Capítulo 5: Materiales y métodos

5.1 Introducción

Con el objetivo de definir unas características tecnológicas para cada uno de los distintos módulos comentados en apartados anteriores, y teniendo en cuenta las diferentes alternativas presentes en el mercado y las distintas especificaciones descritas por los organismos de estandarización pertinentes, en este capítulo se realiza una selección de tecnologías, herramientas software, lenguajes de programación, y en resumen, plataformas implicadas en el desarrollo e implementación del sistema propuesto.

Esta selección de tecnologías no está basada en criterios puramente arbitrarios, sino que se fundamenta en un estudio comparativo a partir de los requerimientos exigidos por la arquitectura, y la enorme casuística que estos mismos entrañan. Asimismo, se han considerado como indispensables las ventajas y libertad proporcionadas por las soluciones de software libre a la hora de realizar la discriminación de tecnologías software, así como el uso de estándares abiertos que ya se encuentran ampliamente extendidos. Con esto se permite alcanzar una mayor interoperatividad entre elementos heterogéneos, y una capacidad de evolución no dependiente de terceros.

Antes de realizar una descripción tecnológica más detallada de cada uno de los módulos del sistema, se realiza una selección de componentes comunes tanto al BGC como al BAS, y que son sustento de toda la arquitectura. Más concretamente nos referimos al sistema operativo, al lenguaje de programación empleado, y a la norma sanitaria implementada en el modelo de información utilizado.

- Sistema operativo: debido a la gran variabilidad de dispositivos que pueden estar involucrados en el sistema, a sus características, y a la gran disparidad de arquitecturas comprometidas, se ha optado por la distribución libre Debian [65] del sistema operativo GNU/Linux en su versión estable Lenny.
  Debian ofrece razones de peso importantes que nos han llevado a tomar esta decisión, entre las que destacan, su ya dilatada trayectoria como distribución por antonomasia dentro del mundo GNU/Linux, su extenso número de paquetes liberados, su gran soporte a todo tipo de arquitecturas de computación, y su ya demostrada robustez y sólida seguridad para todo tipo de aplicaciones de propósito general.
  Esta elección se torna como recomendada, pero no se excluye otro tipo alternativas como los sistemas operativos Microsoft u otro tipo de sistemas UNIX. En general, para esta elección del sistema operativo deben considerarse los siguientes aspectos:
    - Debe soportar interfaces normalizados para el acceso a base de datos u ODBC (Conectividad Abierta con Bases de Datos)
    - Debe ser compatible con el MOM y con la ORDB utilizada.
    - Debe ofrecer características de robustez, seguridad y estabilidad.
    - Debe soportar la familia de estándares POSIX

  Los sistemas operativos embebidos o portables para dispositivos con hardware limitado también deben considerarse, sobre todo para proveer una interfaz (gráfico o no) acorde con estas restricciones, que le permita la producción o el consumo de información dentro del entorno en el que trabaja.
- Lenguaje de programación: a pesar de la disparidad de módulos heterogéneos que constituyen el sistema, que pueden estar construidos en lenguajes de programación de muy distinta índole, se considera al lenguaje de alto nivel C como sostén para la implementación de toda la arquitectura, a expensas de otros lenguajes de propósito general, como por

ejemplo Java, que por su naturaleza posee un rendimiento inferior a C [66], y su funcionamiento depende de la instalación de aplicaciones de terceros.

- Norma sanitaria: basándose en el enfoque dual que distingue la información a través un modelo de referencia, y por otro lado del conocimiento mediante un modelo de arquetipos, se ha optado por la norma del CEN 13606, que actualmente se encuentra en un estado suficientemente maduro y que cuenta con unos 10 años de desarrollo. Su flexibilidad, resistencia a cambios, y su posibilidad de evolución han sido vitales para su elección.
  La elección de esta norma no implica que puedan utilizarse otras, ya sea de forma conjunta o independientemente, mediante la inserción de plataformas de conversión entre normas sanitarias que trabajarían dentro de todos los módulos del CGB como última instancia antes de la publicación o suscripción al MOM. Actualmente existen ontologías que son utilizadas para la integración e interoperatividad entre sistemas de información para comunidades sanitarias combinando datos de HLE y terminologías tales como UMLS, MEDCIN y SNOMED.

