quiera para que sea utilizada en su momento, figura 7.17.

Esta es un funcionalidad muy útil ya que en muchas ocasiones será necesario utilizar diferentes parámetros de configuración para aplicaciones que se comportan de manera similar. Por ejemplo, en el caso de los servidores OPC que deben conectarse con dispositivos iguales pero que se encuentran conectados con parámetros diferentes como direcciones IP o números de teléfono determinados.

Figura 7.16: Formulario contenedor de formulario de configuración del puerto serie e Internet

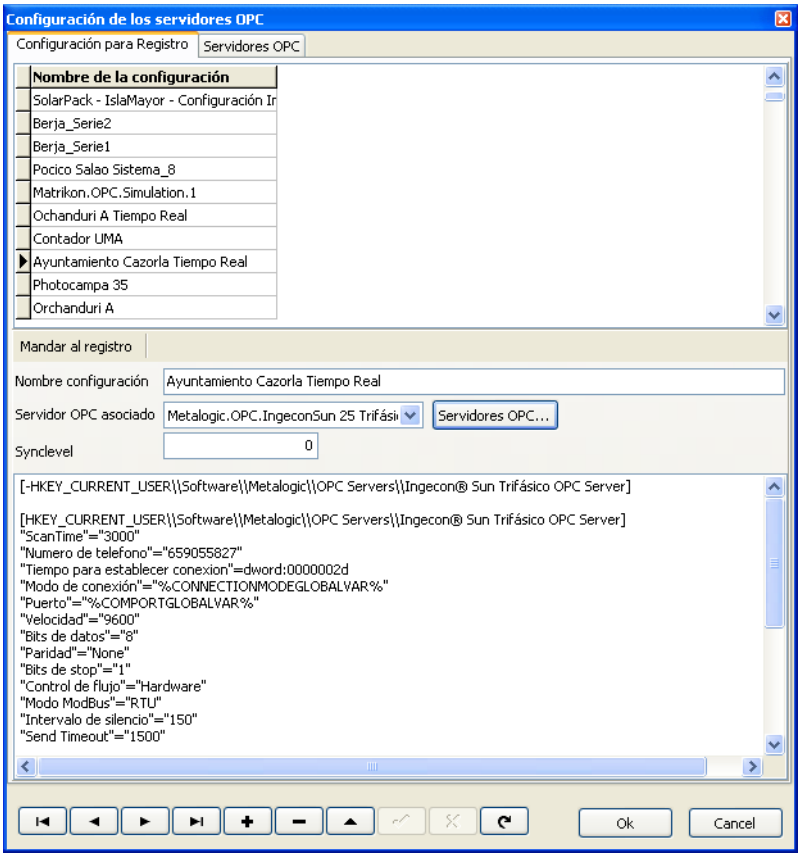

Figura 7.17: Inyección de parámetros de configuración en tiempo de ejecución

7.7. Conclusiones

En cualquier desarrollo de software, es necesario disponer de elementos que, si bien no tienen una función reconocida e identificada respecto al modelo o dominio del problema, permitan servir como apoyo o soporte a la implementación de las clases e interfaces determinadas.

La disposición por parte del marco de trabajo, al usuario que lo utilice, de patrones de diseño que proporcionen mecanismos habituales para la relación de clases e interfaces dentro de un sistema es algo muy deseable, al permitir una mayor flexibilidad para conseguir unos resultados finales apropiados.

El sistema de eventos proporciona mecanismos para conectar clases y permitir el flujo de notificaciones dentro del marco de trabajo entre cualquiera de sus capas. El sistema de acciones y comandos permite definir, asociar y ejecutar algoritmos desde cualquier punto del marco de trabajo. La disposición de un sistema de propiedades dinámico permite la extensión de las clases sin la necesidad de recompilación del código. Disponer además de un sistema de configuración dinámico proporciona la capacidad de particularización de partes del marco de trabajo de manera externa permitiendo adaptarse a diferentes situaciones o situaciones diferentes respecto a las instalaciones a controlar y monitorizar.

"La mejor forma de predecir el futuro es implementarlo"
–David Heinemeier Hansson

8

Modelos para gestión de un sistema de energía solar fotovoltaica

8.1. Introducción

En este capítulo se proponen modelos para la evaluación y predicción del funcionamiento de sistemas energéticos basados en la utilización de modelos de aprendizaje automático. Se presenta también la validación que se ha hecho de estos modelos utilizando datos reales para un tipo de sistema energético no convencional.

Como se ha comentado en un capítulo anterior de esta tesis, desde hace unos años, se ha producido un importante incremento del número de instalaciones de producción de energía a partir de fuentes renovables debido, entre otros motivos, a las políticas de impulso a la diversificación de fuentes energéticas y fomento de la utilización de energías renovables. De entre estas instalaciones, las de gran potencia cuentan con personal especializado para las tareas de evaluación, mantenimiento y gestión de la plantas.

Sin embargo, en las instalaciones de pequeña y mediana potencia no está justificado el coste económico que supone mantener personal especializado para la realización de este tipo de tareas. Por ello, en los últimos años, los fabricantes de los componentes de este tipo de instalaciones, fundamentalmente los fabricantes de inversores, están suministrando programas que solo permiten la monitorización in situ o remota de este tipo de instalaciones, pero no se incluyen rutinas que permitan la evaluación del funcionamiento de las mismas ni la gestión de cara a su integración en las redes

convencionales.

En general, para asegurar un buen funcionamiento de estos sistemas y una correcta integración de los mismos en la red eléctrica hay que dar una adecuada respuesta a dos problemas:

- Evaluar el funcionamiento del sistema, partiendo de los parámetros que se registran en mismo.
- Hacer una predicción a corto plazo de la energía que el sistema va a producir.

En cada sistema energético se recogen valores de varios parámetros en escalas temporales que pueden variar desde 1 a 30 minutos, durante todos los días. Así, al cabo de cierto tiempo, la cantidad de información que se tiene de cada planta puede llegar a ser importante, por lo que sería muy valioso poder sistematizar de alguna manera el tratamiento y análisis de esta información registrada para que pudiese realizarse de forma automática su gestión sin necesidad de contar con expertos.

Por una parte, para resolver el primer problema planteado, el de la evaluación de este tipo de plantas, se han propuesto varios modelos, la mayoría obtenidos a partir del ajuste de los datos experimentales de los subsistemas que integran las plantas o bien a partir de modelos físicos más o menos aproximados de las mismas. Para cada planta, dependiendo de sus parámetros de diseño, se utiliza un único modelo para evaluar su funcionamiento, como por ejemplo los propuestos por: [YCW+04], en el que se presenta un modelo para la gestión de una planta energética híbrida (eólico, solar y baterías como sistema de almacenamiento) basado en la utilización de lógica fuzzy.

En general, estos modelos funcionan bastante bien en determinadas condiciones, pero suelen ser bastante inexactos para condiciones de funcionamiento no previstas en su diseño. Por ejemplo, en el caso de sistemas cuya fuente de energía es la radiación solar, los modelos no suelen funcionar muy bien para días en los que hay presencia de nubes y, como consecuencia, la variabilidad de la fuente energética es alta.

Además, estos modelos de evaluación han sido estimados de forma general, sin tener en cuenta las caracterísiticas de cada instalación, y suelen ser fijos durante todo el periodo de operación del sistema. Sin embargo, se sabe que los parámetros de un sistema pueden cambiar a lo largo de su vida útil, por diversos motivos, como puede ser envejecimiento de los sistemas de generación, degradación de los componentes, presencia de suciedad, etc. Por ello, sería importante poder contar con modelos que permitan tener en cuenta características propias de cada instalación y que permitan también integrar de manera sencilla los posibles cambios que ocurran en los sistemas a lo largo de su vida útil.

Por otra parte, el segundo problema planteado está directamente relacionado con las normativas de cada país, que están obligando a que las compañías productoras de

electricidad mediante fuentes no convencionales, faciliten con antelación los datos de producción de sus sistemas. En este sentido, por ejemplo, el mercado de electricidad en España, operado por Red Eléctrica, cambió de un sistema centralizado a un sistema competitivo en 1998.

En este mercado global, para conseguir una penetración real de los sistemas de producción de electricidad a partir de fuentes no convencionales, como son los sistemas de energía solar, se hace necesario suministrar información de la previsión de producción horaria de energía eléctrica con un día de antelación. La desviación de estas previsiones, para plantas de determinado tamaño, está penalizada económicamente, por lo que disponer de modelos que sean capaces de hacer estas predicciones de manera precisa es fundamental. Sin embargo, esta predicción es difícil en el caso de las instalaciones de energía solar, tanto térmica, termosolar como fotovoltaica, debido a la dependencia de la producción con la variable climatológica radiación solar. El comportamiento de esta variable puede cambiar drásticamente de un día a otro, incluso en el mismo día, por los cambios que puede haber en la atmósfera. Así, aunque es posible conocer con mucha precisión la radiación que llega a la parte exterior de la atmósfera -radiación extraterrestre, una vez que esta penetra en la atmósfera hay diferentes factores que afectan la cantidad de energía que alcanza la superficie de tierra. Entre estos factores el más importante es la presencia de nubes en la atmósfera, que tiene un cierto componente aleatorio.

Por tanto, el desarrollo de herramientas que permitan la supervisión, evaluación del funcionamiento y predicción de la producción de sistemas energéticos que utilicen fuentes de energía no convencionales puede contribuir a la optimización de estos sistemas y a un mayor desarrollo de los mismos.

En este trabajo se proponen modelos que están basados en técnicas estadísticas y aprendizaje automático como medio para representar las relaciones observadas entre las variables independientes y la variable dependiente que se definan en cada problema para los que se propone su utilización. Los dos problemas que se tratan son:

- La evaluación del funcionamiento de un sistema energético en el que intervengan componentes que presenten cierta aleatoriedad, como son los sistemas de energía solar fotovoltaica, debido a la componente no determinista de la fuente de energía de los mismos.

- La predicción a corto plazo de la producción de energía de un sistema que presente alguna componente que no sea totalmente determinista entre las variables que influyen en este parámetro, como son, nuevamente, los sistemas de energía solar fotovoltaica.

Un sistema de energía solar fotovoltaica es una instalación energética que consta de varios subsistemas comunes como puede verse en la figura 8.1.

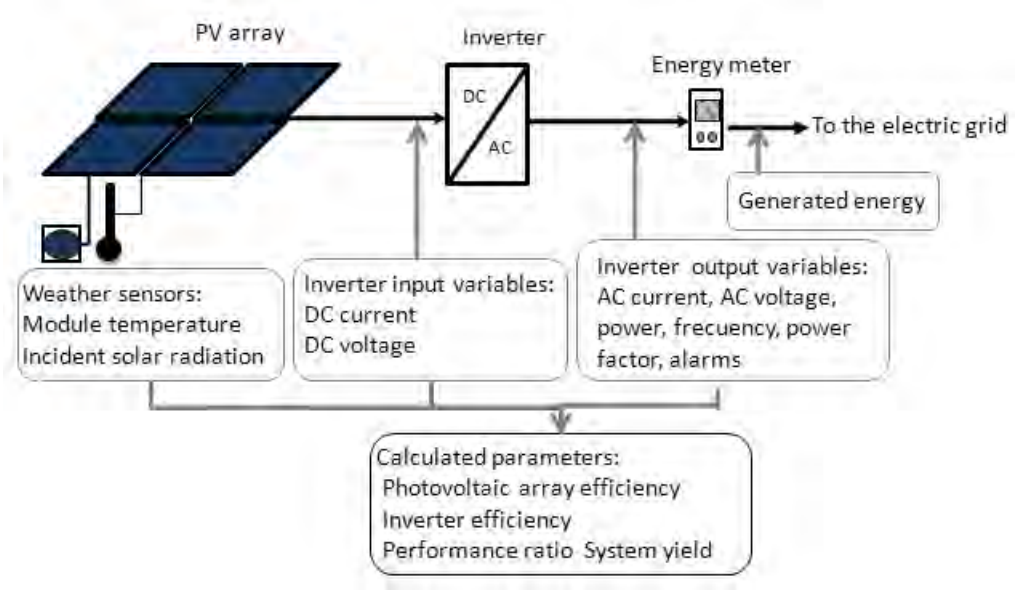

Figura 8.1: Esquema de una planta de energía solar fotovoltaica

Las medidas que con mayor frecuencia se registran en las instalaciones fotovoltaicas, fundamentalmente en los inversores, estaciones meteorológicas y sensores de temperatura son: tensión en continua, intensidad en continua, tensión alterna, intensidad alterna, potencia activa, radiación global, radiación directa, temperatura ambiente, temperatura módulos y frecuencia. Con estas medidas de cada dispositivo se tiene una imagen tanto en el momento instantáneo como de su funcionamiento pasado si se utilizan datos almacenados.

A partir de estas medidas, se obtienen las medidas virtuales o estimadas que serán calculadas por el marco de trabajo usando los mecanismos que se han expuesto en un capítulo previo. Estas medidas pueden ser de dos tipos: medidas diarias y medidas globales. Las medidas diarias son aquellas que describen el comportamiento de un sistema en un día específico, mientras que las medidas globales son las que tienen información de todo el sistema desde que se inició su monitorización. Además de estas información se cuenta con las medidas instantáneas de cada momento en el que se registran por cada dispositivo de la instalación.

8.2. Evaluación de sistemas de energía solar fotovoltaica

Para el caso de los sistemas de energía solar fotovoltaica se han utilizado las medidas diarias que permiten evaluar el comportamiento de este tipo de sistemas propuestas por [Be95].

Partiendo de estos trabajos, se propone, por una parte, la utilización de un índice de rendimiento o productividad diario (daily final yield) y, por otra, la utilización de balances diarios de energía (producida y estimada). El primer parámetro es muy adecuado para comparar distintas instalaciones que tengan tamaños diferentes y que trabajen en diferentes condiciones meteorológicas.

El rendimiento diario final, $Y_{f,d}$, se define como la energía útil diaria producida por el sistema por cada kW_p instalado:

$$Y_{f,d} = \frac{E_d}{P_{STC}} \tag{8.1}$$

donde P_{STC} es la potencia nominal fotovoltaica instalada en condiciones estándar (STC) de $1kW/m^2$ irradiancia solar y 25°C de temperatura de célula.

La energía úitl diaria de salida o energía diaria suministrada por un sistema, E_d se obtiene a partir de la expresión:

$$E_d = \int_d P_{AC}(t)d(t) \approx \sum_{j=1}^{n} P_{AC}^j \triangle t \tag{8.2}$$

donde n es el número de medidas a lo largo del día y P_{AC}^j es el valor registrado de potencia generada a la salida del inversor.

Como ha sido previamente propuesto, ver por ejemplo [ADMC11], la potencia de salida del inversor tiene una relación lineal con la irradiancia solar, si se desprecia el efecto de la temperatura. En este trabajo se propone incluir este efecto en el modelo que permite estimar la potencia de salida generada por el inversor, (P_{AC}^*), de acuerdo con la expresión 8.3.

$$P_{AC}^* = P_{STC}\frac{G_\beta}{1000}(1 + \gamma(T_{mod} - 25))GL \tag{8.3}$$

donde, G_β es la irradiancia global en la superficie de los módulos, β es la inclinación de los módulos, γ es el coeficiente de temperatura de P_m, T_{mod} es la temperatura de los módulos y GL es el coeficiente global de pérdidas del sistema. La Eq.8.3 se ha obtenido a partir de la expresión propuesta por [Ost86]. Esta expresión incluye tanto las pérdidas producidas por la temperatura como otro tipo de pérdidas (suciedad, pérdidas por distribución espectral, etc.)

Los resultados obtenidos con este modelo se utilizan para estimar la energía que debería suministrar el sistema fotovoltaico, E_d^*. La energía diaria estimada que debería producir una instalación, E_d^*, se calcula a partir de la Eq. 8.4.

$$E_d^* = \int_d P_{AC}^*(t)d(t) \approx \sum_{j=1}^{n} P_{AC}^{*(j)} \triangle t \tag{8.4}$$

La propuesta que se hace en este trabajo es evaluar el funcionamiento de una planta utilizando los valores de energía diaria estimada, E_{day}^*, con los de energía realmente producida, E_d.

De manera similar, para detectar problemas en el funcionamiento de una instalación, el valor de rendimiento diario, Eq.8.1, se compara con el valor de rendimiento diario estimado, $Y_{f,day}^*$, calculado según la Eq. 8.5.

$$Y_{f,day}^* = \frac{E_{day}^*}{P_{STC}} \tag{8.5}$$

Por otra parte, se han estimado también los siguientes parámetros:

- Energía diaria producida en alterna, $E_{D,CA}$. Se calcula a partir de la expresión

  $$E_{D,CA} = \int_t P_{ca} t dt$$

  donde t es el tiempo transcurrido durante un día específico y $P_{ca}(t)$ es la potencia en alterna obtenida del inversor en el tiempo t. En secciones, instalaciones y grupos se calcula como la suma de la $E_{D,CA}$ de todos los elementos contenidos en las distintas agrupaciones.

- Energía diaria en continua entregada al inversor por el campo de paneles, $E_{D,CC}$, calculada a partir de la expresión:

  $$E_{D,CC} = \int_t P_{cc} t dt$$

  similar a la propuesta para estimar $E_{D,CA}$.

- Energía diaria recibida por el campo de paneles, E_D, calculada a partir de la expresión:

  $$E_D = \frac{\int_t Rad(t)dt}{3600}$$

  donde $Rad(t)$ es la radiación recibida en el tiempo t.

- Rendimiento diario del inversor, μ_D, calculado como:

  $$\mu_D = \frac{E_{D,CA}}{E_{D,CC}} 100$$

- Productividad media diaria del inversor, $Yield$, calculada a partir de la expresión:

$$Yield = \frac{E_{D,CA}}{W_p}$$

donde W_p es la potencia pico del generador fotovoltaico conectado al inversor. W_p será una medida constante asociada al dispositivo que representa al generador haciendo uso de los mecanismos que proporciona el marco de trabajo indicadas en 5.6.

- Rendimiento genérico medio del inversor, PR, calculado como:

$$PR = \frac{1000 Yield}{100 E_D}$$

En el caso de las medidas globales asociadas a este tipo de instalaciones se propone la utilización de las siguientes medidas:

- Energía total en alterna, $E_{t,CA}$. Es la suma de todos los valores de $E_{D,CA}$ del sistema.
- Energía total en continua, $E_{t,CC}$. Es la suma de todos los valores de $E_{D,CC}$ del sistema.
- Rendimiento global del sistema, μ_g. Se calcula como la media aritmética de todos los valores de μ_D del sistema.
- Productividad global del sistema. $Yield_g$. Se calcula como la media aritmética de todos los valores de $Yield$ del sistema.

8.3. Modelo estadístico propuesto para evaluar el funcionamiento de un sistema de energía solar fotovoltaica

A partir de los valores estimados descritos en la sección anterior y las medidas realmente registradas, en este trabajo se propone que la comprobación del funcionamiento de un sistema energético se realice utilizando una metodología similar a la propuesta en [VAA+09] basada en estadística *descriptiva* e *inferencial*. El valor medio de los parámetros definidos en Eq. 8.1 and 8.2 se propone la estimación de la desviación estándar de estos parámetros para cada tipo de instalación (teniendo en cuenta tipos de módulos fotovoltaicos e inversor). Utilizando los resultados obtenidos, para evaluar el funcionamiento de cada planta se realiza un análisis estadístico entre los valores estimados de estos parámetros y los registrados. Se evalúa si existen diferencias significativas entre ambos valores utilizando el test de Jarque-Bera (suponiendo una distribución normal),

[JBA87]. Los valores se estiman a partir de los registros históricos de cada instalación, utilizando solo aquellos valores para los que la instalación ha funcionado correctamente. Así, para cada parámetro evaluado, cada nuevo par de valores registrado, X, y su estimación, X^*e n un instante i, la diferencia entre ambos se analiza utilizando el siguiente criterio (nivel de significación 5 %):

$$\begin{aligned} d_X^{(i)} &= X^{(i)} - X^{*(i)}; \\ & if\ d_X^{(i)} \notin [-1{,}96\hat{\sigma}, +1{,}96\hat{\sigma}]\ then\ mark\ (i), \end{aligned} \tag{8.6}$$

donde $\hat{\sigma}$ es la desviación estándar del parámetro X. Todos los valores marcados (i) corresponden a situaciones en las que ha habido algún problema en el funcionamiento de la instalación.

Para la potencia generada, $P_{AC}^{(i)}$, y su correspondiente estimación utilizando Eq.8.3, $P_{AC}^{*(i)}$, el criterio es el siguiente:

$$\begin{aligned} d_{P_{AC}}^{(i)} &= P_{AC}^{(i)} - P_{AC}^{*(i)}; \\ & if\ d_{P_{AC}}^{(i)} \notin [-1{,}96\hat{\sigma}, +1{,}96\hat{\sigma}]\ then\ mark\ (i), \end{aligned} \tag{8.7}$$

Para el rendimiento final diario, $Y_{f,d}$, la expresión utilizada para decidir si hay algún problema de funcionamiento en la planta se obtiene a partir de la Eq.8.6 particularizada para este parámetro:

$$\begin{aligned} d_{Y_{f,d}}^{(i)} &= Y_{f,day}^{(i)} - Y_{f,d}^{*(i)}; \\ & if\ d_{Y_{f,d}}^{(i)} \notin [-1{,}96\hat{\sigma}, +1{,}96\hat{\sigma}]\ then\ mark\ (i), \end{aligned} \tag{8.8}$$

La Fig. 8.2 muestra el diagrama de flujo propuesto para la evaluación diaria de una planta fotovoltaica.