5.1 Bloque de Gestión del Conocimiento (BGC)

5.1.1 Módulo de monitorización

Como se comentó en el apartado de la descripción de este módulo, el estándar de comunicación de dispositivos médicos utilizados en la monitorización de datos es irrelevante en la toma de datos. Sin embargo, el archivo de configuración proporcionado por el dispositivo debe estar correctamente construido para transformar los variables monitorizadas a un metalenguaje estandarizado. En este caso, el metalenguaje empleado es XML, que juega un papel importante para el intercambio de una gran variedad de datos de muy diversa índole. Su formato permite la lectura de datos y el intercambio de información entre aplicaciones que se ejecutan en plataformas heterogéneas.

Por tanto, es necesario definir qué elementos van a contener el documento XML, cómo van a estar organizados, sus atributos, y las relaciones existentes entre los mismos y los datos monitorizados por los dispositivos médicos. Para ello se emplea el lenguaje de esquema XML Schema o XSD, que describe la estructura y las restricciones de los contenidos de los archivos XML, más allá de las normas sintácticas que ya impone XML.

Con el objetivo de formalizar este archivo XML construido, a uno que siga el modelo de referencia descrito en la norma CEN/ISO 13606, se hace uso de la tecnología XSLT (Extensible Stylesheet Language Transformations), también denominado transformaciones XSL, que permite efectuar una conversión de documentos XML en otros, e incluso en formatos que no son XML.

Una vez establecido el extracto de HCE conforme al estándar sanitario, es imprescindible la adaptación del archivo XML al modelo de datos utilizado en el MOM. Esta fase puede realizarse de dos formas:

- Trasvase “en bruto”: el documento XML construido se publica en el MOM como un archivo indivisible. Para el caso del middleware DDS, se utilizará un Topic que defina un dato en lenguaje IDL de tipo *sequence unbounded* de octetos para el envío de *raw data* o datos “crudos”, es decir, aquellos que no han sido procesados. Este tipo de secuencias también se utiliza para el envío de archivos de audio, vídeo e imágenes.
  En este caso, el análisis del documento XML publicado debe realizarlo el suscriptor, siendo él el encargado de discriminar la información necesaria, de la prescindible. Se puede utilizar

las tecnologías DOM (Modelo de Objetos del Documento) o SAX (Simple API para XML) para el análisis de archivos XML.
Si se utiliza este tipo de trasvase, debe garantizarse la entrega en orden confiable de todas las muestras de información, para que el suscriptor pueda ensamblar todo el archivo en el otro extremo. Para ello intervienen las políticas de calidad de servicio *DestinationOrderQoSpolicy* (parámetro SOURCE_TIMESTAMP_DESTINATION) y la política *ReliabilityQoSPolicy* (parámetro RELIABLE_RELIABILITY_QOS), ambas descritas en la especificación DDS.
En el caso de la primera política, cada muestra publicada lleva un *timestamp* asociado que es establecido por el DataWriter en el momento del envío. De esta forma el DataReader del suscriptor podrá unir todas las muestras a partir de este tiempo especificado en el origen. Para la segunda política, el servicio DDS entregará todas las muestras y deberá recibir un asentimiento por parte del suscriptor. Si esto no sucede, se volverá a enviar la muestra.
Tanto esta tarea como la anterior, es llevada a cabo de forma automática por DDS y es transparente para el desarrollador.

- Trasvase normal: para este tipo de trasvase, la discriminación de la información prescindible de la que no lo es se realiza por el publicador antes de la transmisión. Con esto se consigue evitar la publicación de datos inservible y acercarse al concepto de personalización médica. Es por ello fundamental definir una metodología de mapeo entre los tipos de datos empleados en el MOM, con los utilizados en el modelo de referencia y de arquetipos de la norma sanitaria.

5.1.2 Módulo de almacenamiento persistente

El componente fundamental dentro de este módulo es la base de datos objeto-relacional, elemento constituyente de los nodos del clúster. Siguiendo la filosofía Open-Source comentada en apartados anteriores, se ha elegido por PostgreSQL en detrimento de MySQL, que se trata de la otra base de datos objeto-relacional de software libre que lidera el mercado. Existen muchos estudios que comparan las características y el rendimiento de ambas, y se ha optado por PostgreSQL debido a sus ventajas en cuanto a integridad referencial, característica que en el caso de MySQL se ignora y se deja al programador de la aplicación [52]. En nuestro caso, la consistencia de la información es vital, y aunque esto afecte negativamente a la velocidad de acceso a los datos, PostgreSQL ha demostrado ser superior en este sentido. A pesar de esto, esta caída de rendimiento es paliada por las características de almacenamiento volátil proporcionadas por el MOM, por lo que de esta forma se garantiza el acceso rápido a la información y se dota al conjunto de la capacidad deseable de trabajar en tiempo real.

La replicación dentro de un nodo debe realizarse de forma periódica, transparente, incremental, y de manera que no afecte al rendimiento en el acceso a la base de datos maestra dentro del nodo. La periodicidad de la replicación va a depender de la importancia de la información y su involucración en la generación de conocimiento. Por ejemplo, una inclusión de un análisis médico dentro de la historia clínica de un paciente no necesita una actualización imperiosa, y la demora de un plazo de tiempo concreto puede ser tolerable. Sin embargo, esto no debería suceder cuando se están monitorizando variables biomédicas a un paciente y se necesita soporte de almacenamiento no volátil. Por tanto, a partir de lo demandado, deben proveerse mecanismos que permitan agrupar diferentes bases de datos y definir unos requisitos de periodicidad que establezcan los tiempos de ejecución del proceso de replicación en un nodo. Esta replicación será llevada a cabo por la herramienta *pg_dumpall* suministrada por PostgreSQL, que permite extraer todas las tablas de la base de datos maestra, almacenarlas en un script, y transportarlas a todas las bases de datos esclavas que compongan el nodo, y de esta forma proporcionar un medio de almacenamiento de

copia de seguridad.

Por otra parte, para el entramado de nodos y sus conexiones, se utiliza un clúster para PostgreSQL, destacando *pg-pool-II* como herramienta más completa para la realización de esta labor. Ofrece capacidades de replicación, balanceo de carga y un pool de conexiones, posibilitando la recuperación de nodos caídos en línea (sin dejar de dar servicio). Se trata de un clúster activo-pasivo, si bien se hace uso del modo pasivo para la lectura con el propósito de mejorar la productividad del sistema.

Estas características proporcionadas por *pg-pool-II*, como son el sistema de balanceo de carga que incluye, puede sustituir al comentado en la descripción del sistema (el que se encuentra por delante de los servidores). En caso de que no lo haga, deben establecerse un número determinado de servidores que se encuentran controlados por medio de un balanceador de carga, es decir, un dispositivo hardware o software que atiende a las aplicaciones externas y asigna las solicitudes que le llegan utilizando un determinado algoritmo. Un ejemplo sencillo que suele utilizarse es el de pesos, que permite lanzar la petición entrante al servidor que se encuentra con una menor carga de trabajo). Dentro de los balanceadores de carga existentes en el mercado, LVS [53] (Servidor Virtual de Linux) se presenta como una alternativa a tener en cuenta gracias a su gran escalabilidad, confiabilidad y robustez.

5.1.3 Módulo de procesamiento

Está involucrado en la generación de nuevo conocimiento a partir de información o conocimiento ya existente. La participación de los modelos multi-escala matemáticos destacan en esta generación.

Con respecto al lenguaje de programación para la construcción de modelos fisiológicos, se ha optado por el software matemático Matlab, que ofrece un entorno de desarrollo integrado y un lenguaje propio. La comunicación con DDS para la toma y suministro de los datos por parte de un modelo matemático construido en Matlab no es un proceso inmediato, ya que ninguna implementación DDS a día de hoy provee soporte para el lenguaje M (empleado por Matlab).