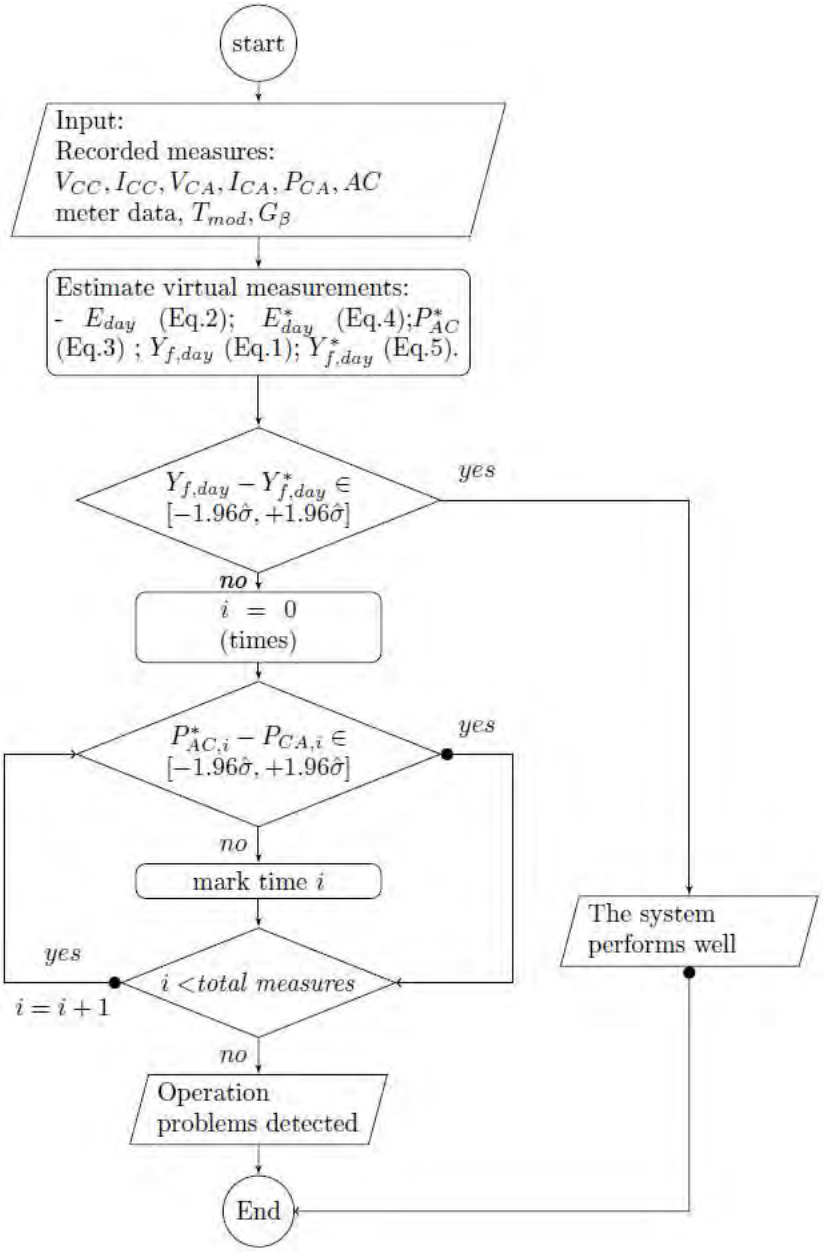

Figura 8.2: Diagrama de flujo para la evaluación de una planta de energía solar fotovoltaica.

8.4. Modelo para la predicción de la producción de energía de un sistema fotovoltaico

La predicción a corto plazo de variables continuas se ha hecho tradicionalmente a partir de la teoría de series temporales. En los últimos años, también se están empezando a utilizar distintas herramientas basadas en modelos de aprendizaje automático. En general, los métodos utilizados para series temporales continuas intentan encontrar modelos que sean capaces de reproducir tanto las características estadísticas como las secuenciales de las series originales y predecir a corto, medio y/o largo plazo el comportamiento de las variables analizadas.

Por una parte, los modelos estadísticos clásicos se basan en la utilización de los modelos estocásticos, que asumen que los datos tienen una estructura interna; esta estructura puede ser identificada utilizando las funciones de autocorrelación y autocorrelación parcial, [BJ76], [GH05], [BD02]. Utilizando estas funciones, se hace una selección previa de los modelos que se van a utilizar. Para estos modelos, el siguiente paso es estimar sus parámetros. En este estimación, se suele asumir que la dependencia entre los valores de la serie es constante a lo largo del tiempo. El hecho de tener que hacer una selección previa de modelos y la restricción de que los parámetros estimados sean constantes a lo largo del tiempo supone una limitación a la hora de utilizar estos modelos para algunas variables para las que se sabe que no tienen este comportamiento.

Por otra parte, también se están empezando a utilizar métodos y técnicas de aprendizaje automático para la predicción de series temporales, como por ejemplo en los trabajos de [GL03], [XDT04], [ZQ05], [WH08], [SC93], [SC94], [HCL98].

Para la predicción de variables que intervienen en el funcionamiento de sistemas energéticos, se pueden citar los trabajos de [PMS07], en el que se propone un modelo para estimar valores de energía solar a partir del parámetro cielo cubierto (medido por United States National Weather Service (NWS), [MLMdCMB05] en el que se propone la utilización de un tipo especial de autómata finito probabilista para la predicción de variables climáticas, [VAR08] en donde se proponen distintas aproximaciones para la estimación de valores de radiación solar, [GPC06] [GNAB09] en los que se propone la utilización de redes neuronales artificiales para la predicción de radiación solar; [CDCL11] que utilizan un mapa autoorganizado (self-organized map, SOM) para la clasificar el tipo de climatología local, aunque en este trabajo se utilizan las predicciones meteorológicas de irradiación solar, humedad relativa y temperatura del emplazamiento, suministradas por los servicios meteorológicos (online); y en [BMP13] se propone la combinación de modelos SARIMA (autoregresivos y medias móviles integrados, estacionales).

En lo que se refiere a la predicción de la energía producida por un sistema fotovoltaico, se pueden citar los siguientes trabajos: [LHHB09] en el que se presenta un modelo para predicción de producción de energía a partir de la predicción de radiación solar; [BMN09] en el que se propone un método on-line para hacer predicción a corto

plazo de la producción de plantas fotovoltaicas utilizando modelos autorregresivos y numéricos; [SRIS09] en el que se analiza la utilización de redes neuronales artificiales para predicción de la producción de sistemas fotovoltaicos; [FOT+11] en que se propone la utilización de un perceptrón multicapa y máquinas de vectores de soporte (support vector machine) para la predicción de la producción de una instalación fotovoltica en Japón; [LSH+11] se presenta y evalua la predicción de potencia de un sistema en la Universidad de Oldenburg proporcionando una predicción de hasta dos días con una resolución horaria; en [SCST12] se proponen varios modelos de predicción de la potencia de salida y la eficiencia de un sistema fotovoltaico pero en una base mensual y anual; en [FOT+12] se propone la utilización de regresiones de vectores de soporte y modelos numéricos también la predicción de una planta fotovoltaica instalada en Japón; en [ZMAP14a] y [ZMAP14b] se presentan los resultados obtenidos de analizar y evaluar modelos, construidos con métodos estadísticos y basados en la predicción de modelos numéricos, para la predicción de la producción de algunas plantas fotovoltaicas en Francia; en [DSFOO+14] se evalúan tres estrategias basadas en predicciones meteorológicas, generación fotovoltaica previa y regresión de máquinas de soporte vectorial, para obtener la predicción regional a un día de generación fotovoltaica; en [YHHP14] se proponen tres etapas, clasificación, entrenamiento y predicción utilizando SOM, redes LVQ (Learnig Vector Quantization) para modelizar los valores pasados de producción fotovoltaica, regresión con máquinas de soporte vectorial para entrenar las entradas/-salidas de temperatura, probabilidad y predicción; y los recientes trabajos de [BTM15] en el que se propone la predicción de la producción con una antelación de 6 horas y [WZM+15] en la que se propone un modelo para la predicción de la potencia fotovoltaica basada en el reconocimiento de patrones meteorológicos y la extracción de características de radiación solar y SVM.

Por otra parte, los modelos estadísticos pueden trabajar con valores continuos pero algunos modelos de aprendizaje automático solo pueden ser utilizado para datos que puedan ser representados de manera discreta o nominal. Esto implica que cuando se quieran utilizar estos modelos será necesario hacer previamente una discretización de las variables que sean continuas, tal y como ha sido señalado en [DK95], [LHTD02]. Para la discretización de valores continuos se han propuesto diferentes métodos, entre otros se pueden citar los trabajos de [DK95], [PT98], [MLRMBR00], [LHTD02] y [Bou04].

Respecto a los resultados obtenidos en la predicción del parámetro radiación solar, una de las conclusiones es que el error y precisión de estos modelos varía pero en todos los casos es significativo, como se concluye en [PMS07], [MLMdCMB05], [VAR08], [MP10], [Rei09], [KOV+11], [CCM+13]. El error, en términos de energía, es alrededor del 32 % en [MP10], varía entre el 30 y el 40 % en [Rei09] y varía entre el 23 y el 28 % en [CCM+13].

La predicción de la producción de energía en un sistema fotovoltaico o termosolar se hace a partir de los parámetros propios de la instalación conocidos, y de la predicción de los valores de energía que el sistema recibirá. Este parámetro es conocido como radiación solar, cuando se trabaja con intervalos de tiempo, o irradiancia, cuando se trabaja con valores instantáneos. Los valores de radiación solar se registran de manera

sistemática en algunas estaciones meteorológicas, en distintos intervalos temporales. En esta tesis se propone la utilización de estos registros históricos de cada planta, cuando estén disponibles, como datos de entrada para el modelo de predicción a corto plazo de la producción de energía de sistemas que utilizan este recurso como fuente energética. En el caso de que no se disponga de esta información, la predicción de la radiación que recibiría la planta se hará utilizando los resultados del modelo propuesto en [MLMPdC10].

La idea es desarrollar un modelo que sea capaz de *aprender* la información que es importante (significativa) para hacer estas predicciones. El modelo se construye en tres fases. En la primera fase se selecciona la variable independiente más significativa utilizando un análisis de regresión multivariante. Utilizando esta variable, las observaciones se dividen en grupos. En la segunda fase se seleccionan para cada grupo las variables más significativas y se estima el modelo para el grupo. En la tercera fase se hace la predicción de los valores del parámetro a corto plazo.

8.4.1. Modelo propuesto para la predicción a corto plazo

Se denomina *predicción* a la estimación de valores futuros de una variable (variable dependiente) en función del comportamiento pasado de la misma y de otras variables que puedan ser significativas para esta estimación (variables independientes). El modelo que se propone en esta tesis se construye en tres etapas.

En la primera etapa se utilizan técnicas estadísticas para seleccionar la información más significativa para predecir la variable dependiente. La información más significativa suele provenir de los valores pasados de la variable, pero en el modelo desarrollado es posible incluir otras fuentes de información, como puede ser el conocimiento de expertos. Esta información de otras fuentes puede ser numérica (valores continuos) o nominal (valores discretos); en el primer caso, las variables pueden ser utilizadas directamente en el análisis de regresión, mientras que en el segundo caso la información se codifica utilizando variables ficticias o *dummy*. Ejemplos de este tipo de información puede ser la estación del año, tipo de día, etc. Entre todas las variables independientes la que es globalmente más significativa juega un papel importante ya que es utilizada para clasificar las observaciones en grupos, en función de los valores de esa variable. Después, para cada grupo se vuelven a determinar cuáles son las variables más significativas de ese grupo; estas variables son las que se utilizarán en la fase siguiente.

En la segunda fase se estima el modelo de predicción de cada grupo. En los trabajos previos de predicción a corto plazo, el mismo modelo de predicción se utiliza sea cual sea la situación actual de la variable para la que se hace esta predicción y, normalmente, solo se utilizan los valores previos de esa variable, tanto cuando se utilizan modelos estadísticos como cuando se utilizan modelos de aprendizaje automático. Sin embargo, como se comprueba con los resultados obtenidos en este capítulo, la posibilidad de utilizar modelos distintos dependiendo de la situación actual hace que se mejoren sig-

nificativamente las predicciones. Es por esta razón por lo que se propone una primera separación de las observaciones en grupos.

En la tercera fase, se realiza la predicción a corto plazo del valor de la variable usando el valor actual de la variable más significativa determinada en la primera fase, los valores de las variables significativas del grupo al que pertenece el valor actual y el modelo seleccionado para ese grupo.

8.4.2. Descripción del procedimiento

Se asume que se dipone de las observaciones $\{(Y_t, X_t)\}_{t=1}^{T}$, donde Y_t es una variable univariante aleatoria, X_t es un vector aleatorio q-dimensional y t representa tiempo. Se está interesado en construir un modelo para predecir la variable Y_{T+1} cuando son conocidas las variables X_{T+1}.

En la primera fase, se realiza un análisis de regresión basado en las observaciones $\{(Y_t, X_t)\}_{t=1}^{T}$ para determinar la información que es significativa para la predicción.

- Primer paso: Estimar por mínimos cuadrados el modelo de regresión lineal utilizando la expresión 8.9:

$$Y_t = \beta_0 + \beta_1 X_{1t} + ... + \beta_q X_{qt} + \text{Error}, \tag{8.9}$$

  usando para ello las T observaciones. En lo sucesivo, y tras ese primer análisis, se supondrá que las componentes del vector aleatorio X_t están ordenados de forma que la primera componente, i.e. X_{1t}, es la que tiene el mayor t-estadístico, en valor absoluto, en esa regresión.

- Segundo paso: Dados los números reales $c_1 < ... < c_{G-1}$, separar las observaciones de la muestra en G grupos de la siguiente forma: la observación tth estará en el grupo 1 si $X_{1t} < c_1$, estará en el grupo g si $X_{1t} \in [c_{g-1}, c_g)$ para $g \in \{2, ..., G-1\}$, y en el grupo G si $X_{1t} \geq c_{G-1}$. En adelante, se utilizará T_g para denotar el número de observaciones del grupo g; obviamente $\sum_{g=1}^{G} T_g = T$. Una vez que la muestra se ha dividido en grupos, se asume que las observaciones están ordenadas según el grupo al que pertenecen, es decir, las primeras T_1 observaciones son las del Grupo 1, y así sucesivamente. Así, dado $g \in \{1, ..., G\}$, las observaciones en el grupo g son aquellas observaciones t tales que $t \in \{T_*^{(g)}, ..., T_{**}^{(g)}\}$, donde se denota $T_*^{(g)} \equiv 1 + \sum_{i=0}^{g-1} T_i$, $T_{**}^{(g)} \equiv \sum_{i=1}^{g} T_i$ and $T_0 \equiv 0$.

- Tercer paso: Para cada $g \in \{1, ..., G\}$, estimar por mínimos cuadros el modelo de regresión lineal (8.9) utilizando solo las T_g observaciones del grupo g, y para $j \in \{1, ..., q\}$, definir $D_j^{(g)} = 1$ si el valor del t-estadístico de X_{jt} en esa regresión es, en valor absoluto, mayor que M_g, donde $M_1, ..., M_G$ son valores fijados previamente.

En este tercer paso de la primera etapa es necesario decidir qué variables explicativas se van a considerar como significativas en los modelos de ajuste para el gth grupo, este umbral de significancia se determina por el valor M_g. Así, $D_j^{(g)} = 1$ significa que la componente jth de X_t se considera como una variable significativa para las observaciones del gth grupo.

El objetivo de la segunda fase es la estimación del autómata finito probabilista de cada grupo utilizando las variables significativas de ese grupo. En cada grupo, el modelo seleccionado es aquel con menor error cuadrático relativo de predicción en la muestra (MRSPE, mean relative square prediction error); para el grupo g y el modelo m (donde $g \in \{1, ..., G\}$ y $m \in \{1, 2, 3\}$) esta cantidad se define según la expresión 8.10:

$$\text{MRSPE}(g, m) = \frac{1}{T_g} \sum_{t=T_*^{(g)}}^{T_{**}^{(g)}} \frac{(\widehat{Y}_{t,m} - Y_t)^2}{Y_t^2}, \tag{8.10}$$

donde $\widehat{Y}_{t,m}$ es la predicción de Y_t obtenida a partir del modelo m. Para las observaciones en el grupo g, los valores de predicción $\widehat{Y}_{t,m}$ se obtienen aplicando el procedimiento que se describe a continuación para las observaciones del grupo$g$.

Este modelo se basa en la construcción de un tipo especial de autómata finito. Como algunas de las variables utilizadas (la mayoría) son continuas y este modelo solo puede ser utilizado para variables discretas, el primero paso para poder utilizar este modelo es discretizar las variables continuas. Para ello, cada componente de X_t es discretizada utilizando un método de discretización estático; en concreto, $j \in \{1, ..., q\}$ y dados los números reales $d_1^{(g,j)} < ... < d_{I(g,j)}^{(g,j)}$, considerar la variable aleatoria

$$X_{jt}^{(g)*} := \begin{cases} d_1^{(g,j)} & \text{if } X_{jt} < d_1^{(g,j)} \\ (d_1^{(g,j)} + d_2^{(g,j)})/2 & \text{if } X_{jt} \in [d_1^{(g,j)}, d_2^{(g,j)}) \\ \dots & \dots \\ (d_{I(g,j)-1}^{(g,j)} + d_{I(g,j)}^{(g,j)})/2 & \text{if } X_{jt} \in [d_{I(g,j)-1}^{(g,j)}, d_{I(g,j)}^{(g,j)}) \\ d_{I(g,j)} & \text{if } X_{jt} \geq d_{I(g,j)}^{(g,j)} \end{cases} \tag{8.11}$$

Cuando se ha hecho el proceso de discretización, utilizando un autómata finito probabilista (PFA) para el grupo gth de$g$, para cada observación t encontrar el subíndice s in $\{T_*^{(g)}, ..., T_{**}^{(g)}\}$ que cumple $X_{js}^{(g)*} D_j^{(g)} = X_{jt}^{(g)*} D_j^{(g)}$ para todos los $j \in \{1, .., q\}$, utilizando el procedimiento descrito en [MLMdCMB05]; el conjunto de subíndices que satisface esta condición se denotará por $\mathcal{S}_t$. Así, la predicción para Y_t, denotada como $\widehat{Y}_{t,3}$, es la mediana de $\{Y_s^{(g)*}\}_{s \in \mathcal{S}_t}$.

En el caso de que se optara por dar un intervalo de predicción, entonces el intervalo de predicción para Y_t se define como el intervalo $[d_k^{(g,0)}, d_{k+1}^{(g,0)})$ que contiene $\widehat{Y}_{t,3}$, donde

$d_1^{(g,0)} < ... < d_{I(g,0)}^{(g,0)}$ son los valores que se han utilizado para discretizar la variable dependiente. Hay que señalar que también podría haberse usado la moda o el valor medio $\{Y_s^{(g)*}\}_{s\in\mathcal{S}_t}$ en lugar de la mediana, dependiento del tipo de error que se quiera minimizar en cada caso.

También hay que tener en cuenta que el autómata finito probabilista que se propone utilizar es un tipo especial de autómata que permite que algunos valores pasados de la serie temporal de la variable dependiente sean aprendidos por el autómata y otros sean olvidados, en función de lo significatividad de cada uno y no de la distancia al valor actual; así, aunque este autómata está basado en el propuesto en [RST94], la diferencia apuntada es esencial para el modelo que se propone; otra diferencia importante es que en el modelo propuesto se permite incorporar información de diferente tipo, no solo la propia de la serie temporal de la variable dependiente. Como punto débil de este tipo de modelos, la necesidad ya apunta tener que discretizar las variables que sean continuas. Es por ello, que en esta tesis se propone un método para seleccionar los intervalos de discretización de manera dinámica.

En la tercera fase, partiendo de la observación X_{T+1}, el procedimiento para predecir Y_{T+1} es el mismo que el que se ha descrito en el paso segundo de la segunda fase para predecir Y_t.

8.4.3. Selección de los datos de entrada al modelo de predicción propuesto

En la práctica, el modelo propuesto requiere que se fijen de antemano varios datos de entrada. Se enumeran estos datos y se dan algunas indicaciones sobre cómo pueden seleccionarse estos datos:

1. Los valores $c_1, ..., c_{G-1}$ que se utilizan para determinar los G grupos en los que se dividirán las observaciones de la muestra, teniendo en cuenta cuál es la variable independiente más significativa.

2. Los valores $M_1, ..., M_G$ que se utilizan para determinar cuándo una variable se considera significativa en cada una de las G regresiones. En principio, todos estos valores pueden fijarse a 1.96, lo que significa que se considera un nivel de significancia 0.05 para determinar si una variable es o no significativa.

3. En el autómata finito probabilista, los valores $d_1^{(g,j)}, ..., d_{I(g,j)}^{(g,j)}$ que se utilizan para la discretización de la variable explicativa jth, continua, en el grupo gth. El problema de cómo elegir los rangos que pueden utilizarse para discretizar una variable continua ha sido ampliamente tratado en contextos similares al que se presenta aquí, por ejemplo en [LHTD02], [DK95] and [PT98]), ya que muchos algoritmos utilizados en aprendizaje automático supervisado solo pueden utilizarse para datos discretos. En esta tesis se propone un método de discretización que

permite explorar varios discretizados posibles y seleccionar el mejor o aquel con el que se obtenga un error inferior al máximo error admisible ε en la estimación del modelo de autómata finito probabilista para el problema que se esté analizando. El objetivo del método que se propone es forzar a la prueba de distintas discretizaciones, seleccionando en cada una diferentes umbrales de discretización.

4. Los parámetros necesarios para la construcción del autómata finito probabilista, según el procedimiento de estimación propuesto en [RST94], a saber, el umbral de probabilidad y el orden (memoria o longitud) del autómata, dependiendo del número de observaciones que tiene cada nodo que se va construyendo en el autómata. Esta dependencia evitará el problema del sobreajuste. Los criterios que se han utilizado en esta tesis son los que se proponen en el trabajo de [MLMdCMB05].