Para solventar este contratiempo, es necesario el uso de una base de datos intermediaria que permita la comunicación entre Matlab y DDS. Para la conexión de Matlab con la base de datos puede utilizarse *TheDatabaseToolbox* [54] mediante ODBC/JDBC (Conectividad Abierta con Bases de Datos/Conectividad Base de Datos Con Java). La base de datos bróker será MySQL con tablas de tipo MEMORY, que tienen la particularidad de que trabajan en memoria principal y por tanto son mucho más eficientes para trabajar con modelos matemáticos con fuertes restricciones temporales. Debe de tenerse en cuenta que este almacenamiento será temporal, por lo que el espacio reservado será bajo, y el consumo de recursos será mínimo. Debe configurarse apropiadamente la base de datos para adaptarla a estas necesidades.

5.1.4 Módulo de creación de conocimiento

Para la definición de arquetipos, la norma CEN/TS 13606 hace uso del lenguaje de descripción de arquetipos ADL (Lenguaje de Definición de Arquetipos), que está diseñado especialmente para establecer restricciones sobre un determinado modelo de referencia, de tal forma que se indique qué tipo de información puede contener el arquetipo y cuál no. ADL se compone a su vez de dos sintaxis bien diferencias: dADL (Lenguaje de Definición de Datos), y cADL (Lenguaje

de definición de restricciones), así como el proceso de mapeo del propio lenguaje.

El elemento de intercambio entre nodos dentro del sistema es el extracto, que puede contener cualquier tipo de información clínica, y que se genera basándose en los diferentes arquetipos definidos por los expertos en el dominio sanitario. Estos arquetipos deben estar disponibles en un repositorio destinado para tal fin dentro de un determinado canal u evento del MOM, de tal manera que cualquier usuario pueda acceder a ellos mediante una suscripción. Estos arquetipos deben estar clasificados en categorías siguiendo una ontología determinada, para que la suscripción a este repositorio pueda realizarse de manera personalizada a través de la filtración de contenidos. Una vez seleccionado un arquetipo en concreto, se obtiene la plantilla asociada que está almacenada en la base de datos.

Dentro de la estructura que especifica un arquetipo, existe una sección de *definición* donde se describe el concepto clínico que se quiere representar. Esta descripción se construye estructurando y restringiendo las entidades del modelo de referencia. También se dispone de una sección de *ontología*, en donde se realiza una descripción de las entidades que aparecen en la sección de *definición*, y donde se asocian con terminologías médicas, dotándolas así de información semántica.

Actualmente existen herramientas para la creación, edición y mantenimiento de arquetipos [49] pero ninguna soporta el concepto de arquetipos de integración. Es por ello que se opta por la herramienta LinkEHR-Ed [55], que se trata de un editor que facilita esta tarea.

LinkEHR-Ed es una herramienta que pone en común las necesidades de los profesionales sanitarios que desarrollan la definición de los arquetipos y la de los expertos en tecnologías de la información que realizan los mapeos con las fuentes de datos. Ambos son los encargados de desarrollar arquetipos para la integración de información clínica de manera colaborativa. LinkEHR-Ed es una parte fundamental de LinkEHR, un sistema en desarrollo para la publicación y el acceso inteligente a la HCE de los pacientes basado en una arquitectura de modelo dual

Para garantizar que los arquetipos y extractos son correctos y se adecúan a las normas del lenguaje ADL y XML respectivamente, es necesario llevar a cabo un proceso de validación mediante alguna herramienta destinada a este propósito. La validación semántica será llevada a cabo por la aplicación ADL workbench [56], mientras que para la validación estructural del XML se empleará el lenguaje de esquema XML Schemas.
.

5.2 Bloque de Administración del Sistema (BAS)

Al término de este trabajo no se han definido las especificaciones software que sustentan este bloque, ya que al tratarse de un componente administrador del Bloque de Gestión del Conocimiento (CGB) depende en gran medida de él, y por tanto su descripción tecnológica debe establecerse una vez que se ha detallado más exhaustivamente la del CGB.