En cualquier caso, la experiencia y el conocimiento acerca del comportamiento de la variable dependiente que puedan tener los expertos es muy importante a la hora de seleccionar estos datos de entrada. Es importante destacar, que una vez decididos estos datos de entrada, el procedimiento de construcción del autómata finito probabilista para una muestra funciona de forma automática.

8.4.4. Discretización de datos continuos

En este apartado se describe el procedimiento para seleccionar los intervalos de discretización que se utilizarán para discretizar las distintas variables independientes continuas que se incluyan en el grupo g. En los símbolos utilizados para representar las distintas variables que intervienen en el procedimiento se ha eliminado la referencia al grupo al que pertenecen las observaciones, g.

Como se utilizará un discretizado estático, según la expresión 8.11, la forma en que se construye cada uno de estos intervalos utilizados para discretizar la variable continua j del grupo g es la siguiente:

Sean $nvar$ el número de variables significativas continuas del grupo g y $ninterv_m^j$ el número de intervalos en que se va a dividir el rango de la variable j en la iteración m. Para la iteración m, que cumple $1 \leq m \leq niter^j$, siendo $niter^j$ el número de discretizaciones posibles (iteraciones) que se van a probar para la variable significactiva j del grupo g

---

`Valores de los intervalos de discretización` m:

$$ancho = \frac{(max^j - min^j)}{ninverv_m^j}$$

$d_i^j = ancho * i$ where $i = 1, ..ninterv_m^j$

Utilizando estos valores en las disintas discretizaciones de cada variable significativa, se describe a continuación el procedimiento para obtener el autómata finito probabilista para el grupo g:

`for` $j = 1, ..., nvar$ $iter^j = 1$ `//primera iteración para todas las var`

$\epsilon = 0{,}10$ `máximo error admisible`

`do` {

`Discretizar las variables continuas, 8.11, utilizando los valores que se han definido para la iteración` $iter^j$ `de cada variable`

`Construir el autómata finito probabilista`

`Estimar el error cuadrático medio relativo, MRSPE`

`if(MRSPE>`ϵ `)` $iter^{nvar} = iter^{nvar} + 1$;

`for` $(h = nvar; h > 1; --h)$

`if` $(iter^h > niter^h)$ {

$iter^h = 1$;

$iter^{h-1} = iter^{h-1} + 1$;

}

} `while` $(MRSPE > \epsilon)$ `AND` $(iter^1 1 < niter^1 + 1)$

De esta forma, el proceso de construcción del autómata puede finalizar antes de que se hayan probado todas las discretizaciones posibles, si en alguna de las iteraciones se cumple que el error cuadrático medio relativo en la muestra es menor que el definido para el problema para el que se esté usando este modelo ϵ.

8.4.5. Validación del modelo propuesto para la predicción a corto plazo

El modelo propuesto puede ser utilizado para la predicción de la producción de energía en sistemas cuya recurso o fuente energética sea variable y difícil de estimar mediante modelos físicos o métodos analíticos clásicos. Un ejemplo de estos sistemas son los sistemas de energía solar fotovoltaica. En estos sistemas la fuente energética es la radiación solar, que presenta una componente no determinista que hace que sea imposible conocer con antelación la cantidad exacta de radiación que recibirá un sistema y, por tanto, la energía que producirá. Una descripción detallada de este tipo de sistemas energéticos puede encontrarse en [LH03].

Para la validación del modelo propuesto se han utilizado datos de radiación solar, temperatura y producción de energía de tres instalaciones fotovoltaicas distintas.

La energía producida por un sistema fotovoltaico se estima a partir de la expresión 8.12.

$$P_{AC} = \mu_{inv} * P_m^{STC} * \frac{G_\beta}{1000} * (1 + \gamma Ç(T_{mod} - 25)) \tag{8.12}$$

donde μ_{inv} es la eficiencia del inversor, P_m^{STC} es la potencia del generador fotovoltaico en condiciones estándar de radiación y temperatura $(1000 W m^{-2}, 25^oC)$, G_β es la radiación global incidente en la superficie de los módulos en ($W m^{-2}$, β es la inclinación de los módulos, γ es el coeficiente de temperatura y T_{mod} es la temperatura de los módulos. En el caso de los módulos de silicio monocristalino, que son el tipo de módulos que tienen las instalaciones de las que se han obtenido los datos que se utilizan en la validación, el valor del coeficiente de temperatura es $0{,}48\,\%^oC$.

El modelo propuesto se ha utilizado para la predicción de los valores de radiación solar que recibirá una instalación y, con estos valores y utilizando la expresión 8.12 se ha estimado la producción de energía de una planta a corto plazo (día siguiente); en trabajos previos también se ha apuntado ya la fuerte dependencia de la producción de este tipo de sistemas energéticos con los valores de radiación solar y un parámetro derivado de esta, el índice de transparencia atmosférico, [KU05], [NTI+10].

Los valores de radiación solar presentan una tendencia estacional debida, fundamentalmente, a los cambios en la posición relativa sol-tierra a lo largo del año y para cada día. Una de las formas más utilizadas para eliminar esta tendencia estacional es utilizar un parámetro que se obtiene a partir de los valores de radiación solar, dividiendo estos valores por lo que se conoce como radiación solar extraterrestre. Los valores de radiación solar extraterrestre pueden estimarse con una aproximación muy aceptable a partir de la expresión que se propone en [Iqb84]. Usando estos valores y los de radiación solar en la superficie de la tierra se calculan los valores de índice de transparencia atmosférico, definido según la expresión 8.13.

Tabla 8.1: Descripción de las características de las instalaciones

Loc	Latitud/Longitud	Potencia pico (kW)	Inclinación	Periodo
1	43,30/ − 1,95	14,08 kW	20	01/10/2009-10/12/2010
2	43,18/3,00	13,86 kW	20	01/10/2009-10/12/2010
3	43,37/ − 1,85	20,16 kW	30	01/11/2009-10/12/2010
4	43,37/ − 1,85	20,16 kW	30	01/11/2009-10/12/2010

$$k_t = \frac{G_t}{G_{0,t}} \tag{8.13}$$

donde t representa tiempo, G_t son los valores registrados de radiación solar para el intervalo temporal t y $G_{0,t}$ son los valores de radiación solar extraterrestre para ese intervalo temporal. Normalmente se suele trabajar con valores horarios y/o diarios de estos parámetros.

Datos utilizados para la validación

Los datos que se han utilizado para la validación del modelo son las medidas registradas en cuatro plantas fotovoltaicas instaladas en diferentes emplazamientos. De todas las medidas registradas se han utilizado los siguientes valores de los siguientes parámetros:

- Energía generada a la salida del inversor
- Radiación recibida en la superficie de los módulos
- Temperatura de los módulos

Además, también se ha utilizado la estación del año en la cuál se ha registrado cada medida. Esta información ha sido incluida utilizando variables ficticias o dummy. En la tabla 8.1 se resumen las características de cada una de las instalaciones.

Las variables independientes que se han utilizado son: índices de transparencia atmosféricos obtenidos a partir de los valores de radiación global registrados cada 15 minutos utilizando la expresión 8.13, los valores de temperatura de los módulos y la estación del año en la que se ha registrado cada variable.

8.4.6. Resultados de la validación del modelo propuesto

En la primera fase se ha estimado el modelo de regresión lineal de la expresión 8.9, que corresponde a la expresión 8.14 para las variables seleccionadas.

Tabla 8.2: Intervalos utilizados y variables significativas para cada intervalo

Intervalos	Variables significativas
$[0{,}0 - 0{,}2[$	$K_{t,d-2}, K_{t,d-3}, S_{3,t,d}$
$[0{,}2 - 0{,}4[$	$K_{t,d-2}, K_{t,d-1}, S_{3,t,d}$
$[0{,}4 - 0{,}6[$	$K_{t,d-2}, S_{3,t,d}$
$[0{,}6 - 0{,}8[$	$K_{t,d-2}, S_{1,t,d}$
$[0{,}8 - 1{,}0]$	$K_{t,d-1}, K_{t,d-2}, S_{1,t,d}$

$$k_{t,d} = \beta_0 + \beta_1 k_{t,d-1} + \beta_2 k_{t,d-2} + \beta_3 kt, d-3 + \beta_4 S_{1,t,d} + \beta_5 S_{2,t,d} + \beta_6 S_{3,t,d} + T_{t,d} + Error \tag{8.14}$$

donde k representa al índice de transparencia estimado a partir de la expresión 8.13, t es tiempo, representa los valores registrados cada 15 minutos, d es día, por lo que los subíndices del índice de transparencia significan son relativos siempre al instante de tiempo de la variable dependiente. Así, por ejemplo, $t, d-1$ significa el valor del índice de transparencia para el mismo intervalo de tiempo (hora-minuto) registrado el día anterior. S_i son las variables dummy utilizadas para codificar la estación del año; solo se han utilizado tres variables para evitar problemas de multicolinearidad. Por tanto se han usado la constante β_0 y 7 variables independientes.

En la regresión de la expresión 8.14, de todas las variables independientes utilizadas la que ha resultado ser más significativa para predecir el valor siguiente del índice de transparencia ha sido el índice de transparencia registrado en el mismo intervalo de tiempo del día anterior, es decir, $k_{t,d-1}$. Utilizando esta variable la muestra se ha separado en 5 grupos distintos dependiendo del valor de esta variable. Los valores utilizados para construir estos grupos se han estimado a partir de la expresión 8.15.

$$c_i := \begin{cases} i/5 & \text{for } i = 1, .., 4 \\ 1{,}0 & \text{for } i = 5 \end{cases} \tag{8.15}$$

Para cada uno de los grupos se ha vuelto a estimar la regresión 8.14 para determinar las variables significativas de cada grupo. El resultado se muestra en la tabla 8.2.

En la segunda fase, se ha construido un autómata finito probabilista para cada grupo utilizando las variables significativas del grupo. A partir de esos autómatas se ha obtenido la predicción de los valores del índice de transparencia y a partir de estos, los valores de irradiancia. Con estos valores y utilizando la expresión 8.12 se ha estimado la energía producida por cada una de las instalaciones fotovoltaicas. El error estimado se calculado a partir de la expresión 8.10. Los resultados que se han obtenido se muestran en la tabla 8.3.

Tabla 8.3: Error medio de predicción del modelo propuesto

L1	L2	L3	L4
0.18	0.14	0.17	0.16

8.5. Conclusiones

En este capítulo se han propuesto modelos para la gestión de sistemas de energía solar fotovoltaica. Por una parte, se ha propuesto un modelo estadístico que permite hacer una evaluación automática del funcionamiento de una instalación, a partir del análisis de varios parámetros que sirven para determinar este. En el modelo propuesto se hace uso de conceptos de estadística inferencial.

Por otra parte, se ha propuesto un modelo que permite predecir la energía que va a producir un sistema fotovoltaico a partir de la predicción de la radiación que recibirá. Este modelo solo utiliza información de la que se registra en una planta fotovoltaica, entre otra, los valores de radiación recibida los días previos. El modelo se construye a partir de tres estados. Para la construcción del mismo se ha utilizado un tipo especial de autómatas finitos probabilistas. El error del modelo propuesto es menor del 20 % frente a modelos tradicionales cuyo error es de un 25 % aproximadamente, lo que supone una mejora significativa.

"Hoy en día la mayoría del software existe no para resolver un problema, sino para actuar de interfaz con otro software"

– I. O. Angell

9

Extensión del marco de trabajo para el desarrollo de sistemas de gestión de instalaciones de energía solar fotovoltaica

9.1. Introducción

En la primera parte de esta memoria se ha presentado el marco de trabajo genérico propuesto para modelizar y desarrollar aplicaciones para la gestión y monitorización de instalaciones energéticas. La finalidad última de un marco de trabajo es poder adaptarse y utilizarse para su utilización en diferentes problemas del mismo dominio aplicando las técnicas de extensibilidad y configuración.

En este capítulo se propone la utilización de este marco de trabajo genérico para crear soluciones para la gestión de instalaciones energéticas orientadas a la energía solar fotovoltaica gracias a la potencia del mismo. Además, se propone y muestra cómo incluir los modelos de evaluación y predicción descritos en el capítulo 8 en la extensión del marco de trabajo.

Según lo descrito en los primeros capítulos de esta memoria, se dispone de mecanismos para modelizar instalaciones genéricas y extenderlas al dominio de la fotovoltaica (véase capítulo 5), desarrollar mecanismos para la comunicación con dispositivos de manera genérica (véase capítulo 4), herramientas y procedimientos para almacenar la información de las instalaciones que se van a monitorizar y gestionar (véase capítulo 6), algoritmos para sincronizar y mantener la información de dichas instalaciones de ma-

nera coherente (véase 6.6) así como una capa de soporte para personalizar y modificar el marco de trabajo para adaptarlo a cada necesidad (véase capítulo 7).

En este capítulo se muestra cómo el marco de trabajo permite incluir el dominio del problema de manera modular y así describir de manera sencilla las particularidades de la gestión y monitorización de instalaciones de energía solar fotovoltaica.

9.2. Gestión de un sistema de plantas de energía solar fotovoltaica

La gestión de instalaciones de energía solar fotovoltaica de pequeño y medio tamaño se ha hecho tradicionalmente en las propias plantas, de manera manual por personal más o menos especializado y utilizando los datos de estas instalaciones obtenidos a partir de programas propios desarrollados por los fabricantes de los dispositivos que se integran en la instalación. Como ya se ha apuntado en esta tesis, esto presenta numerosos inconvenientes.

Uno de los más importantes es el hecho de que cuando los dispositivos que hay en una instalación pertenecen a distintos fabricantes, cada fabricante suministra su propio software de gestión, y el usuario final debe utilizar varios programas para la gestión del sistema. Además, muchas veces este software solo permite automatizar la tarea de recuperación de la información (monitorización), y no incluye herramientas que automaticen la evaluación del funcionamiento y la predicción de producción de estas plantas. Desde el punto de vista del ingeniero o especialista que va a gestionar este tipo de sistemas dispersos y que tienen subsistemas con dispositivos de distintos fabricantes, el marco de trabajo que se propone en esta tesis, permitirá desarrollar programas de gestión que integren todos estos distintos dispositivos en un único programa, así como integrar varias instalaciones y permitirá acceder a la información desde distintos canales, como se muestra en la figura 9.1.

El marco de trabajo propuesto proporciona un repertorio de clases, interfaces y la interrelación entre ellos, que pueden ser utilizados para la generación de distintos programas relacionado con la gestión de plantas de energía solar fotovoltaica.

El proceso para desarrollar estos sistemas pasa por los siguientes pasos:

1. Modelizar todos los elementos de cada planta de energía solar fotovoltaica. A partir de esta modelización se genera una representación persistente para el conjunto de plantas para las que se va a generar el programa. El modelizado puede hacerse utilizando las clases e interfaces que proporciona el marco de trabajo o utilizando herramientas automáticas creadas también a partir de este marco de trabajo. Los elementos que normalmente registran información en una planta de energía solar fotovoltaica son los inversores -que almacenan la información del generador

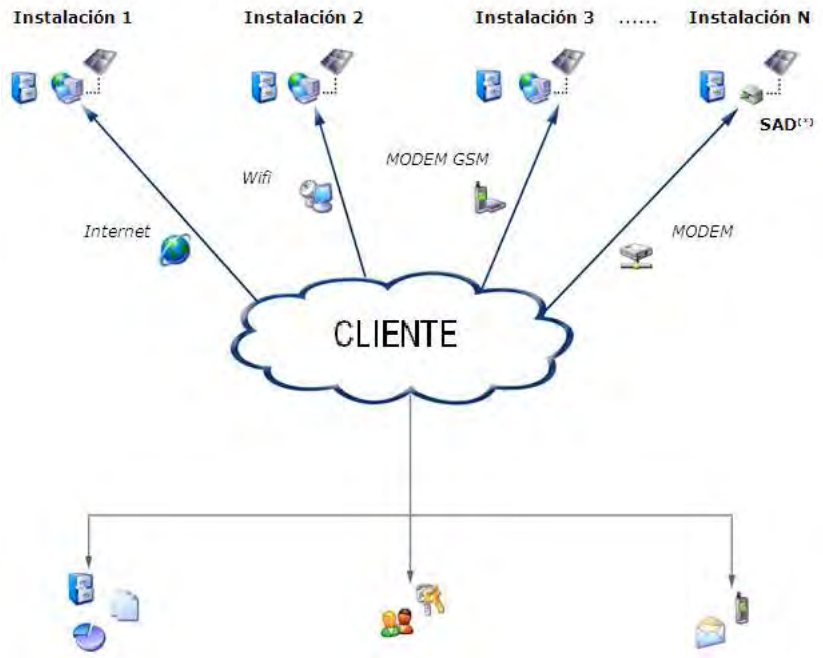

Figura 9.1: Unificación de protocolos y dispositivos en un único sistema de gestión y publicación multicanal de la información

fotovoltaico-, los sensores de radiación (células calibradas y/o piranómetros), los sensores de temperatura (tanto para medir temperatura ambiente como temperatura de los módulos) y contadores de energía. Además, algunas plantas pueden tener sus propios sistemas de adquisición de datos (datalogger, PCs, etc.), por lo que también será necesario modelizar esos dispositivos.

2. Seleccionar los dispositivos para parametrizar y asociar sus características físicas con los canales y medios de comunicación apropiados. Si fuera necesario, se describen o añaden nuevos atributos, medidas (canales) y las medidas virtuales. Finalmente, los diferentes elementos se unen y asocian con los componentes encargados de conectar de manera real responsables de las comunicaciones para cada medida.

3. Definir los componentes que tendrá el programa y sus funcionalidades: a partir del modelizado de las plantas, se especifican qué componentes tendrá la aplicación de gestión de las mismas. Entre otros, se podrán incluir un sistema de planificación de vigilancia de instalaciones, visualización de los datos, ejecución de análisis y filtros de los datos descargados, publicación web y avisos de eventos y alarmas. Para todas estas cuestiones el marco de trabajo proporciona elementos para su sencilla construcción además de las herramientas que se ha desarrollado para estas cuestiones.

4. Construir la aplicación visual haciendo uso de las clases e interfaces proporcionadas por el marco de trabajo.

A partir de la arquitectura propuesta en el marco de trabajo se puede generar un

sistema de gestión de plantas fotovoltaicas que interactúe con las plantas y en el que se incluyan las reglas de negocio y modelos de esas plantas, tal y como se muestra en la figura 9.2. En esta figura puede verse un sistema completo de capas que integra la arquitectura y las herramientas que se pueden incorporar a los sistemas de gestión que se desarrollen a partir de esta propuesta.

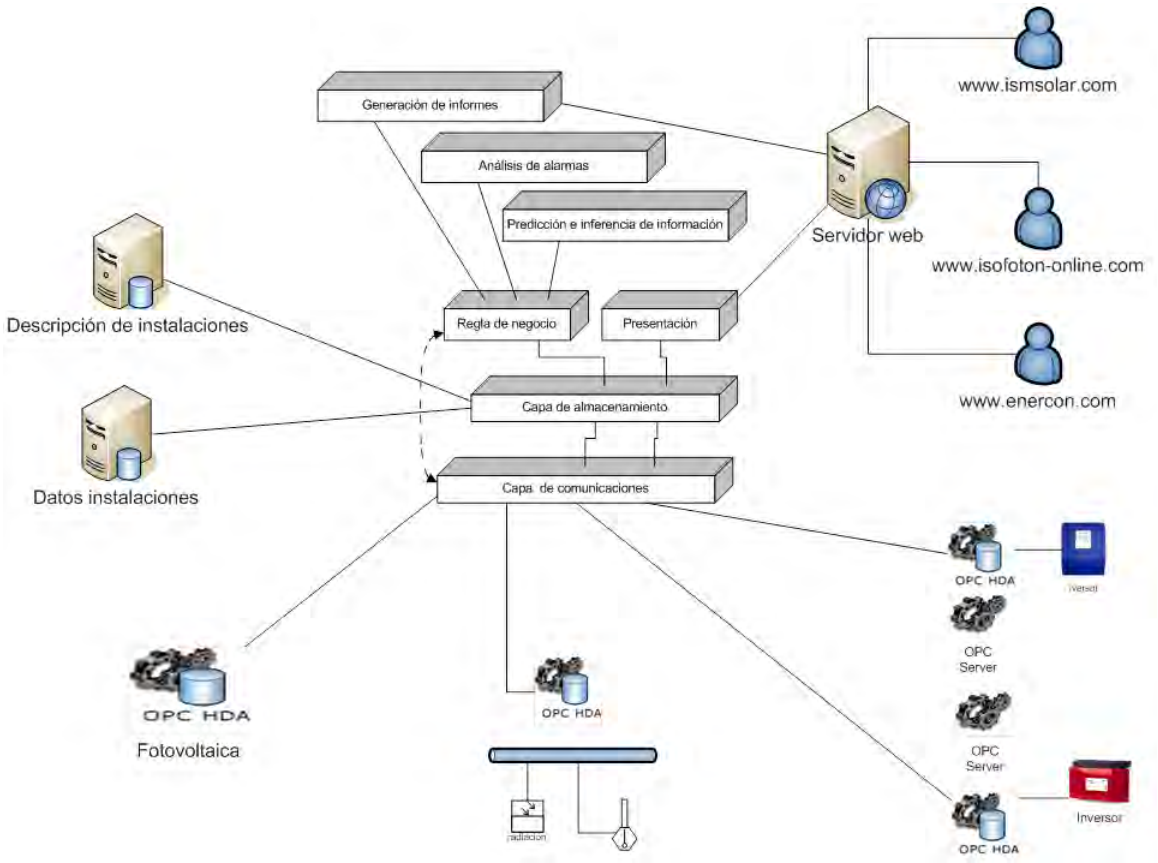

Figura 9.2: Estructura lógica de la arquitectura software propuesta para sistemas energéticos

Desde la capa inferior, que se encarga de la gestión de las comunicaciones con los dispositivos, hasta la capas más altas para la publicación de datos en web pasando por el modelizado de las plantas y sus dispositivos así como la gestión de los usuarios, es posible disponer de un sistema integral de gestión de instalaciones energéticas.

Hasta el momento se han propuesto y diseñado mecanismos y herramientas para poder disponer de un sistema integral de gestión de sistemas energéticos pero para una implantación real es necesario disponer de una infraestructura apropiada que acompaña a todo despliegue de software y, no formando parte del mismo, es necesario para su funcionamiento. En la figura 9.3 se hace una descripción de los elementos necesarios para poder desplegar un sistema de este tipo.

La gestión de muchas instalaciones hace necesaria disponer de varios servidores para la publicación web, un servidor para la base de datos y otro de respaldo así como poder balancear la descarga de datos para permitir la gestión de multitud de instalaciones al

mismo tiempo.

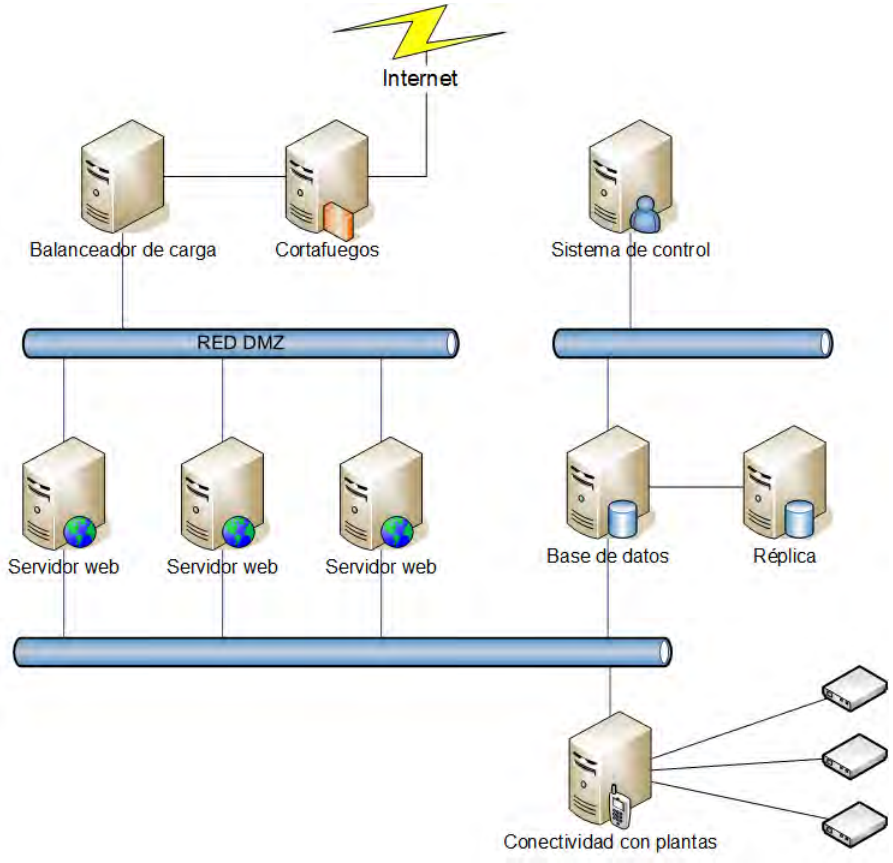

Figura 9.3: Infraestructura de comunicaciones para la gestión de varias instalaciones

9.3. Inclusión del dominio fotovoltaico

El marco de trabajo proporcionado se ha construido de manera genérica y utilizando los mecanismos apropiados de extensión del mismo la propuesta es incorporarle la particularidad de gestionar instalaciones energéticas en el ámbito de la energía solar fotovoltaica.

9.3.1. Incorporación de mecanismos de gestión

Una vez están definidas las medidas necesarias para disponer de la información de funcionamiento de una planta fotovoltaica se incorpora esta información en el marco de trabajo utilizando los mecanismos que ofrece. En este caso la solución que se propone, de manera genérica , es la de modelizar el dominio como si de un nuevo dispositivo se tratase y hacer que dicho dispositivo virtual efectúe la valoración de dichos resultados

energéticos basándose en la información del modelizado de la planta y de la información del resto de dispositivos. Esta solución permite incluir parámetros y fórmulas de cualquier dominio energético.

El marco de trabajo proporciona el flujo de funcionamiento para la gestión de cualquier sistema energético, conectando con todos los dispositivos de una planta y descargando los datos de los mismos (Fig. 6.6). El tratar el dominio energético, para el que se propone la extensión del marco de trabajo, como un dispositivo haciendo uso de la tecnología OPC-HDA, permite añadir de manera modular una gestión de la información desde el marco y hacia el marco al gestionar la información de los canales, procesarlos y volver a almacenarlos en el mismo sistema, figura 9.4.

Así, mediante la aplicación del flujo de descarga de datos de los dispositivos, se dispondrá de un análisis preliminar del estado completo de la planta haciendo muy eficiente la gestión de la instalación.

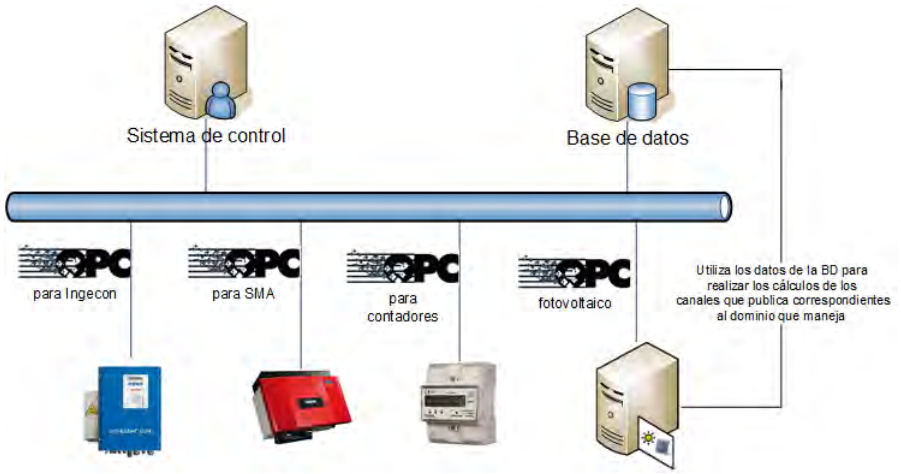

Figura 9.4: Inclusión del dominio fv basado en dispositivo virtual

Como se puede observar en la figura 9.4, el sistema de gestión y cálculo de los valores de funcionamiento de una planta fotovoltaica se ha incorporado como un dispositivo más por lo que cuando se descargan los datos de dicho dispositivo, lo que realmente se está haciendo es calcular los valores de dichos canales basándose en la información disponible en la base de datos del resto de dispositivos y almacenándose de nuevo en el sistema para su posterior análisis o publicación.

El servidor OPC encargado de gestionar la información en el dominio de los sistemas fotovoltaicos se ha realizado utilizando las clases e interfaces proporcionadas por el marco de trabajo al igual que los que se han desarrollado para los diferentes dispositivos, como puede verse en la figura 9.5.

En esta figura puede verse cómo se ha creado un servidor OPC-HDA *TISMH-DADBServer* que contiene la lógica para la conexión a la misma base de datos que está usando el marco de trabajo y por tanto, el lugar donde se almacena la información de los canales de todos los dispositivos de una planta. Con esta conexión se permite

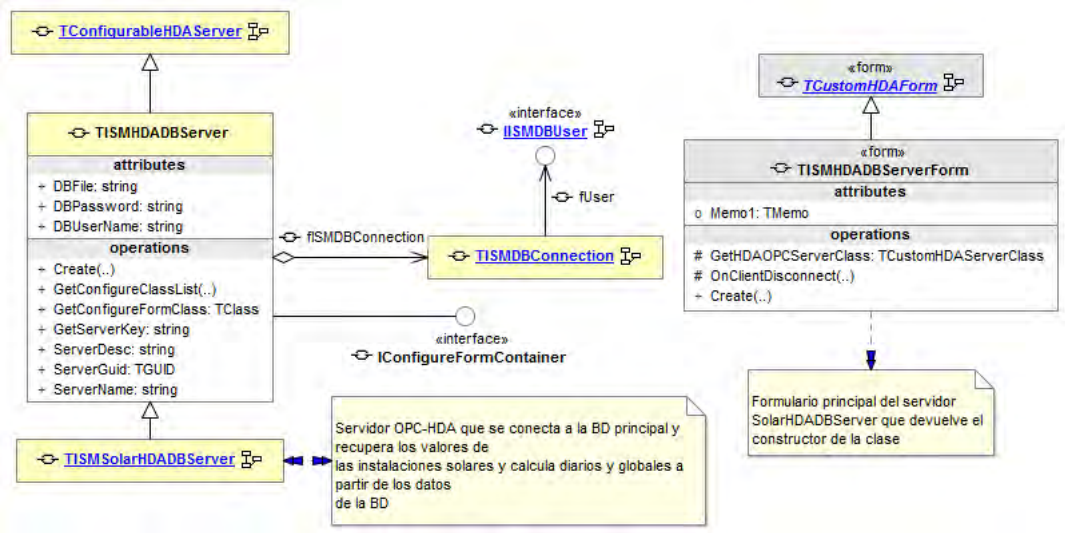

Figura 9.5: Servidor OPC-HDA con la lógica de gestión del dominio fotovoltaico

por lo tanto acceder desde esta clase a toda esa información.

El siguiente caso ha sido particularizar esa clase en una específica que realiza las siguientes funciones:

1. Publicación por parte del servidor OPC-HDA de los canales necesarios para la gestión de una instalación fotovoltaica
2. Conexión con la base de datos (ver capítulo 6) utilizando los parámetros apropiados suministrando dichos valores a través del marco de trabajo
3. Cálculo de los canales virtuales definidos a partir de la información de los canales básicos de todos los dispositivos de cada planta

En la figura 9.6 puede verse el diagrama de flujo que representa el funcionamiento del servidor OPC a la hora de publicar la información de los canales de todo el sistema. En ella puede verse cómo se recorre toda la base de datos buscando los grupos de instalaciones, las instalaciones y los dispositivos que las conforman y cómo se añaden los canales que va encontrando para poderlos a disposición del cliente OPC.

De esta manera, utilizando cualquier cliente OPC-HDA, o cualquier herramienta desarrollada con este marco de trabajo, es posible inspeccionar las variables que dan información sobre el estado energético de la instalación ya que estas se calculan en función de la información suministrada por los dispositivos que la conforman, cuya información ha sido volcada en la base de datos anteriormente, figura 9.7.

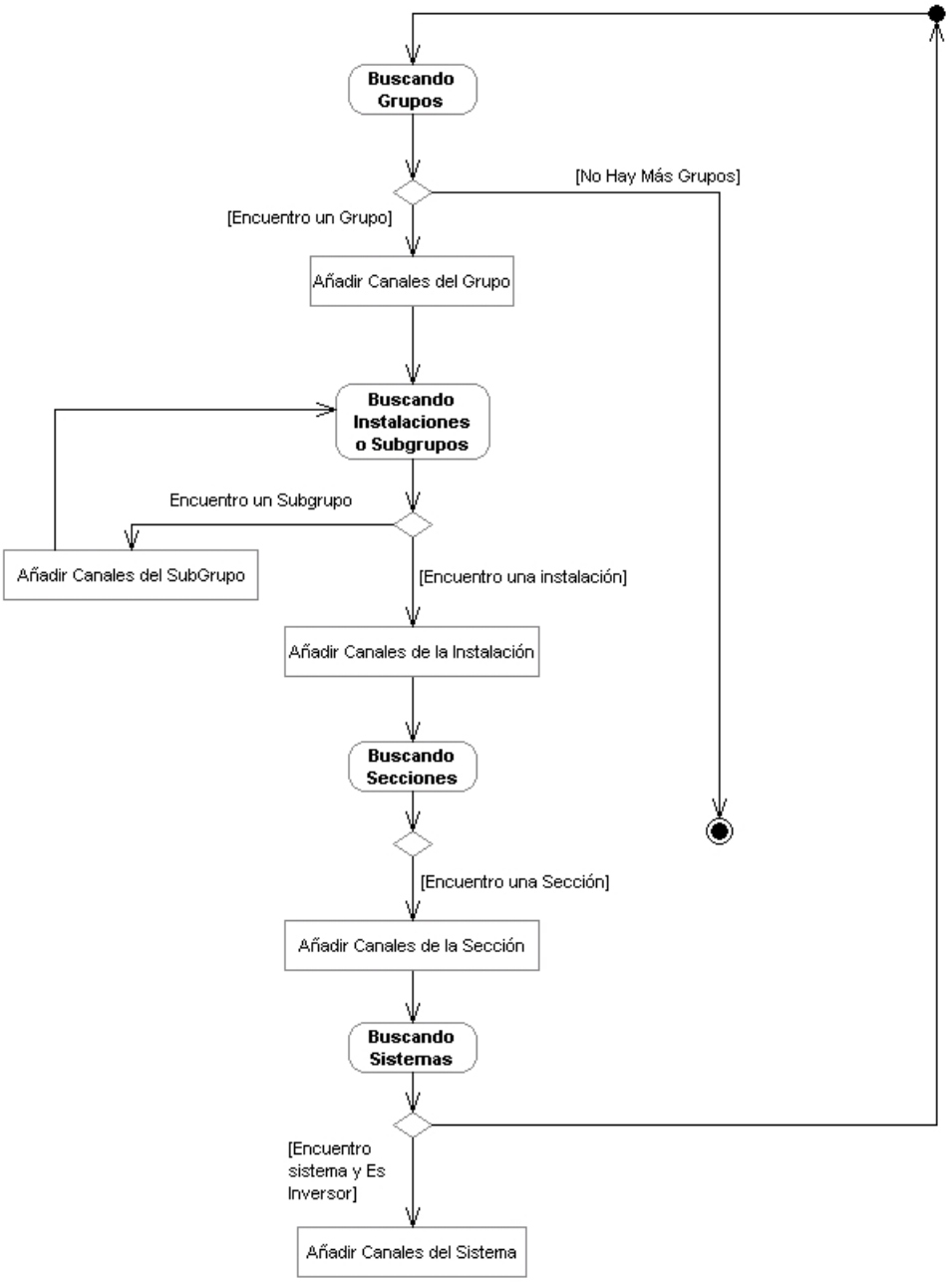

Figura 9.6: Diagrama de estados de publicación de canales del servidor HDA de base de datos

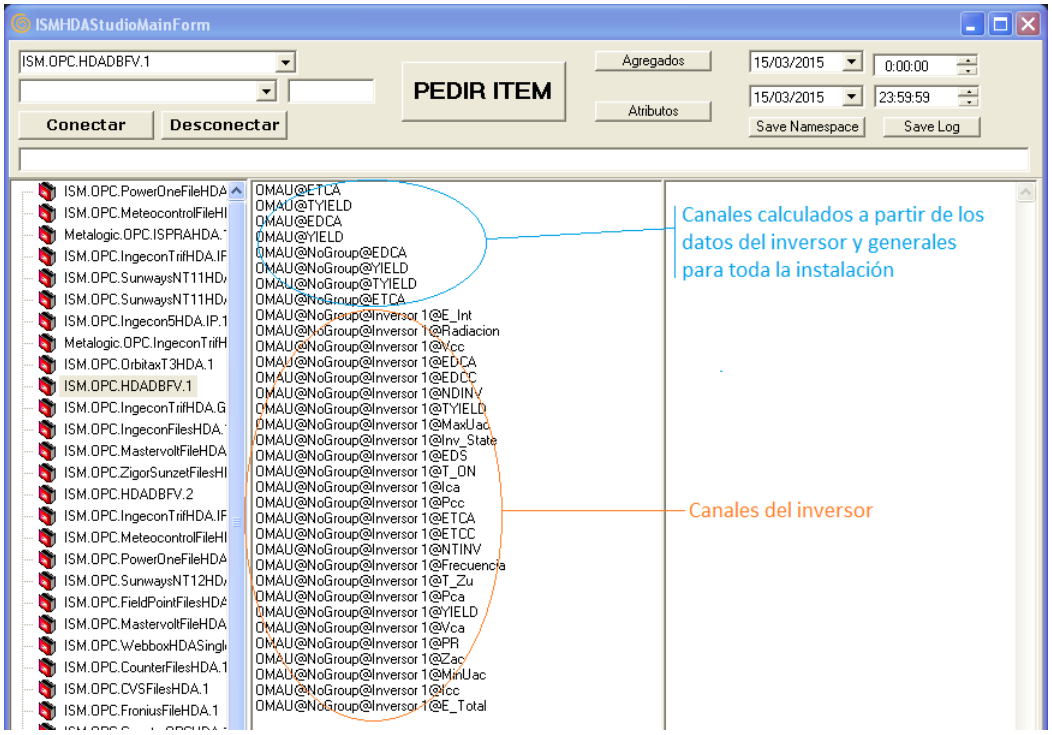

Figura 9.7: Cálculo de canales fotovoltaicos para una instalación determinada

Una vez encontrados y publicados los canales físicos de los dispositivos reales y los canales virtuales definidos para la gestión energética, estos están disponibles tanto para las herramientas generadas con este marco de trabajo como para cualquier cliente estándar OPC-HDA. En el caso de pedir la información de los canales virtuales definidos, el servidor desarrollado conoce la forma de calcularlos y efectúa dicho cálculo utilizando la información de los canales físicos y de los canales virtuales que necesite.

9.3.2. Inclusión de modelos de evaluación y predicción

La evaluación y la predicción de la producción de una planta energética de pequeña y media potencia en el marco de trabajo que se propone en esta tesis se basa en la inclusión de los modelos correspondientes en el sistema utilizando el estándar OPC. En el capítulo 8 se describen los modelos que se han desarrollado para realizar estas tareas. Son modelos que se basan en la utilización de técnicas de aprendizaje automático que permiten la incorporación de las características propias de cada instalación en los modelos, lo que permite ajustar estos al funcionamiento real de cada planta.

El modelo de cada instalación se construye utilizando diferentes fuentes de datos. Normalmente, cuando se desarrolla un modelo utilizando un conjunto de datos se hace en un entorno controlado en el que la fuente de datos suele ser única. Sin embargo, cuando estos modelos se integran en diferentes plantas se debe resolver el problema de integrar tanto los modelos previamente propuestos como los datos que se siguen registrando, de manera que el modelo sea capaz de reflejar el comportamiento y posible

evolución del sistema.

Esto es, por ejemplo, importante en el caso de ciertas instalaciones de energía en el que el generador puede ir degradándose con el paso del tiempo (como ocurre con los sistemas de energía solar fotovoltaica, en los que el proceso de degradación de los módulos es normal y dependiente de la tecnología utilizada). Para resolver estas cuestiones, la solución que se propone en esta tesis se basa implmentar estos modelos y el acceso a datos utilizando la tecnología OPC. En este caso, el modelo de evaluación puede ser tratado como un elemento más del sistema. En la figura 9.8 se muestra un esquema genérico de la la arquitectura que integra la utilización de modelos de evaluación y predicción con el sistema general de monitorización y gestión.

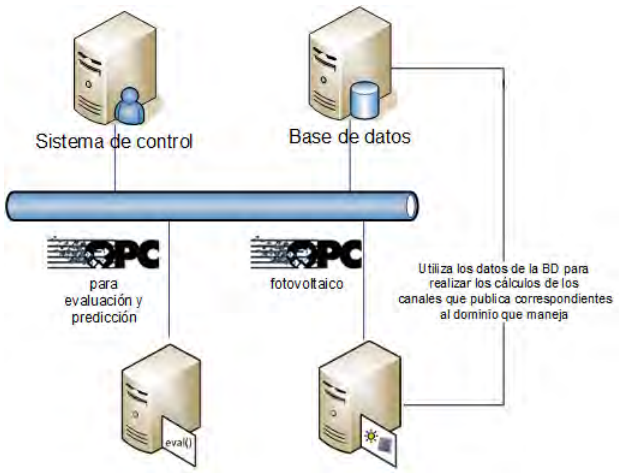

Figura 9.8: Inclusión de modelos de predicción y evaluación

En este esquema se muestra la capa de comunicación que permite que los datos se extraigan de los dispositivos utilizando la tecnología OPC HDA (OPC Historical Data Access). El gestionar el sistema de evaluación como si de otro dispositivo de la planta se tratara permite resolver un problema de manera genérica y el cambio de los modelos a utilizar puede ser fácilmente reemplazado.

La decisión de incorporar los modelos de evaluación y predicción implica que estos se asimilan a un nuevo dispositivo que genera los datos necesarios utilizando la información de la base de datos. El recipiente utilizado para el modelo se comporta como un cliente OPC que puede acceder a todos los datos en la base de datos como si se tratara de cualquier dispositivo de conexión para el acceso de base de datos a través de la tecnología OPC. Además, el contenedor se comporta como un servidor OPC que le permite ser fácilmente integrado en el sistema. Los posibles cambios en los modelos que se utilizan solo afectarán al servidor OPC que los implementa y no requerirán ningún cambio en la arquitectura del sistema.

Esta configuración permite que los modelos que utilizan datos de diferentes dispositivos y fuentes puedan ser reutilizados en otros sistemas cambiando solo el contenedor de estos modelos. Otra ventaja adicional es que los modelos que se desarrollen son independientes del lenguaje de programación ya que la interacción con el resto del sistema

no se hace a través de una librería o servicio sino usando la interfaz estándar OPC.