5.3 Middleware Orientado Mensajería (MOM)

Después del análisis pormenorizado que se realizó en el capítulo 3 acerca de los diferentes MOM existentes, tecnologías e implementaciones asociadas, se ha optado por la especificación DDS como MOM y OpenSplice como implementación DDS. Las razones de la elección de DDS frente a otras alternativas, son las siguientes:

- Presenta un conjunto de políticas de calidad de servicio suficientemente amplia y variada, permitiendo que la comunicación sea altamente parametrizable.
- Arquitectura *De-centralized unbrokered*, que elimina intermediarios de comunicación y cuellos de botella.
- Disminución del acoplamiento entre entidades gracias al paradigma publicador/suscriptor
- Independencia de plataforma o lenguaje de programación empleado.
- Filtrado de contenido mediante *FilteredContentTopic* y *MultiTopic*.
- Arquitectura simplificada respecto a otros middleware, flexible y adaptable gracias al descubrimiento de la información de forma automática y transparente al usuario.
- Soporte de tiempo real.
- Característica deseable de escalabilidad por medio del uso de particiones.
- Mayor predictibilidad en la entrega de la información y menor latencia debido a que los datos se almacenan en una cola a nivel local.
- Testeada con éxito en entornos militares y en la industria aeroespacial.
- Tecnología en constante desarrollo y con perspectivas futuras alentadoras.
- Conectividad con otras tecnologías análogas, como por ejemplo, otros middleware orientados a mensajería, o los orientados a objetos.
- Comunicación con otras implementaciones DDS diferentes gracias al protocolo RTPS.
- A pesar de ser una tecnología prácticamente nueva, la mayoría de las implementaciones DDS están construidas bajo una base de CORBA, lo que garantiza la robustez de este estándar, que se encuentra ampliamente extendido en sus más de 10 años de desarrollo.

En cuanto a la implementación DDS, OpenSplice presenta las siguientes ventajas respecto a sus competidores:

- Suministra una documentación extensa y completa, tanto a nivel de usuario como a nivel del desarrollador.
- Es unas de las soluciones software que implementa un mayor número de políticas de calidad de servicio.
- Está suficientemente testeado y probado en ámbitos de muy diversa índole.
- Provee un entorno de desarrollo potente para la construcción de aplicaciones DDS, diseño de diagramas de relaciones entre entidades, y facilita herramientas de configuración del Global Data Space, y para la monitorización de Topics en tiempo real.
- Provee la característica de persistencia de la información mediante archivos XML o bases de datos, si bien es cierto que sólo se ofrece en la versión de evaluación.
- Suministra APIs para C, C++, Java, C#, y es compatible con la mayoría de sistemas operativos actuales, así como un gran número de sistemas embebidos y portables.
- Se encuentra en constante evolución, y nuevas mejoras importantes se han introducido en versiones recientes.
- OpenSplice DDS se encuentra implementado por los creadores y fundadores de la propia especificación DDS.

A continuación se hace un breve repaso sobre cada uno de las implementaciones DDS que existen en la actualidad, finalizando con una comparativa.

5.3.1 Implementaciones de DDS

- **OpenSplice DDS**

OpenSpliceDDS [57] es una implementación DDS de Prismtech que proporciona una infraestructura para la distribución de datos en tiempo real, ofreciendo servicios de middleware a las aplicaciones, y caracterizándose por su alto rendimiento, escalabilidad y predictibilidad.

Ofrece soporte para todos los perfiles DCPS definidos en la especificación DDS y para el nivel DLRL. OpenSplice DDS fue desarrollado inicialmente por Thales Naval Netherlands (TNL), uno de los coautores de la especificación DDS, y es el resultado de 15 años de experiencia en el desarrollo de sistemas de información distribuidas en los Combat Management Systems (CMS).