9.4. Conclusiones

En este capítulo se ha analizado cómo utilizar el marco de trabajo genérico para desarrollar aplicaciones de gestión de sistemas de energía solar fotovoltaica. La propuesta que se hace se basa en modelizar el dominio como si de un nuevo dispositivo se tratase y hacer que dicho dispositivo virtual efectúe la valoración de funcionamiento del sistema basándose en la información del modelizado de la planta y de la información del resto de dispositivos. Como se ha comprobado, esta solución permite incluir parámetros y fórmulas de cualquier dominio energético. Con la propuesta establecida en este capítulo, se trata el dominio de la energía solar fotovoltaica como un dispositivo haciendo uso de la tecnología OPC-HDA que permite añadir una gestión de la información desde y hacia el marco de trabajo, al gestionar la información de los canales, procesarlos y volver a almacenarlos en el sistema. El servidor OPC encargado de gestionar la información se ha desarrollado utilizando las clases e interfaces proporcionados por el marco de trabajo.

Otra de las conclusiones de este capítulo es que es posible incluir los modelos de predicción y evaluación de sistemas fotovoltaicos en el marco de trabajo asimilando los mismos a un nuevo dispositivo que genera toda la información necesaria. Así, los posibles cambios en los modelos que se utilizan afectarán solo al servidor OPC que los implementa y no será necesario modificar la arquitectura del sistema. Se han desarrollado componentes que permiten la integración de estos modelos en el marco de trabajo, de manera que se pueden generar programas de gestión que incluyan la generación automática de este tipo de tareas.

"La principal causa de complejidad en el software es que los fabricantes implementan casi todas las características que solicitan los usuarios"

– Niklaus Wirth

10

Aplicación del marco de trabajo para el desarrollo de herramientas para la gestión de sistemas energéticos

10.1. Introducción

A partir de la arquitectura y marco de trabajo presentados en el capítulo 3, el objetivo de este capítulo es hacer una propuesta que sirva para el diseño y creación de herramientas y aplicaciones a partir del marco de trabajo propuesto con las que se puedan realizar todas las tareas relacionadas con la gestión de sistemas energéticos de pequeña y media potencia de energía solar fotovoltaica. En las siguientes secciones se detallan las funcionalidades de cada una de estas herramientas y se muestran algunos ejemplos de las herramientas que se pueden desarrollar en el marco de trabajo. Como aportación importante, hay que señalar que se trata de herramientas que se han implementado y que los resultados que se presentan en este capítulo son de aplicaciones que están operativas. No se presentan en general los detalles de implementación, solo en el caso de que supongan una aportación relevante que pueda ser útil para otros sistemas.

A partir del marco propuesto, las aplicaciones necesarias para la gestión de sistemas energéticos que deben diseñarse y construirse deberían permitir, al menos, realizar las siguientes tareas:

- Modelizado visual de sistemas, incluyendo todos los componentes y conectores

incluidos en cada sistema.

- Sincronización con la plantas de un sistema.
- Visualización de datos de los dispositivos y sistemas.
- Evaluación y predicción del funcionamiento de sistemas.
- Generación de informes del funcionamiento de sistemas.
- Envío de mensajes de correo electrónico y/o sms con alarmas y eventos para los gestores de los sistemas.
- Publicación web que incluya información sobre el funcionamiento de las plantas

10.2. Metodología de desarrollo

Haciendo uso de las clases e interfaces proporcionadas por el marco de trabajo propuesto y con la ayuda de los algoritmos y flujos de control incluidos, se han desarrollado diferentes aplicaciones que dan respuesta a las necesidades de gestión y monitorización de sistemas energéticos. Esto por un lado, valida la efectividad del trabajo desarrollado, y por otro lado da soluciones reales a problemas reales ya que su utilización está siendo aplicada a casos existentes.

Gracias a la utilización de las metodologías ágiles, la aparición de nuevas funcionalidades durante la aplicación a casos reales de estos sistemas, ha permitido poder incorporar las demandas de manera sencilla haciendo uso de las técnicas de refactorización y al desarrollo dirigido por pruebas (ver 2.3).

10.3. Herramienta integral para gestión de sistemas energéticos

El marco de trabajo propuesto proporciona las piezas y herramientas necesarias para construcción de cualquier software de gestión energética. Es por ello que se ha desarrollado una herramienta que aglutina todas las funcionalidades necesarias haciendo uso de las facilidades que proporciona un desarrollo basado en este tipo de metodología.

En la figura 10.1 se muestran un conjunto de herramientas que permitirá validar el marco desarrollado y disponer a su vez de una solución completa para la gestión de sistemas energéticos.

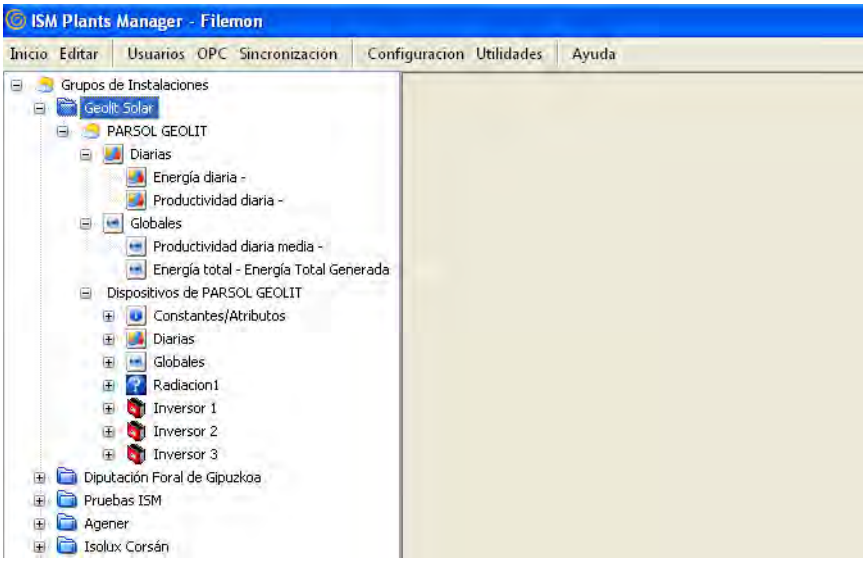

Figura 10.1: Vista general de la gestión de instalaciones energéticas

Sobre una aplicación básica y utilizando las clases y los comandos que se pueden aplicar sobre ellos (ver 7.4), se dispondrá de las funcionalidades necesarias para la gestión de las instalaciones.

10.3.1. Gestión de instalaciones energéticas

El alta de una nueva instalación genera una serie de acciones automatizadas proporcionadas por el marco de trabajo. Una es la creación de las clases e interfaces apropiadas en tiempo de ejecución y otra es el almacenamiento automático de dicha información en la base de datos de trabajo seleccionada.

Los datos comunes de descripción y localización pueden ser suministrados para así disponer de un catálogo lo más correcto posible de la información de la instalación energética que estamos incluyendo en el sistema, figura 10.2.

Es también posible y deseable la organización de las instalaciones en diferentes grupos para una mejor gestión y control de las mismas, figura 10.3.

El siguiente paso tras el alta de una nueva instalación es la de indicar los dispositivos que la conforman.

10.3.2. Gestión de dispositivos de una instalación

Para modelizar un sistema energético o instalación hay que crear una representación de todos los elementos que conforman el sistema utilizando instancias de las clases que

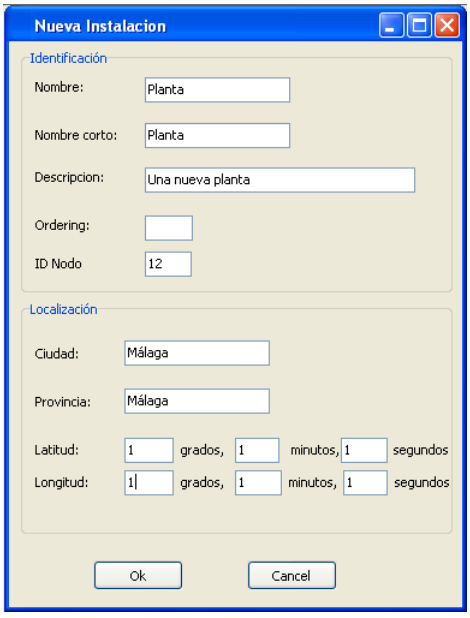

Figura 10.2: Formulario de alta de una instalación

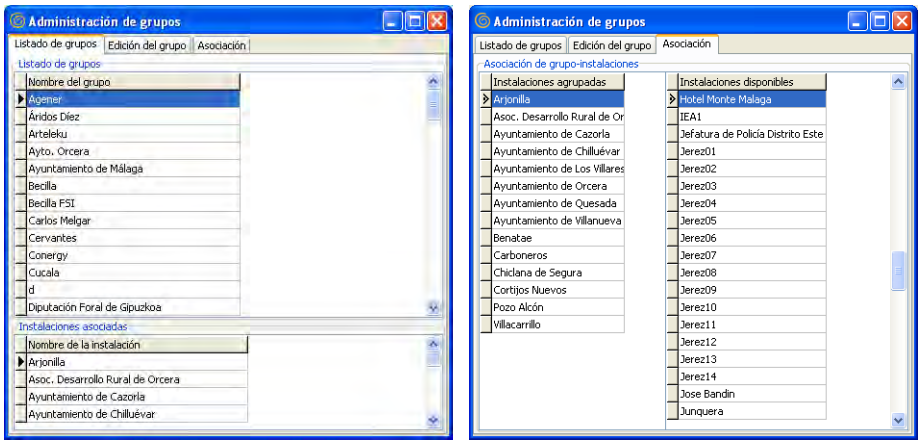

Figura 10.3: Agrupación de instalaciones en grupos

implementan las interfaces del marco de trabajo. Los elementos más numerosos a la hora de modelizar y que más tiempo conlleva son las medidas y los dispositivos asociados.

Para cada dispositivo integrado en el sistema es necesario disponer de toda la información de sus canales así como de información interna propia del dispositivo. Esta tarea puede ser bastante tediosa y repetitiva si se parte de cero a la hora de modelizar los dispositivos de cada sistema. Además, la mayoría de los dispositivos son comerciales y pueden encontrarse en muchas instalaciones distintas.

Por ello, en el marco de trabajo se ha dotado la capacidad de poder crear una librería de dispositivos con sus canales y medidas correctamente y completamente modelizadas. De esta manera, cuando se definen y añaden los dispositivos durante el modelizado de una instalación se puede utilizar este repositorio previamente creado. En la figura 10.4 se muestra una vista del repositorio de dispositivos predefinidos que se ha creado en el

marco de esta tesis.

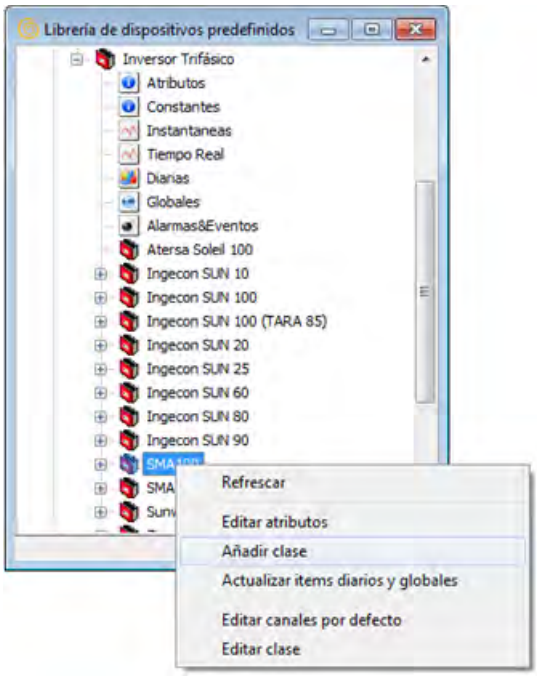

Figura 10.4: Librería de dispositivos predefinidos

Esta librería de dispositivos permite la inclusión de nuevos dispositivos y medidas asociados a los dispositivos predefinidos lo que permite diseñar de una manera rápida los sistemas energéticos a partir de esa librería. Esta librería predefinida es fruto de la utilización de dispositivos reales en instalaciones reales.

Las clases e interfaces desarrolladas para disponer de un repositorio de dispositivos en el marco de trabajo se encargan principalmente de disponer de un mecanismo de persistencia de datos sobre la base de datos. Así, el modelizado de un dispositivo y sus canales asociados se puede almacenar y recuperar de la base de datos para copiarse dentro del dispositivo físico que se pretende modelizar como se muestra en la figura 10.5.

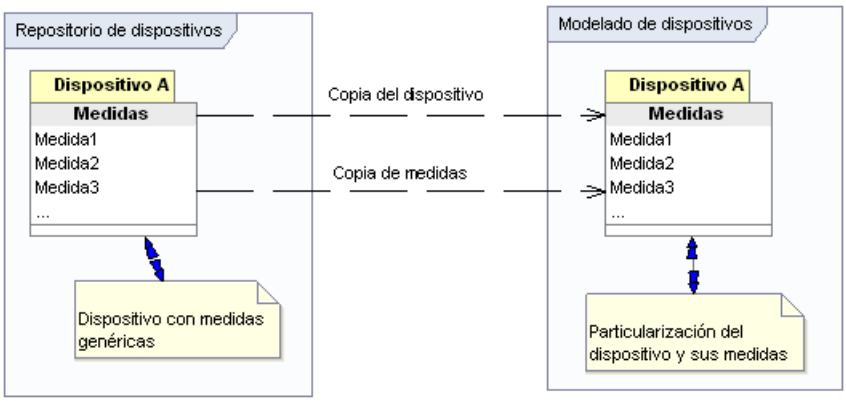

Figura 10.5: Personalización de medidas y dispositivos

En el copiado se crea el dispositivo y sus medidas pero al tratarse de una copia y no una referencia al repositorio, existe la posibilidad de particularizar los atributos del dispositivo, cambiar los tipos de las medidas, sus nombres e incluso extender o reducir el número de medidas del dispositivo. En la figura 10.6 se muestra un ejemplo de cómo se crea un dispositivo a partir de uno predefinido.

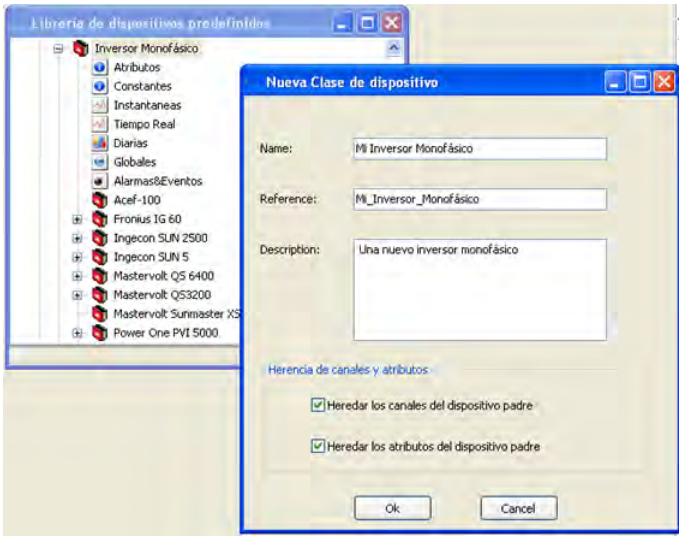

Figura 10.6: Formulario de alta de dispositivos

Por otra parte, existen dispositivos que están relacionados entre sí. Por ejemplo, en plantas de energía solar fotovoltaica, un inversor siempre estará relacionado con un generador. Estas relaciones también deben ser modelizadas. Para poder establecer la relaciones que existen entre dos dispositivos se ha creado el comando *conectar*. Además, también es interesante que se pueda, en cada momento, poder verificar qué sistemas están conectados a un dispositivo. Para ello, se propone la creación de un componente visual que permita acceder a esta información. Se puede visualizar la información de qué dispositivos están conectados al sistema que se especifique, como puede verse en la figura 10.7.

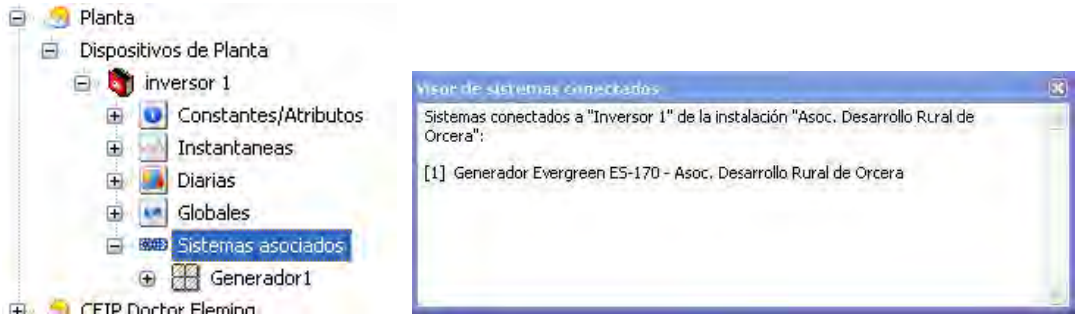

Figura 10.7: Árbol con los dispositivos conectados a un dispositivo y componente para la visualización de los sistemas conectados a un dispositivo

10.3.3. Gestión de medidas de los dispositivos

La base de toda la información de una instalación la tienen las medidas o canales de información de los dispositivos. Para disponer de ellas es necesario definir parámetros como su tipo o el canal de comunicación real que suministrará los valores de estos canales. Así, las medidas definidas en cada dispositivo dejarán de ser una mera descripción de los canales de los mismos a ser las vías de comunicación desde donde fluye los datos correspondientes al funcionamiento de instalación, figura 10.8.

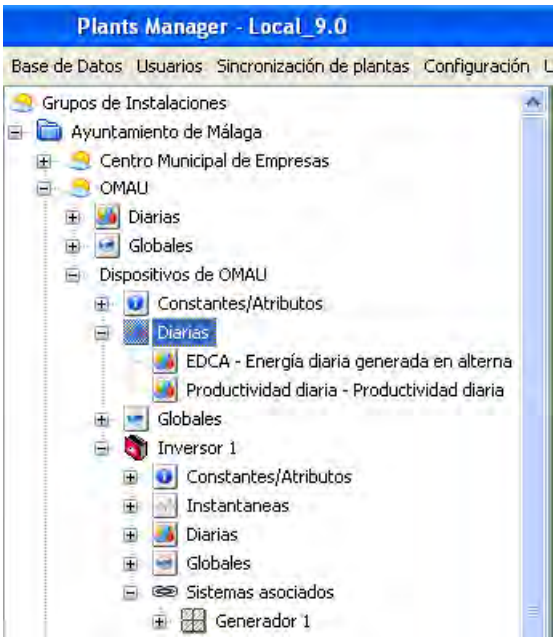

Figura 10.8: Gestión de instalaciones y dispositivos

Una de las tareas que se debe realizar en la modelización de un planta, relacionada con las comunicaciones y recuperación de datos, es la configuración de los items OPC de los distintos dispositivos incluidos en la planta. Sin una correspondencia entre los canales definidos y los mecanismos de comunicación reales no se dispondría de la información correspondiente. Por ello, se ha desarrollado un componente visual que permite la edición de los items OPC en el entorno de modelizado. En la figura 10.9 se muestra este componente.

Otra de las tareas frecuentes que debe realizarse en el diseño de un sistema de gestión, una vez desarrollados e incorporados los servidores OPC al marco de trabajo, es la configuración de los mismos para cada instalación en la que vayan a utilizarse, según los dispositivos de esa instalación. Por ello, es importante poder contar con herramientas que faciliten esta tarea. A partir del marco de trabajo, puede fácilmente construirse una aplicación que permita la configuración del servidor OPC de cada dispositivo que se encuentre en la instalación que se va a modelizar. Una vez construida la parte visual, solo es necesario almacenar en la base de datos la información específica de la configuración de ese OPC para la instalación que se está modelando.

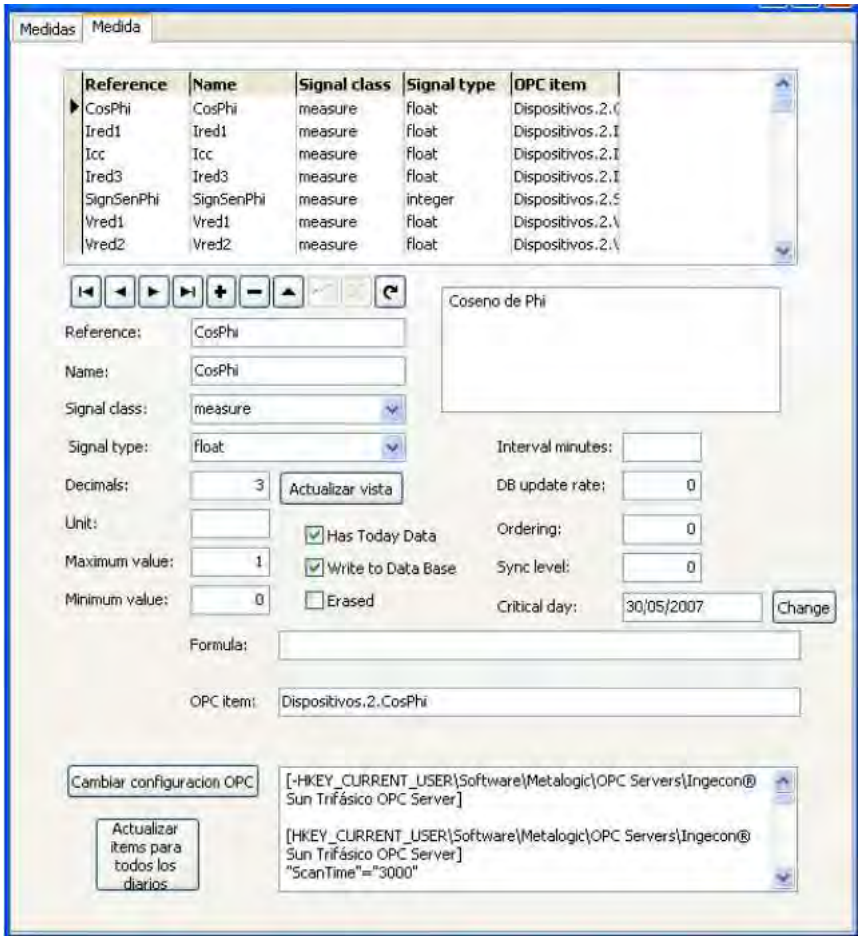

Figura 10.9: Componente para la edición de un item OPC

El marco de trabajo proporciona los servidores OPC disponibles que son los encargados de descargar los datos de cada dispositivo, figura 10.10. Cada una de las medidas definidas de cada dispositivo deben ser asociadas a canales de estos servidores OPC que en última instancia son los responsables de recuperar esta información.