Para garantizar escalabilidad, flexibilidad y extensibilidad, OpenSpliceDDS posee una arquitectura, ilustrada en la Figura 5.1, que se basa en la utilización de un segmento de memoria compartida que, además de permitir interconectar todas las aplicaciones del equipo, almacena un conjunto de servicios extensibles. Estos servicios proporcionan unas funcionalidades *plug and play* para la configuración de mecanismos de durabilidad (almacenamiento tolerante a fallos y persistencia de la información), servicios de red, control remoto y monitorización (por medio del protocolo SOAP y de la herramienta DDS Tuner Tools), entre otros servicios. La configuración de los mismos es llevada a cabo de forma sencilla a través de archivos XML o mediante la interfaz gráfica OSPLCONF, que genera dichos archivos de forma automática.

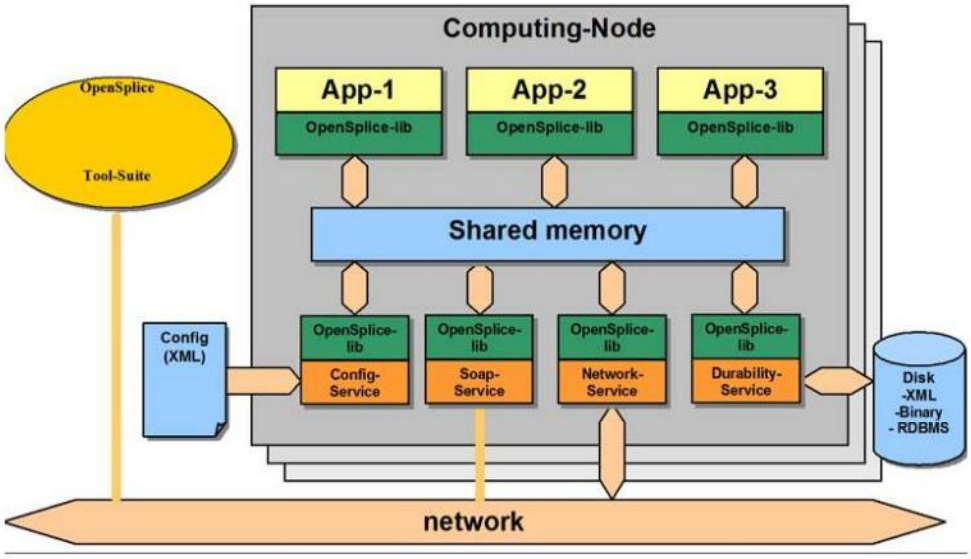

Figura 5.1. Arquitectura de OpenSplice DDS

OpenSplice DDS puede manejar millones de mensajes por segundo, asegurando un alto determinismo y muy bajas latencias. Su soporte avanzado para el tráfico de red permite que el sistema permanezca estable incluso en condiciones temporales de sobrecarga.

OpenSplice DDS se encuentra disponible para la mayoría de los sistemas operativos más extendidos, incluyendo Linux, Windows y Solaris, y otros más específicos para sistemas embebidos, como VxWorks e Integrity. Igualmente sucede con los lenguajes de programación, donde se suministran APIs para C, C++ y Java. La riqueza de plataformas y lenguajes permitidos permite que los programadores seleccionen la más apropiada en función de sus necesidades. OpenSplice también proporciona la herramienta de desarrollo PowerTools, basada en Eclipse, y que incluye desarrollo, despliegue, modelado de entidades o monitorización en tiempo de ejecución, entre otras funcionalidades.

- **OpenDDS**

OpenDDS [58] es una implementación DDS abierta, impulsada por el Object Computing Inc, y desarrollada en C++, Java y JMS binding. Se encuentra construido bajo el nivel de abstracción ACE (Entorno de Comunicación Adaptativo), para proveer portabilidad y aprovechar capacidades de la implementación de CORBA TAO, tales como su compilador IDL, y como base para el OpenDDS DCPS Information Repository (DCPSInfoRepo), que actúa como mecanismo de descubrimiento entre publicadores y suscriptores.