Además, una cuestión importante es la referente a la reutilización de los servidores en diferentes instalaciones. Dispositivos similares compartirán servidores OPC similares salvo los parámetros de comunicación responsables de la conexión a una u otra planta. Es por eso que en el marco de trabajo se proporciona la capacidad de poder lanzar una diferente configuración para cada servidor en relación a la instalación a la que queremos conectar como puede verse en la figura 10.11.

De esta manera, cuando sea necesario conectarse a una planta, y por ende, a un dispositivo determinado utilizando un protocolo que implementa un servidor OPC, se lanzará una configuración determinada que hará que el comportamiento de dicho servidor sea apropiado para la conexión con dicha instalación en referencia a parámetros como número de teléfono, direcciones IP o parámetros del protocolo.

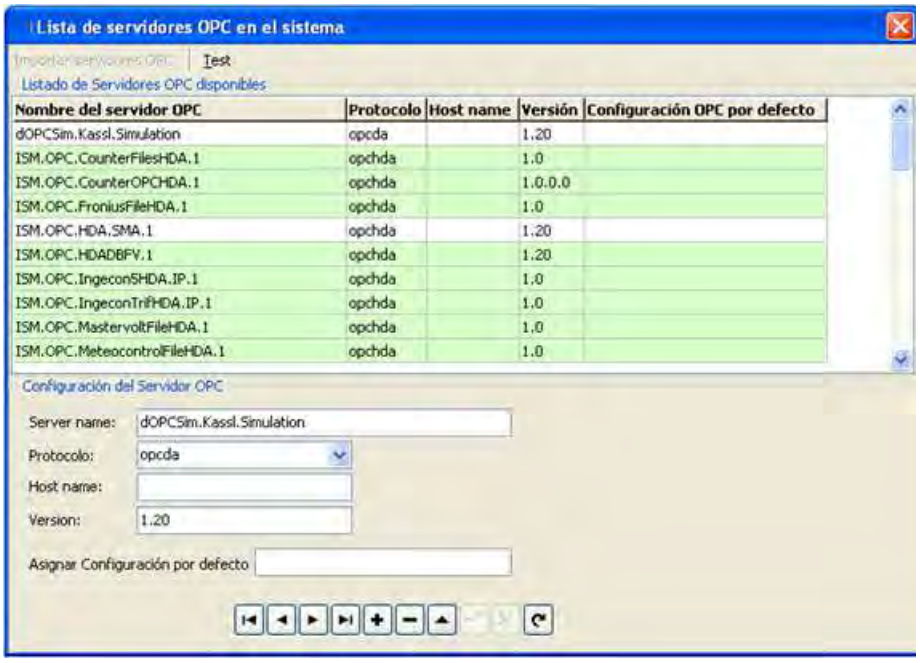

Figura 10.10: Librería de servidores OPC disponibles en el sistema

10.3.4. Gestión de usuarios y perfiles

Con las herramientas que el marco proporciona para la gestión de usuarios y roles, se ha integrado un mecanismo de alta de usuario y de perfiles con diferentes permisos asociados a comandos del marco de trabajo. Con esta funcionalidad se permite o se deniega la utilización de ciertos comandos por parte del usuario que está utilizando el sistema para establecer así diferentes niveles o privilegios de acceso, figura 10.12.

Durante el alta de un usuario y su asociación a una planta determinada se puede establecer el nivel de acceso o rol que desempeñará dicho usuario respecto a dicha planta, permitiendo así disponer de diferentes niveles de acceso para cada instalación.

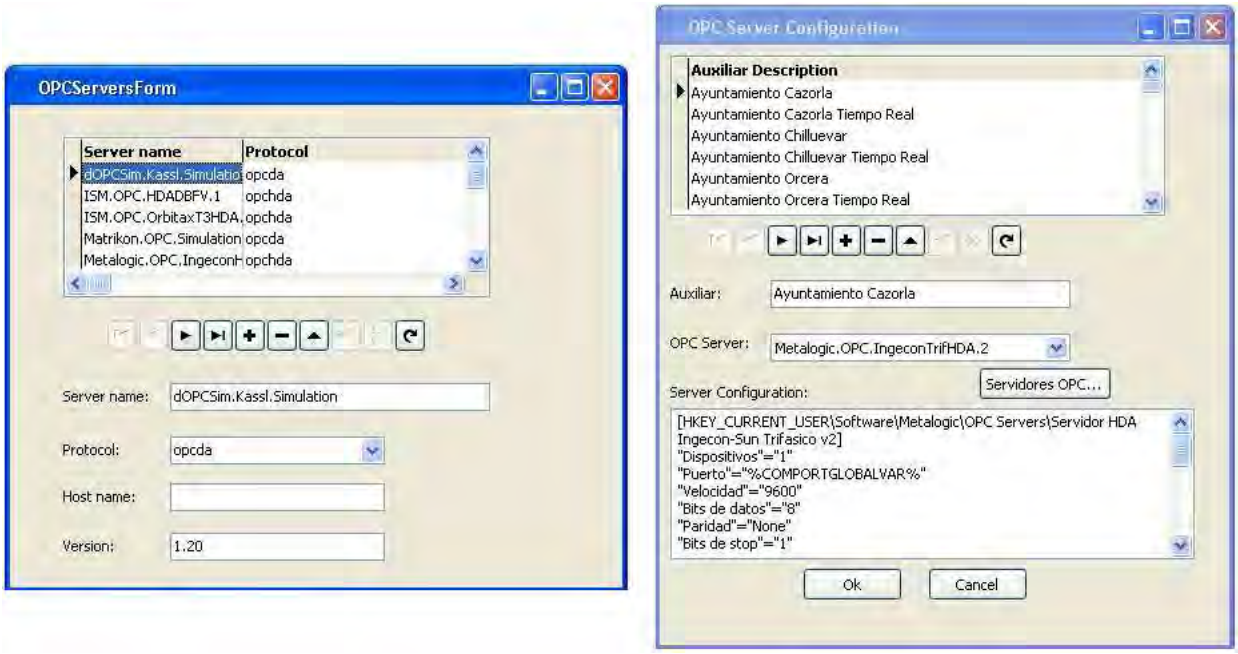

Figura 10.11: Configuración de servidores OPC

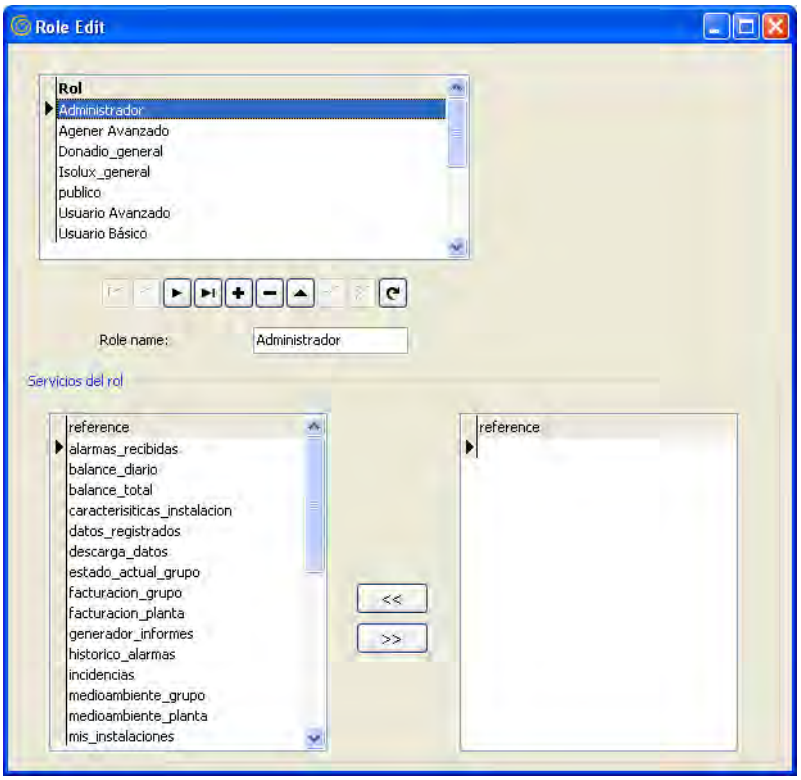

Figura 10.12: Roles y permisos sobre el marco de trabajo

10.3.5. Sincronización de instalaciones

Disponer de la información de todos los dispositivos de una instalación conlleva establecer una estrategia para que la información extraída sea lo más coherente posible frente a problemas de comunicación o de otra índole. Como se ha descrito en 6.6, los algoritmos presentados ofrecen dichas cualidades y se encuentran implementados dentro del marco de trabajo. Para cualquier desarrollo, en este caso, desde el sistema integral que se desarrolla con el marco, es posible instanciarlo y disponer así de la información de todos los dispositivos de las instalaciones que se están gestionando y disponer de dicha información en la base de datos a disposición de las clases que lo necesiten para la evaluación o para la inferencia de cualquier tipo de conclusión. Esto se consigue con el componente, que se muestra en la figura 10.13, que implementa los algoritmos presentados en el apartado 6.8.

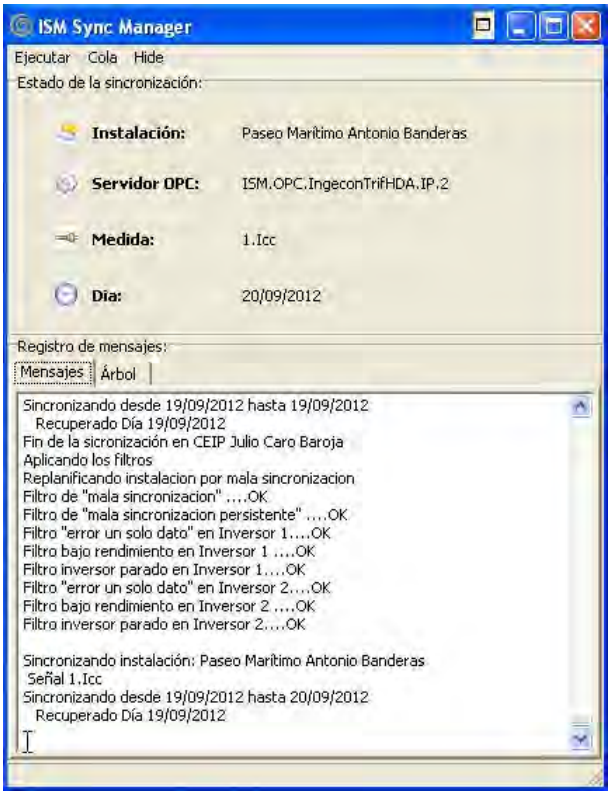

Figura 10.13: Sincronizador de instalaciones

10.3.6. Visualización de datos de las instalaciones

En los apartados anteriores se ha descrito cómo modelizar las instalaciones con todos sus dispositivos además disponer de mecanismos para la descarga de los datos. Una parte fundamental es la de poder visualizar la información de las instalaciones. El hecho de tener la información almacenada de manera homogénea permite el acceso a ella a través de las clases e interfaces suministradas por el marco de trabajo. De

esta manera, es posible realizar cualquier interfaz de usuario para poder visualizarla y recuperarla en diversos formatos como puede verse en el la figura 10.14.

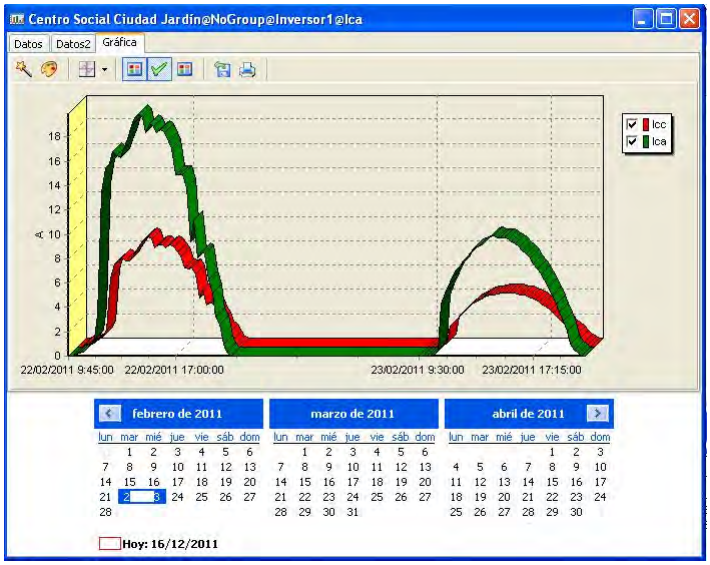

Figura 10.14: Visualización de canales en la aplicación

El acceso a la información puede ser realizado a través de cualquier lenguaje web para poder visualizarlo en navegadores u otros dispositivos sin demasiada dificultad debido a que la información se encuentra almacenada y accesible de manera relacionada. En la figura 10.15 se muestra un ejemplo de cómo se pueden visualizar los canales en la web. Como se puede observar, además de la información númerica es a veces necesario tener acceso a otro tipo de información, en alarmas o eventos que puedan lanzar los dispositivos y son importantes recoger para su posterior análisis y estudio o actuación inmediata.

Figura 10.15: Visualización de canales en la web

10.4. Caso de estudio: Sistema de monitorización integral para instalaciones dispersas

A partir de la extensión del marco de trabajo presentada en los apartados previos, se ha desarrollado un sistema de gestión que integra información de 40 instalaciones de energía solar fotovoltaica dispersas. Cada instalación es de un tipo diferente con dispositivos diferentes y la potencia pico de las mismas oscila entre 1.8 y 40 kW. La necesidad de disponer de un sistema de control y monitorización de las mismas y debido a la total heterogeneidad de las mismas, hacía este de un problema que podría ser totalmente resuelto con la metodología y arquitectura propuestas en este trabajo.

Para dar una solución al monitorización y gestión de este grupo de instalaciones se planteó el diseño de la arquitectura siguiendo la estructura propuesta en 9.2 y cuyo esquema físico se muestra en la figura 10.16

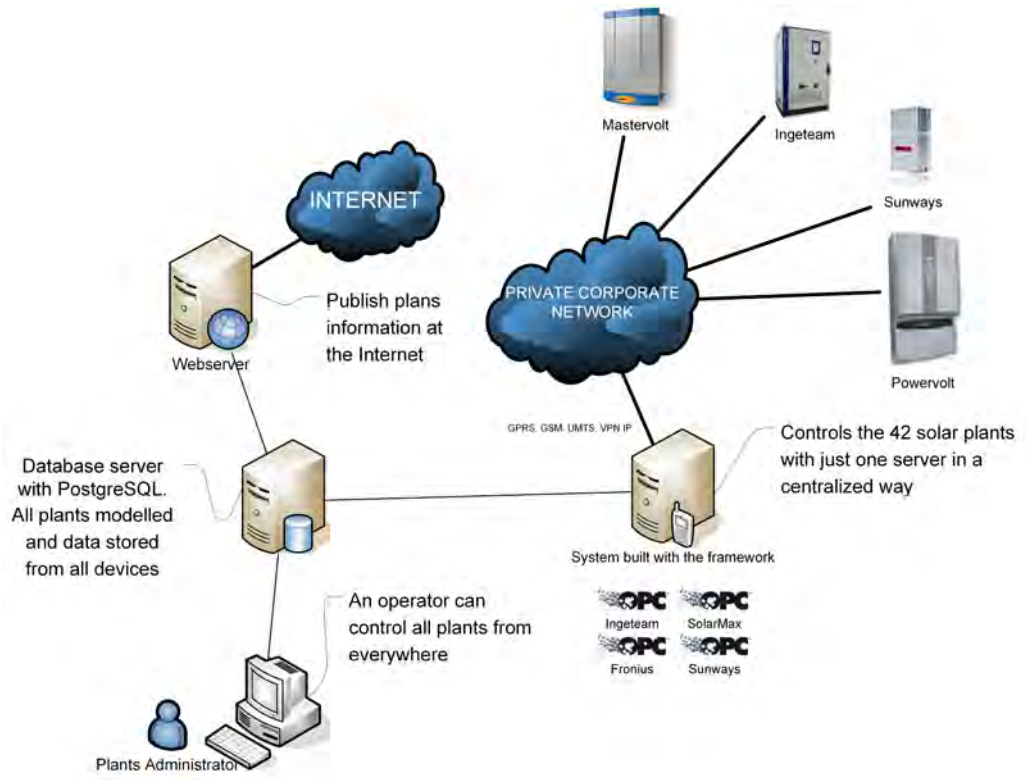

Figura 10.16: Esquema físico del sistema

En estas instalaciones había dispositivos diferentes y de diferentes marcas. Algunos de ellos se enumeran a continuación:

- Inversores: Ingecon Sun 10, Ingecon Sun 20, Sunways NT6000, Sunways NT4000, Sunways NT10000, Mastervolt QS3200, Mastervolt QS6400, Mastervolt XS6500, Power One PVI 5000, SolarMax 4000C, Fronius IG60, Sunny Boy SWR-2000NE.

- Sensores de radiación: células calibradas y piranómetros.

- Sensores de temperatura de módulos: PT-100.

- Dispositivos para establecer las comunicaciones: módem GPRS interno (en el inversor), módem GPRS externo.

- Sistemas de adquisición de datos (Data loggers): algunas instalaciones tienen su propio sistema local de adquisición de datos. Los registradores de datos instalados son: Max Web(módem GPRS interno), Easy Fronius, Solar Log 1000, MasterVolt Data Control (servidor ftp), Sunny Webbox.

Además, la forma de comunicación con las plantas era igualmente diferente con distintos escenarios:

- Acceso a la intranet local

- Comunicaciones GPRS o UMTS (módem externo)

- Acceso a Internet (con funcionalidades de cliente VPN-IPsec).

Así, según el tipo de dispositivos incluidos en cada planta, la aplicación que se ha desarrollado utiliza el servidor OPC que le corresponda a ese dispositivo (ver 4) añadiendo además la configuración apropiada para cada tipo o medio físico de comunicación (ver 7.6).

En la figura 10.17 se muestra la arquitectura de la aplicación desarrollada que encaja perfectamente con la arquitectura que proporciona el marco de trabajo (ver 3.2).

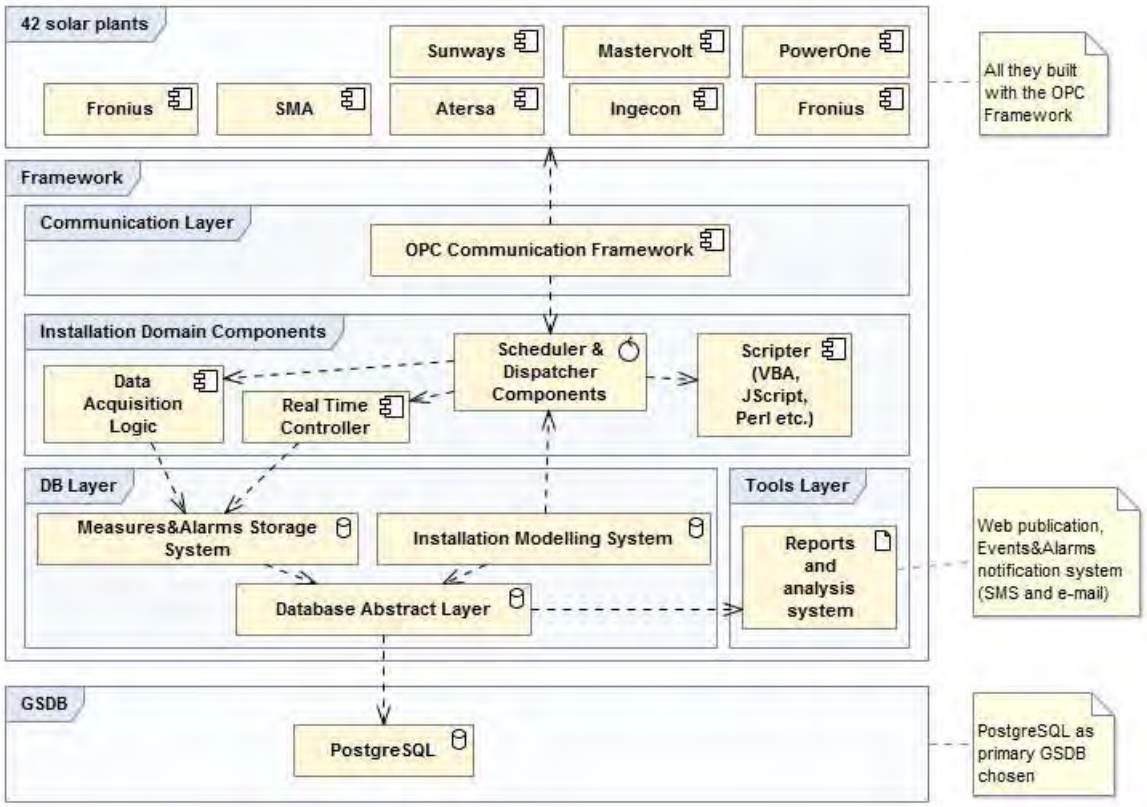

Figura 10.17: Componentes incluidos en el programa de gestión y su interrelación con los subsistemas de las plantas fotovoltaicas

La aplicación que se ha creado incluye un componente para la recuperación y almacenamiento de todos los datos de las plantas (gestor de las sincronizaciones), un componente para analizar y evaluar los datos según el modelo que se describe, un componente para la publicación web de la información de las plantas y un componente para el envío de alarmas y eventos.