OpenDDS está escrito conforme a los perfiles *Minimun* y *Persistence* establecidos en la especificación DDS, aunque actualmente no soporta toda la familia de políticas de calidad de servicio, ni entidades relacionadas con el filtrado de contenido, como por ejemplo los *ContentFilteredTopics* y los *MultiTopic*.

También se provee de un framework de transporte para facilitar la distribución de datos en la red, ofreciendo los protocolos de transporte por defecto TCP/IP, UDP/IP e IP multicast para redes IPv4 e IPv6.

En comparación con otras implementaciones software, OpenDDS no ofrece ninguna plataforma de desarrollo o herramientas de monitorización en tiempo real, aunque están previstas para futuras releases. Se pretende también incorporar la implementación del perfil *Ownership* y el *Content-Subscription*, así como el soporte para alcanzar la interoperatividad con otras implementaciones DDS.

- **CoreDX**

CoreDX [59] es un middleware destinado a las comunicaciones centradas en datos para sistemas embebidos. Se encuentra desarrollado por Twin Oaks, cuyo trabajo se basa en alcanzar la interoperatividad entre sistemas, con especial atención a los empotrados. Con una experiencia de alrededor de 30 en el desarrollo y soporte de sistemas para la industria Aeroespacial y de Defensa, Twin Oaks Computing garantiza un Middleware robusto y esencial para la interoperatividad entre sistemas abiertos.

Concretamente, CoreDX DDS es una implementación del nivel DCPS de DDS, incorporando una API para el desarrollo de aplicaciones, un conjunto de políticas de calidad de servicios, y la implementación nativa del protocolo RTPS, que no hace uso de aplicaciones intermediarias ni aplicaciones externas que sirvan como pasarela, alcanzando el mayor rendimiento posible. Incluye soporte para Linux, Windows y Solaris, además de otros sistemas operativos embebidos, como son LynxOs, QNX y VxWorks. También incluye paquetes para la implementación de aplicaciones escritas en C, C++ y Java, y herramientas para la generación automática de código.

CoreDX, al contrario que otras implementaciones DDS, no se encuentra construido sobre CORBA, sino que hace uso interfaces nativos del Sistema Operativo, reduciendo el tiempo de

latencia y la demanda de recursos por parte de la aplicación. Además, al estar escrito en C, permite obtener un rendimiento óptimo y una mayor portabilidad a otras plataformas.

- **Otros productos**

  - InterCOMS DD de Gallium Visual Systems Inc. es un Middleware de alto rendimiento para la distribución de datos, que permite definir y compartir rápidamente datos en tiempo real a través de diferentes sistemas, redes, plataformas y procesadores. Junto a la herramienta InterMAPhics, forma parte de la suite InterWORPX, destinada al desarrollo de sistemas críticos, y proporciona una maximización de los recursos de la red, facilitando la integración de sistemas distribuidos complejos. Concretamente, InterCOM es una implementación del Middleware DDS, conforme al perfil mínimo definido en la versión 1.2, y proporcionando un API que se encuentra disponible en C++ y Java, ofreciendo un SDK para Windows, Linux y Solaris, y apoyándose en una completa documentación y en tutoriales para la ayuda al desarrollador.
  - MilSOFT DDS: provee un estándar DDS API en C++ y una interfaz para aplicaciones Java a través de JNI (Interfaz Nativa de Java), ofreciendo un servicio centrando en datos bajo el modelo publicador/suscriptor, y eliminando la necesidad de servidores en el middleware. Actualmente incluye soporte completo para el perfil *Minimum*y *Persistence,* e implementa de forma parcial el *Content Subscription* y el *Ownership*.
  - Micro DDS: es un Middleware reducido para microcontroladores que permite la comunicación necesaria con las aplicaciones de clientes en dispositivos pequeños, tales como sensores, actuadores, circuitos integrados, etc. Mediante la utilización de un API sencillo de utilizar (4 métodos), es posible aprovechar de las ventajas de DDS, tales como la tolerancia a fallos, baja latencia, o el autodescubrimiento de nodos.

- **Comparativa**

En la siguiente tabla se resumen las características de cada una de las implementaciones DDS que se han comentado anteriormente.