En el desarrollo de los servidores OPC se han definido los canales de cada dispositivo y el dominio del modelo fotovoltaico se ha incluido dentro de un dispositivo simulado siguiendo la estrategia indicada en 9.3.

Para evaluar el funcionamiento de esta instalaciones, se han calculado las siguientes *medidas virtuales* como valores acumulados a partir de los valores instantáneos registrados:

- Energía diaria recibida en cada instalación.
- Energía diaria producida.
- Energía diaria suministrada a la red.

El valor de potencia de salida del inversor (P_j) en Eq.8.2 se ha obtenido en intervalos que varían de 5 a 15 minutos dependiendo de cada instalación. Además, se han calculado las *medidas virtuales* siguientes:

- Rendimiento diario final, Eq. 8.1.
- Energía estimada suministrada a la red, Eq. 8.4.
- Potencia de salida del inversor, Eq. 8.3.

En el caso de los módulos de silicio monocristalino, el valor del coeficiente γ en la Eq.8.3 es 0,48 %/^{o}C (este tipo de módulos es el que hay en las instalaciones). Todas las instalaciones han sido evaluadas utilizando la expresión Eq.8.6 particularizada para las medidas virtuales previamente listadas. Esta evaluación permite detectar problemas en el funcionamento de las plantas, a partir de los criterios definidos en la ecuación 8.6.

La figura 10.18 muestra los valores medios diarios del rendimiento final diario frente a la potencia instalada de cada instalación; cuyos valores varían entre 2.38kW$_p$ and 60kW$_p$. Los valores de rendimiento final diario obtenidos varían entre 2.26 y 5.18.

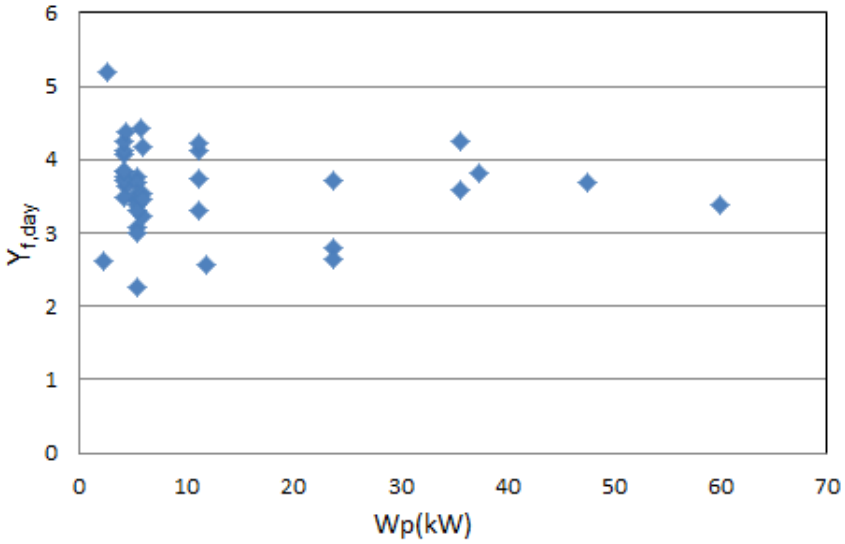

Figura 10.18: W_p vs $Y_{f,day}$ para todas las instalaciones analizadas

Como ejemplo de los resultados obtenidos con el procedimiento de evaluación propuesto, en la figura Figure 10.19 se muestran los valores del rendimiento final diario para cinco de las instalaciones analizadas, para el periodo desde el 28 de mayo al 19 de junio de 2013.

Los valores de rendimiento final diario en color verde corresponden a una instalación con un valor para este parámetro menor que el esperado, para el día 14 de junio.

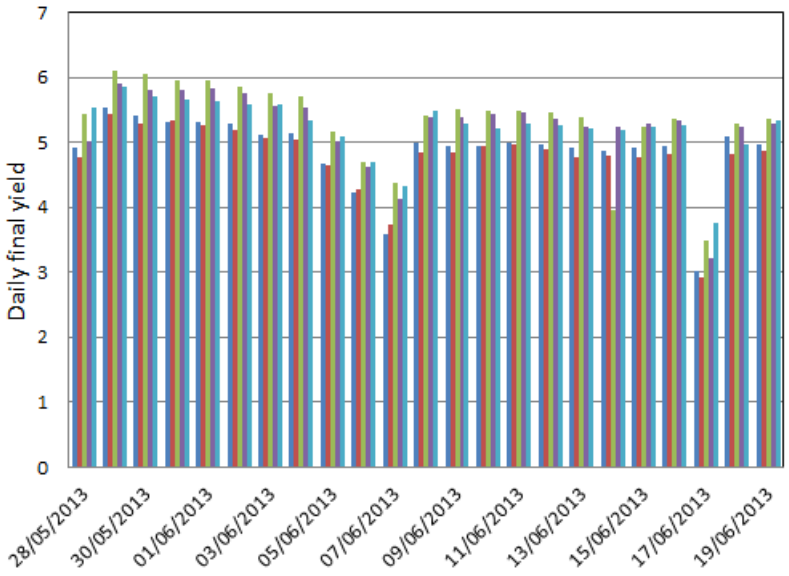

Figura 10.19: Rendimiento final diario para cinco instalaciones

La evaluación automática de esta instalación permitió detectar este problema. Así, cuando se detectan problemas de este tipo, la evaluación de la instalación hace que se pasen a evaluar los valores instantáneos de P_{CA} de acuerdo con el proceso descrito en la figura 8.2 para poder detectar las causas del problema. Este análisis detallado permite concluir que la instalación tuvo problemas con el inversor ese día, como puede observarse en la figura 10.20: en particular, las variaciones de tensión y/o frecuencia de la red eléctrica por encima de los rangos establecidos para el inversor han causado sus paradas (desconexiones) y por lo tanto las pérdidas en la producción de energía.

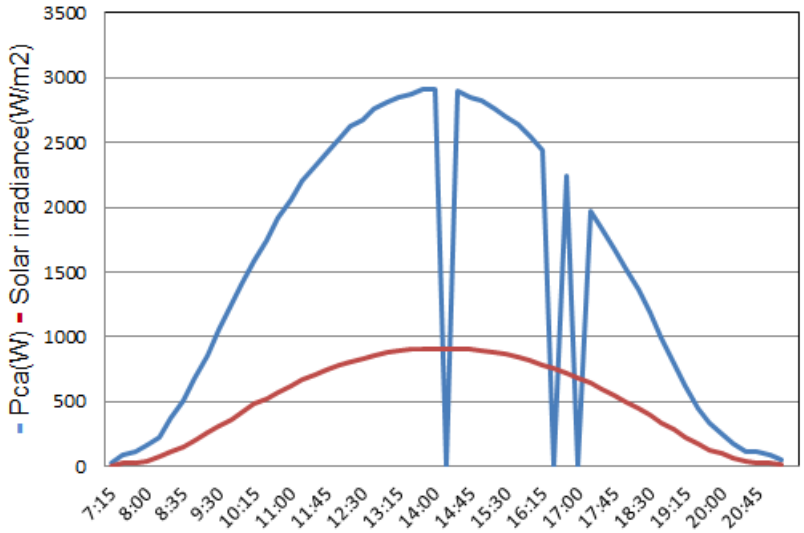

Figura 10.20: Valores registrados de P_{CA} y de irradiancia para un día con rendimiento final diario bajo

Para analizar los problemas detectados para una instalación, se pueden visualizar

los valores de la potencia medida, P_{CA}, frente a la estimada utilizando la ecuación 8.3, figura 10.21 . En la misma se han marcado con círculos todos los puntos que corresponden a situaciones de funcionamiento incorrecto detectadas, de acuerdo con la Eq.8.7.

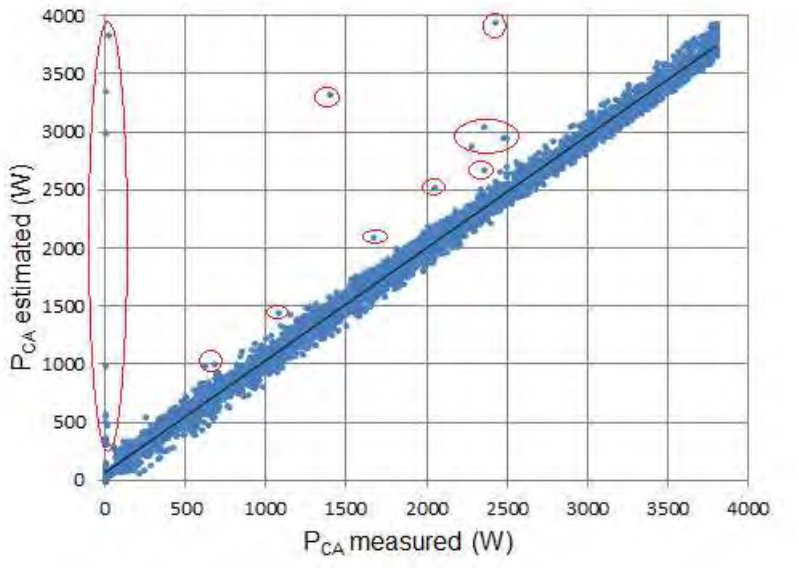

Figura 10.21: P_{CA} (medidas) versus P^*_{CA} (estimadas)

Utilizando la metodología propuesta, los problemas más frecuentes que se han detectado en estas instalaciones son:

A) $P^*_{AC} > 0$ y $P_{AC} = 0$. Estas situaciones corresponden a paradas del sistema, fundamentalmente debidas a a:

- fallos en la conexión a la red
- fallos del inversor.

B) $P^*_{AC} >> P_{AC}$. Estas situaciones corresponden a:

- fallos en el array fotovoltaico
- desconexión parcial del generador fotovoltaico (ramas de circuito abierto)

C) $P^*_{AC} > P_{AC}$. Esta situación se produce, principalmente, por la presencia de pequeñas sombras en el generados fotovoltaico.

Así, gracias a la aplicación creada a partir del marco de trabajo que se propone en este trabajo, la información de todas las plantas está accesible de manera uniforme proporcionando además una manera visual sencilla para mostrar y gestionar la información de todas las instalaciones y disponer de un sistema para la evaluación y vigilancia energética apropiado.

10.5. Conclusiones

En este capítulo se ha comprobado la capacidad del marco desarrollado para la gestión de sistemas energéticos aplicándolo a problemas reales. Gracias a este desarrollo, la resolución de este tipo de problemas se puede hacer de manera rápida debido a que el marco proporciona todas las piezas necesarias para la construcción de este tipo de sistemas dejando los detalles de la implementación de manera interna y permitiendo al desarrollador centrarse en proporcionar las funcionalidades deseadas.

11

Conclusiones y líneas de trabajo futuras

En este trabajo se ha propuesto la utilización de marcos de trabajo y arquitecturas software para el desarrollo de software que permita la gestión de sistemas energéticos. Esta arquitectura describe las partes de una sistema energético y la interacción entre ellas, así como los patrones que se utilizan para su composición y las restricciones de esos patrones.

Se han propuesto un conjunto integrado de clases e interfaces que pueden ser utilizadas por los desarrolladores de sistemas de adquisición de datos, gestión energética, evaluación del funcionamiento y predicción de la producción de sistemas energéticos para el desarrollo de soluciones software que den respuesta a estas necesidades. Gracias a lo cuál se consigue reducir el coste de desarrollo, su velocidad de desarrollo e implantación y la reutilización del software desarrollado.

Con la propuesta que se hace, de abordar la construcción de software para la gestión de sistemas energéticos a partir de un marco de trabajo, se consigue:

- Facilitar el desarrollo de nuevo software de gestión ya que se proporciona código que no se tendrá que reescribir.
- Evitar los detalles de bajo nivel.
- Minimizar los tiempos de desarrollo.
- Construir una arquitectura consistente entre aplicaciones.

Una de las características más críticas que debía tener en cuenta el sistema pro-

puesto era el rendimiento del mismo, ya que debía permitir desarrollar aplicaciones para sistemas reales donde existen dispositivos con requisitos cambiantes y de diversa naturaleza, es decir, no se pretendía disponer solo de un marco teórico sino que este marco de trabajo debía poder ser utilizado en instalaciones reales.

La arquitectura software que se propone en esta tesis se basa en la separación del software en varias capas, cada una de las cuales incluye una serie de componentes basados en interfaces especializados que ofrecen funcionalidades para la definición, modelizado, comunicación, almacenamiento, supervisión, evaluación y predicción del funcionamiento de sistemas energéticos genéricos. Además, tiene en cuenta las similitudes que presentan los sistemas energéticos genéricos, dejando las particularidades de cada tipo de sistema para su consideración como extensiones al marco de trabajo.

En esta arquitectura se han definido los estándares utilizados para el establecimiento de conexiones y coordinación entre los componentes de cada capa. Las capas propuestas son: Acceso a la base de datos, Sistema de almacenamiento, Capa abstracta de base de datos, Capa del dominio, y Capa de herramientas. La capa del dominio incluye el generador de scritpts, el sistema en tiempo real para la conexión física con las plantas, el sistema de adquisición de datos históricos, el planificador y el gestor de la comunicaciones.

Gracias a la arquitectura propuesta basada en capas o niveles se pueden construir soluciones para la gestión de instalaciones genéricas, donde cada capa es la responsable de cubrir cada una de las necesidades que surgen para la gestión de este tipo de instalaciones.

Las propuestas hechas para el modelizado de cada instalación y de cada dispositivo, la definición e implementación del protocolo de comunicaciones y la caracterización de cada canal de información hacen que se pueda disponer de un sistema de tratamiento de la información de la instalación que se vaya a gestionar.

Para los desarrollos hechos en este trabajo se han utilizado metodologías ágiles. En concreto se ha hecho uso de las técnicas de refactorización, el desarrollo dirigido por pruebas y la integración continua. La metodología de desarrollo que se ha propuesto, se basa en la utilización de la implementación frente a abstracciones. Por ello, el marco de trabajo desarrollado está formado por un conjunto de interfaces relacionadas y las clases que implementas dichas interfaces. Además, se ha propuesto una solución al problema del diamante en la arquitectura software, y para el dominio objeto del trabajo, puesto que se ha utilizado un lenguaje que no permite la herencia múltiple: la implementación por delegación de interfaces y la inyección de dependencias.

Haciendo uso de la composición de clases y la delegación de interfaces, el marco de trabajo puede extenderse y seguir un esquema desacoplado al no existir relaciones directas entre clases en favor de las relaciones establecidas entre interfaces. La inyección de dependencias resuelve la posible limitación de establecer dentro de una clase la decisión de qué implementación utilizar. El mecanismo propuesto es la inyección en

el constructor. Otro de los objetivos que se han conseguido es proponer un sistema de comunicaciones que permite una conectividad abierta utilizando estándares abiertos. El sistema de comunicaciones propuesto está totalmente separado de las demás funcionalidades que se incluyen en el marco de trabajo. Para ello, la propuesta se basa en la utilización del estándar OPC. Por otra parte, basándose en el modelo que se propone en esta tesis, la integración de modelos de evaluación y predicción en los sistemas de gestión energética que se desarrollen a partir del marco de trabajo propuesto, se hará partiendo del mismo esquema de abstracción que se propone para las comunicaciones. Se ha desarrollado un repositorio de servidores OPC que puede ser directamente utilizado en la gestión de sistemas energéticos.

Se ha hecho una propuesta de modelo que permite describir cualquier sistema genérico a partir de los componentes propios del dominio de instalaciones energéticas, y modelizar los elementos que conforman un sistema energético, desde la instalación de manera general hasta las medidas o puntos de información pasando por los diferentes dispositivos así como los posibles niveles de acceso o usuarios que tienen alguna capacidad de gestión sobre la planta. Se ha propuesto, finalmente, un modelizado adecuado para las medidas, que son la última información y, en definitiva, más importante, ya que son las responsables de disponer de los parámetros de funcionamiento de las instalaciones, y por lo tanto su gestión y su capacidad de configuración son quizás la parte más importante al basarse toda la información y toma de decisiones en el contenido de las mismas.

A partir del marco de trabajo general para sistemas energéticos se ha hecho una extensión del mismo para su utilización en sistemas de energía solar fotovoltaica. La propuesta que se hace se basa en modelizar el dominio como si de un nuevo dispositivo se tratase y hacer que dicho dispositivo virtual efectúe la valoración de funcionamiento del sistema basándose en la información del modelizado de la planta y de la información del resto de dispositivos. Como se ha comprobado, esta solución permite incluir parámetros y fórmulas de cualquier dominio energético

Para la parte de evaluación y predicción del funcionamiento de este tipo de sistemas se han desarrollado varios modelos. Por una parte, se ha propuesto un modelo estadístico que permite hacer una evaluación automática del funcionamiento de una instalación, a partir del análisis de varios parámetros que sirven para determinarlo. Por otra parte, se ha propuesto un modelo que permite predecir la energía que va a producir un sistema fotovoltaico a partir de la predicción de la radiación que recibirá. Este modelo solo utiliza información que se registra en una planta fotovoltaica, entre otra, los valores de radiación recibida los días previos. El modelo se construye a partir de tres estados. Para la construcción del mismo se ha utilizado un tipo especial de autómatas finitos probabilistas. El error del modelo propuesto es menor del 20 % frente a modelos tradicionales cuyo error es de un 25 % aproximadamente, lo que supone una mejora significativa. Estos modelos se han incorporado al marco de trabajo desarrollado.

La propuesta que se desarrolla en este trabajo permite la integración de las diferentes tecnologías y protocolos de comunicación utilizados en sistemas energéticos

así como la monitorización remota y la evaluación automática del funcionamiento de estos sistemas. Como parte fundamental y novedosa, el marco de trabajo desarrollado permite la integración en el sistema de componentes que automatizan las tareas de evaluación del funcionamiento de estos sistemas y la predicción de la producción de los mismos. Estos componentes incluyen modelos que utilizan técnicas de aprendizaje automático.

Finalmente, se ha comprobado la capacidad del marco desarrollado para la gestión de sistemas energéticos aplicándolo a problemas reales. Como ejemplo práctico de aplicaciones en las que el marco de trabajo puede ser muy útil se presenta distintos tipos de programas desarrollados para la supervisión, gestión y evaluación de instalaciones de energía solar fotovoltaica. Gracias a las propuestas de este trabajo, la resolución de este tipo de problemas se puede hacer de manera rápida debido a que el marco proporciona todas las piezas necesarias para la construcción de este tipo de sistemas.

Algunas de las líneas que quedan abiertas, y que pueden completar el trabajo aquí presentado son las siguientes:

- Utilizar el marco de trabajo propuesto para la gestión de otros tipos de sistemas energéticos, como pueden ser eólicos o termosolares.
- Profundizar y mejorar los modelos de evaluación del funcionamiento de sistemas energéticos, buscando modelos generales basados en parámetros estadísticos.
- Mejorar los modelos de predicción del funcionamiento de sistemas energéticos.

Bibliografía

[ADMC11] L.M. Ayompe, A. Duffy, S.J. McCormack, and M. Conlon. Measured performance of a 1.72 kw rooftop grid connected photovoltaic system in ireland. *Energy Conversion and Management*, 52(2):816–825, 2011. cited By (since 1996) 3.

[AOB76] R.M. Amundson, B.D. Osecky, and W. Bennett. A hierarchical network for data acquisition and control. *Industrial Electronics and Control Instrumentation, IEEE Transactions on*, IECI-23(1):76 –79, 1976.

[Ass14] Photovoltaic Industry Association. Global market outlook for photovoltaics 2014-2018. Technical report, Photovoltaic Industry Association, 2014.

[BD02] P.J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer Texts in Statistics, 2002.

[Be95] G. Blaesser and D. Munro (editors). Guidelines for the assessment of photovoltaic plants. document b. analysis and presentation of monitoring data. report eur 16339 en. Technical report, Joint Research Centre, European Commission. Institute for Systems Engineering and Informatics, 1995.

[Bec99] Kent Beck. *Extreme Programming Explained. Embrance Change.* Addison-Wesley Pearson Education, 1999.

[BFSS09] A. Benameur, S. Fenet, A. Saidane, and S.K. Sinha. A pattern-based general security framework an ebusiness case study. In *HPCC: 2009 11TH IEEE International Conference on High Performance Computing and Communications*, pages 339–346, 2009. 11th IEEE International Conference on High Performance Computing and Communications, Seoul, SOUTH KOREA, JUN 25-27, 2009.

[BFTH10] F. Balagtas-Fernandez, M. Tafelmayer, and H. Hussmann. Mobia modeler: Easing the creation process of mobile applications for non-technical users. In *IUI 2010*, pages 269–272, 2010. Proceedings of the 14th ACM International Conference on Intelligent User Interfaces, Hong Kong, PEOPLES R CHINA, FEB 07-10, 2010.

[BJ76] G.E.P. Box and G.M. Jenkins. *Time Series Analysis forecasting and control*. Prentice Hall, 1976.

[BM98] M. Benghanem and A. Maafi. Data acquisition system for photovoltaic systems performance monitoring. *IEEE Transactions on Instrumentation and Measurement*, 47 (1), 1998.

[BMN09] P. Bacher, H. Madsen, and H.A. Nielsen. Online short–term solar power forecasting. *Solar Energy*, 83(10):1772–1783, 2009.

[BMP13] M. Bouzerdoum, A. Mellit, and A. Massi Pavan. A hybrid model (sarima–svm) for short-term power forecasting of a small-scale grid-connected photovoltaic plant. *Solar Energy*, 98, Part C(0):226 – 235, 2013.

[Bou04] M. Boulle. Khiops: A statistical discretization method of continuous attributes. *Machine Learning*, 55:53–69, 2004.

[BPE+98] P. Benjamin, M. Painter, M. Erraguntla, C. Marshall, and R. Mayer. A framework for adaptive process modeling and execution (fame). In *Seventh IEEE International Workshops on Enabling Technlogies: Infrastructure for Collaborative Enterprises (WET ICE '98)*, pages 3–9, 1998. 7th IEEE International Workshops on Enabling Technologies - Infrastructure for Collaborative Enterprises (WET ICE 98), STANFORD, CA, JUN 17-19, 1998.

[BTM15] R.J. Bessa, A. Trindade, and V. Miranda. Spatial-temporal solar power forecasting for smart grids. *IEEE Transactions on Industrial Informatics*, 11(1):232–241, 2015.

[CCM+13] C.Voyant, C.Paoli, M.Muselli, , and M.Nivet. Multihorizon solar radiation forecasting for mediterranean locations using time series models. *Renewable and Sustainable Energy Reviews*, 28(0):44–52, 2013.

[CDCL11] C. Chen, S. Duan, T. Cai, and B. Liu. Online 24-h solar power forecasting based on weather type classification using artificial neural network. *Solar Energy*, 85(11):2856–2870, 2011. cited By 0.

[DK95] J. Dougherty, R. Kohavi, and M. Sahami. . Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelf International Conference on Machine Learning. Los Altos, CA.*, pages 194–202. Morgan Kaufmann, 1995.

[DO03] R. Dawson and B. O'Neil. Simple metrics for improving software process performance and capability: A case study. *Software quality control journal*, 11 (3):243–258, 2003.

[DR99] V. Devedzic and D. Radovic. A framework for building intelligent manufacturing systems. *IEEE Transactions on Systems Man and Cibernetics Part C-Applications and Reviews*, 29(3):422–439, AUG 1999.

[DSFOO+14] J.G. Da Silva Fonseca, T. Oozeki, H. Ohtake, K.-I. Shimose, T. Takashima, and K. Ogimoto. Forecasting regional photovoltaic power generation - a comparison of strategies to obtain one-day-ahead data. volume 57, pages 1337–1345, 2014. cited By 0.

[FCTB10] D.P. Filev, R.B. Chinnam., F. Tseng, and P. Baruah. An industrial strength novelty detection framework for autonomous equipment monitoring and diagnostics. *IEEE Transactions on Industrial Informatics*, 6(4):767 –779, 2010.

[FOT+11] J.G.S. Fonseca, T. Oozeki, T. Takashima, G. Koshimizu, Y. Uchida, and K. Ogimoto. Forecast of power production of a photovoltaic power plant in japan with multilayer perceptron artificial neural networks and support vector machines. In *26th European Photovoltaic Solar Energy Conference and Exhibition, Hamburg (Germany)*, pages 4237–4240, 2011.

[FOT+12] J.G.S. Fonseca, T. Oozeki, T. Takashima, G. Koshimizu, Y. Uchida, and K. Ogimoto. Use of support vector regression and numerically predicted cloudiness to forecast power output of a photovoltaic power plant in kitakyushu, japan. *Progress in Photovoltaics: Research and Applications*, in press:1–9, 2012.

[Fow99] Martin Fowler. *Refactoring: Improving the design of existing code.* Addison-Wesley, 1999.

[FS97] M.E. Fayad and D.C. Schmidt. Object-oriented applications frameworks. *Communications of the ACM*, 40 (10):32–38, 1997.

[FSH01] K. Feldmann, T. Stockel, and B. Haberstumpf. Conception and implementation of an object request broker for the integration of the process level in manufacturing systems. *Journal of.Systems Integration*, 10 (2):169–180, 2001.

[GCTL08] Huan Guo, Guohua Chen, Yong Tang, and Lin Li. Intelligent solar energy monitoring system under pervasive computing environment. pages 1557–1564, 2008.

[GH05] Jan G. De Gooijer and Rob J Hyndman. 25 years of iif time series forecasting: A selective review. Monash Econometrics and Business Statistics Working Papers 12/05, Monash University, Department of Econometrics and Business Statistics, 2005.

[GH09] V.C. Gungor and G.P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Transactions on Industrial Electronics*, 56(10):4258 –4265, 2009.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison-Wesley Publishing Company, 1995.

[GL03] J.J. Guo and P.B. Luh. Selecting input factors for clusters of gaussian radial basis function networks to improve market clearing price prediction. *IEE Transactions on Power Systems*, 18 (2):665–672, 2003.

[GLP07] M. Gagliarducci, D.A. Lampasi, and L. Podesta. Gsm-based monitoring and control of photovoltaic power generation. *Measurement*, 40:314–321, 2007.

[GNAB09] A. Ghanbarzadeh, A.R. Noghrehabadi, E. Assareh, and M.A. BehrangIn. Solar radiation forecasting based on meteorological data using artificial neural networks. In *7th IEEE International Conference on Industrial Informatics, volume 1–2*, pages 227–231, 2009.

[God97] G. Godena. Conceptual model for process control software specification. *Microprocessors and Microsystems*, 20 (10):617–630, 1997.

[GOGM13] Javier Gamez Garcia, Juan Gomez Ortega, Alejandro Sanchez Garcia, and Silvia Satorres Martinez. Robotic software architecture for multisensor fusion system. *IEEE Transactions on Industrial Electronics*, 56(3):766 – 777, 2013.

[Gor01] Alan Gordon. *Programación COM y COM+.* Anaya Multimedia, 2001.

[GPC06] R.A. Guarnieri, E.B. Pereira, and S.C. Chou. Solar radiation forecast using artificial neural networks in south brazil. In *Proc. 8 ICSHMO*, pages 1777–1785, 2006.

[Gri91] William G. Griswold. *Program Restructuring as an Aid to Software Maintenance.* PhD thesis, University of Washington, 1991.

[GTCS09] H. Guozhen, C. Tao, C. Changsong, and D. Shanxu. Solutions for scada system communication reliability in photovoltaic power plants. In *IEEE 6th International Power Electronics and Motion Control Conference*, pages 2482–2485, 2009.

[Har01] Eric Harmon. *Programación COM en Delphi.* Danypress, 2001.

[HCL98] J. Hwang, S.M. Chen, and C.H. Lee. Handling forecasting problems using fuzzy time series. *Fuzzy sets and Systems*, 100:217–228, 1998.

[Hol04] D.W. Holley. Understanding and using opc maintenance and reliability applications. *Computing Control Engineering Journal*, 15(1):28 – 31, feb.-march 2004.

[Iqb84] M. Iqbal. *An introduction to solar radiation.* Academic Press Inc. New York London, 1984.

[JBA87] C.M. Jarque, A. Bera, and K. Anil. A test for normality of observations and regression residuals. *International Statistical Review*, 55(2):163–172, 1987.

[JF88] R.E. Johnson and B. Foote. Designing reusable classes. *Journal of Object Oriented Programming*, 1 (2):22–35, 1988.

[JZA+06] X. Jiao, G. Zheng, P. A. Alexander, M.T. Campbell, O.S. Lawlor, J. Norris, A. Haselbacher, and M.T. Heath. A system integration framework for coupled multiphysics simulations. *Engineering with Computers*, 22(3-4):293–309, 2006.

[Ker04] Joshua Kerievsky. *Refactoring to Patterns*. Addison-Wesley, 2004.

[KKV03] K. Kalaitzakis, E. Koutroulis, and V. Vlachos. Development of a data acquisition system for remote monitoring of renewable energy systems. *Measurement*, 34:75–83, 2003.

[KOV+11] Ahmet Koca, Hakan F. Oztop, Yasin Varol, , and Gonca Ozmen Koca. Estimation of solar radiation using artificial neural networks with different input parameters for mediterranean region of anatolia in turkey. *Expert Systems with Applications*, 38(7):8756–8762, 2011.

[KU05] R Kumar and L. Umanand. Estimation of global radiation using clearness index model for sizing photovoltaic system. *Renewable energy*, 30 (15), 2005.

[LH03] A. Luque and S. Hegedus. *Handbook of Photovoltaic Science and Engineering*. John Wiley and Sons Ltd. England, 2003.

[LHHB09] E. Lorenz, J. Hurka, D. Heinemann, and H.G. Beyer. Irradiance forecasting for the power prediction of grid-connected photovoltaic systems. *IEEE Journal of Special Topics in Earth Observations and Remote Sensing*, 2:2–10, 2009.

[LHTD02] H. Lui, F. Hussain, C. Lim Tan, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6:393–423, 2002.

[LLH+05] J. Liu, K.W. Lim, W.K. Ho, K.C. Tan, A. Tay, and R. Srinivasan. Using the opc standard for real-time process monitoring and control. *IEEE Software*, 22 (6):54–59, 2005.

[LSH+11] E. Lorenz, T. Scheidsteger, J. Hurka, D. Heinemann, and C. Kurz. Regional pv power prediction for improved grid integration. *Progress in Photovoltaics: Research and Applications*, 19(7):757–771, 2011. cited By 0.

[LTC01] R.C. Luo, J.H. Tzou, and Y.C. Chang. Desktop rapid prototyping system with supervisory control and monitoring through internet. *IEEE/ASME Transactions on Mechatronics*, 6(4):399 –409, December 2001.

[Mey99] Bertrand Meyer. *Construcción de software orientado a objetos*. Prentice Hall, 1999.

[MLMdCMB05] L. Mora-López, J. Mora, M. Sidrach de Cardona, and R. Morales-Bueno. Modelling time series of climatic parameters with probabilistic finite automata. *Environmental modelling and software*, 20 (6):753–760, 2005.

[MLMPdC10] L. Mora-López, J. Mora, M. Piliougine, and M. Sidrach de Cardona. An intelligent memory model for short-term prediction: An application to global solar radiation data. *Lecture Notes in Artificial Intelligence*, 6098:596–605, 2010.

[MLRMBR00] L. Mora-López, I. Fortes Ruis, R. Morales-Bueno, and F. Triguero Ruiz. Dynamic discretizacion of continuous values from time series. *Lecture Notes in Artificial Intelligence*, 1810:280–291, 2000.

[MP10] Adel Mellit and Alessandro Massi Pavan. A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid connected pv plant at trieste, italy. *Solar Energy*, 84(5):807–821, 2010.

[MPSD10] P. Mariño, F. Poza, S.Otero, and M.A. Domínguez. Reconfigurable industrial sensors for remote condition monitoring and modeling. *IEEE Transactions on Industrial Electronics*, 57(12):4199 –4208, 2010.

[MT07] J.R. Moyne and D.M. Tilbury. The emergence of industrial control networks for manufacturing control, diagnostics, and safety data. *Proceedings of the IEEE*, 95(1):29 –47, jan. 2007.

[NTI+10] Y. Nakada, H. Takahashi, K. Ichida, T. Minemoto, and H. Takakura. Influence of clearness index and air mass on sunlight and outdoor performance of photovoltaic modules. *Current Applied Physics*, 10 (2,1), 2010.

[Ost86] C.R. Osterwald. Translation of device performance measurements to reference conditions. *Solar Cells*, 19:269–279, 1986.

[Ple94] P. Pleinevaux. An analysis of the mms object model. *IEEE Transactions on Industrial Electronics*, 41(3):265 –268, June 1994.

[PMS07] R. Perez, K. Moore, and P. Stackhouse. Forecasting solar radiation. preliminary evaluation of an approach based upon the national forecast database. *Solar Energy*, 81 (6):809–812, 2007.

[PT98] P. Perner and S. Trautzsch. Multi-interval discretization methods for decision tree learning. In *SSPR/SPR*, pages 475–482, 1998.

[Rei09] G. Reikard. Predicting solar radiation at high resolutions: A comparison of time series forecast. *Solar Energy*, 83:342–349, 2009.

[RST94] D. Ron, Y. Singer, and N. Tishby. Learning probabilistic automata with variable memory length. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, 1994.

[SAL+] Thomas Strasser, Filip Andren, Felix Lehfuss, Matthias Stifter, and Peter Palensky. Online reconfigurable control software for ieds. *IEEE Transactions on Industrial Informatics*.

[SC93] Q. Song and B.S. Chisson. Forecasting enrollments with fuzzy time series. part i. *Fuzzy Sets and Systems*, 54:269–277, 1993.

[SC94] Q. Song and B.S. Chisson. Forecasting enrollments with fuzzy time series. part ii. *Fuzzy Sets and Systems*, 62:1–8, 1994.

[Sch97] D.C. Schmidt. Lessons learned building reusable oo frameworks for distributed software. *Communications of the ACM*, 40(10):85–87, 1997.

[SCST12] Y. Su, L.-C. Chan, L. Shu, and K.-L. Tsui. Real-time prediction models for output power and efficiency of grid-connected solar photovoltaic systems. *Applied Energy*, 93:319–326, 2012. cited By 0.

[SLM98] D.C. Schmidt, D.L. Levine, and S. Mungee. The design of the tao real-time object request broker. *Computer Communications*, 21 (4):294–324, 1998.

[SPG+08] S.Chowdhury, P.Day, G.A.Taylor, S.P.Chowdhury, T.Markvart, and Y.H.Song. Supervisory data acquisition and performance analysis of a pv array installation with data logger. In *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century IEEE*, pages 1–8, 2008.

[SRIS09] S.I. Sulaiman, T.K.A. Rahman, I.Musirin, and S. Shaari. Performance analysis of evolutionary ann for output prediction of a grid-connected photovoltaic system. In *World Academy of Science, Engineering and Technology, volume 53*, 2009.

[SWKL97] A.P. Stublen, C.M. Wellman, S.A. Kell, and T.W. Langston. Justifying and planning an energy monitoring system. *IEEE Industry Applications Magazine*, pages 54–61, 1997.

[Szy98] C. Szyperski. *Component Software*. Addison-Wesley, Reading, MA, 1998.

[TCD05] A. Talevski, E. Chang, and T.S. Dillon. Reconfigurable web service integration in the extended logistics enterprise. *IEEE Transactions on Industrial Informatics*, 1(2):74 – 84, May 2005.

[TVK09] S. Theiss, V. Vasyutynskyy, and K. Kabitzsch. Software agents in industry: A customized framework in theory and praxis. *IEEE Transactions on Industrial Informatics*, 5(2):147 –156, May 2009.

[VAA+09] S. Vergura, G. Acciani, V. Amoruso, G.E. Patrono, , and F. Vacca. Descriptive and inferential statistics for supervising and monitoring

the operation of pv plants. *IEEE Transactions on Industrial Electronics*, 56(11):4456–4464, 2009.

[VAR08] VARIOS. *Modeling Solar Radiation at the Earth's Surface. Recent Advances.* Springer., 2008.

[WH08] C.H. Wang and L.C. Hsu. Constructing and applying an improved fuzzy time series model: Taking the tourism industry for example. *Expert Systems with Applications*, 34:2732–2738, 2008.

[WKS+01] L. Wills, S. Kannan, S. Sander, M. Guler, B. Heck, J.V.R. Prasad, D. Schrage, and G. Vachtsevanos. An open platform for reconfigurable control. *IEEE Control Systems*, 21 (3):49–64, 2001.

[WL07] L. Wang and K.H. Liu. Implementation of a web-based real-time monitoring and control system for a hybrid wind-pv-battery renewable energy system. In *International Conference on Intelligent Systems Applications to Power Systems*, pages 1–6, 2007.

[WZM+15] F. Wang, Z. Zhen, Z. Mi, H. Sun, S. Su, and G. Yang. Solar irradiance feature extraction and support vector machines based weather status pattern recognition model for short-term photovoltaic power forecasting. *Energy and Buildings*, 86:427–438, 2015. cited By 0.

[XDT04] L. Xu, Z. Y. Dong, and A. Tay. *Recent Advances in Simulated Evolution and Learning*, chapter Chapter 40: Time Series Forecast with Elman Neural Networks and Genetic Algorithms, pages 747–768. World Scientific Books, 2004.

[YCW+04] J.M. Yang, K.W.E. Cheng, J. Wu, P. Dong, and B. Wang. The study of the energy management system based-on fuzzy control for distributed hybrid wind-solar power system. In *Power Electronics Systems and Applications, 2004. Proceedings. 2004 First International Conference on*, pages 113–117, 2004.

[YHHP14] Hong-Tzer Yang, Chao-Ming Huang, Yann-Chan Huang, and Yi-Shiang Pai. A weather-based hybrid method for 1-day ahead hourly forecasting of pv power output. *IEEE Transactions on Sustainable Energy*, 4(3):917 – 926, 2014.

[ZMAP14a] M. Zamo, O. Mestre, P. Arbogast, and O. Pannekoucke. A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production, part i: Deterministic forecast of hourly production. *Solar Energy*, 105(0):792 – 803, 2014.

[ZMAP14b] M. Zamo, O. Mestre, P. Arbogast, and O. Pannekoucke. A benchmark of statistical regression methods for short-term forecasting of photovoltaic electricity production. part ii: Probabilistic forecast of daily production. *Solar Energy*, 105(0):804 – 816, 2014.

[ZPLZ09] Jun-Hong Zhou, Chee Khiang Pang, F.L. Lewis, and Zhao-Wei Zhong. Intelligent diagnosis and prognosis of tool wear using dominant feature identification. *IEEE Transactions on Industrial Informatics*, 5(4):454 –464, 2009.

[ZQ05] G. P. Zhang and M. Qi. Neural network forecasting for seasonal and trend time series: Is data preprocessing necessary. *European Journal of Operational Research*, 160:501–514, 2005.

Publicaciones

- Martínez-Marchena, Ildefonso; Sidrach-de-Cardona, Mariano; Mora-López, Llanos. (2014) **Framework for monitoring and assessing small and medium solar energy plants**. Journal of Solar Energy Engineering-Transactions of the ASME. Vol 137(2), 021007 (Sep 30, 2014).
Índice de impacto: 1.132. DOI: 10.1115/1.4028398.

- Mora-Lopez, Llanos; Martínez-Marchena, Ildefonso; Piliougine-Rocha, Michel; Sidrach De Cardona-Ortin, Mariano. (2011). **Binding statistical and machine learning models for short-term forecasting of global solar radiation**. Lecture notes in computer science, 7014, 294-305.

- Mora-Lopez, Llanos; Martínez-Marchena, Ildefonso; Sánchez-Illescas, Pedro Jesús; Sidrach De Cardona-Ortin, Mariano. (2010). **Binding machine learning models and opc technology for evaluating solar energy systems**. Lecture notes in computer science, 6098, 606-615.

- Sidrach de Cardona, Mariano; Carretero Rubio, Jesús; Pereña, Agustín; Mora López, Llanos; Martínez Marchena, Ildefonso. (2005). **Monitorización wireless de una instalación fotovoltaica de 56 kWp**. Era solar, pp. 56-64. 1/7/2005. ISSN 0212-4157.

- Jiménez-Pérez, Pedro Francisco; Martínez-Marchena, Ildefonso; Navarro-tapia, Diego; Piliougine-Rocha, Michel; Mora-Lopez, Llanos; Sidrach De Cardona-Ortin, Mariano. **Experiencias en la monitorización de sistemas fotovoltaicos conectados a red**. XV Congreso Ibérico y X Congreso Iberoamericano de Energía Solar, VIGO, ESPAÑA, 2012.

- Mora-Lopez, Llanos; Martinez-Marchena,Ildefonso; Piliougine-Rocha, Michel; Sidrach De Cardona-Ortin, Mariano. **Machine learning approach for next day energy production forecasting in grid connected photovoltaic plants**. World Renewable Energy Congress (WREC), LINKOPING, SWEDEN, 2011.

- Martínez-Marchena, Ildefonso; Mora-Lopez, Llanos; Piliougine-Rocha, Michel; Sidrach De Cardona-Ortin, Mariano. **An integrated software for monitoring and evaluation solar photovoltaic installations**. 25TH European Photovoltaic Solar Energy Conference and Exhibition. 5TH World Conference on Photovoltaic Energy Conversion, VALENCIA, ESPAÑA, 2010. ISBN: 3-936338-26-4. DOI: 10.4229/25thEUPVSEC2010-4BV.1.99

- Martinez-Marchena,Ildefonso; Mora-Lopez, Llanos; Piliougine-Rocha, Michel; Sidrach De Cardona-Ortin, Mariano. **Software para la monitorización y evaluación de instalaciones fotovoltaicas**. IV Conferencia Latino Americana de Energía Solar (IV ISES-CLA) Y XVII Simposio Peruano de Energía Solar (XVII-SPES), CUZCO (PERÚ), 2010.

- Sidrach De Cardona-Ortin, Mariano; Mora-Lopez, Llanos; Piliougine-Rocha, Michel; Martinez-Marchena,Ildefonso. **OPC-based software for monitoring photovoltaic power plants**. 23RD European Photovoltaic Solar Energy Conference, VALENCIA, SPAIN, 2008. ISBN: 3-936338-24-8.
DOI: 10.4229/23rdEUPVSEC2008-5BV.2.34

- Mora-Lopez, Llanos; Sidrach De Cardona-Ortin, Mariano; Martínez-Marchena, Ildefonso. **A framework for the development of monitoring systems software**. First International Conference on Software and Data Technologies , SETÚBAL, PORTUGAL, 2006.

- Martínez-Marchena, Ildefonso; Pereña,Agustín; Mora-Lopez, Llanos; Sidrach De Cardona-Ortin, Mariano; Carretero-Rubio, Jesús Eduardo. **Real time wireless monitoring of a photovoltaic system using OPC technology**. International Congress of Energy and Environment Engineering and Management, PORTALEGRE, PORTUGAL, 2005.

- Mora-Lopez, Llanos; Sidrach De Cardona-Ortin, Mariano; Martínez-Marchena, Ildefonso. **OPC technology for the wireless monitoring of a remote energy installation**. International Conference on Automation, Control and Instrumentation (IADAT), BILBAO, 2005.

- Mora-Lopez, Llanos; Sidrach De Cardona-Ortin, Mariano; Martínez-Marchena, Ildefonso. **Generador de sistemas de monitorización y control**. III Conferencia Internacional de Control de Sistemas Industriales, CUBA, 2000.