En la Tabla I se especifican los vendedores para cada una de las implementaciones, así como las licencias de los productos que ofrecen. Cabe destacar que OpenDDS y OpenSplice DDS son actualmente las únicas aplicaciones DDS con licencia gratuita y de software libre, si bien es cierto que Prismtech ofrece también otras alternativas más completas que sí son cerradas y comerciales.

Tabla I: Comparativa de las implementaciones DDS: vendedores y licencias

Producto	Vendedor	Licencia
RTI Data-Distribution Service	RTI	Comercial
OpenSplice DDS	Prismtech	LGPL y Comercial
OpenDDS	Object Computing Inc.	Open Source
CoreDX	Twin Oaks Computing Inc.	Comercial
MilSOFT DDS	MilSOFT	Privado
InterCOM DDS	Gallium Visual Sytems	Privado
MicroDDS	IcoupConsulting	?

En la Tabla II se resume los sistemas operativos y lenguajes de programación soportados por las diferentes implementaciones DDS. Como se puede observar, los sistemas operativos Windows y Linux están presentes en casi todas las implementaciones. También existe un amplio soporte para sistemas operativos embebidos tales como INTEGRITY o QNX.

En cuanto a los lenguajes de programación, C++ se decanta como el más extendido dentro de las implementaciones DDS analizadas.

Tabla II: Comparativa de las implementaciones DDS: Sistemas operativos y lenguajes de programación soportados

Producto	Sistemas operativos	Lenguajes
RTI Data-Distribution Service	INTEGRITY, Linux, SELinux, Embedded Linux, Lynx, QNX, Solaris, VxWorks, Windows XP, Vista, 2003 y CE	C, C++, Java, C#, Ada
OpenSplice DDS	Linux, Windows XP y 2003, Solaris, AIX, VxWorks, INTEGRITY	C, C++, Java, C#
OpenDDS	Linux, Windows XP, Vista, 2003 y Mobile 6, Solaris, Macintosh OS, QNX	C++
CoreDX	Linux, Windows XP y Vista, Solaris, Lynx, QNX, VxWorks	C++, Java
MilSOFT DDS	Windows, ?	C++
InterCOM DDS	Windows y Linux	C++, Java
MicroDDS	?	?

En la Tabla III se perfilan una serie de características que complementan a DDS. Entre ellas destaca el soporte al protocolo de interoperatividad DDSI/RTPS, que ya se ha convertido en un estándar de la OMG y es una funcionalidad deseable para toda implementación DDS. Asimismo, el nivel arquitectónico DLRL de DDS solo es soportado actualmente por OpenSplice DDS, aunque RTI está trabajando en ello. De la misma forma, RTI es el único vendedor que ofrece un marco de seguridad totalmente operativo en sus productos, aunque Prismtech ya ha definido un borrador del suyo propio. Por último, el soporte para bases de datos (DBMS) es actualmente ofrecido únicamente por RTI Data-Distribution Service y por OpenSplice DDS.

Tabla III. Comparativa de las implementaciones DDS: características (I)

Producto	Perfiles	DDSI/RTPS	DLRL	Seguridad	DBMS
RTI Data-Distribution Service	Minimun, Content-subscription*, Persistence, Ownership, Objectmodel*	Sí (2.1)	No	Sí	Sí
OpenSplice DDS	Minimun, Content-subscription*, Persistence, Ownership, Objectmodel*	Sí (2.1)	Sí*	Sí (draft)	Sí
OpenDDS	Minimum, Persistence, Objectmodel*	No	No	No	No
CoreDX	Minimun, Content-subscription*, Persistence, Ownership, Objectmodel*	Sí (2.1)	No	No	No
MilSOFT DDS	Minimun, Content-	Sí (2.0)	?	?	?

	subscription*, Persistence, Ownership*				
InterCOM DDS	?	?	?	?	?
MicroDDS	?	Sí	?	?	?

La Tabla IV es una continuación de las características de las implementaciones DDS. Entre ellas destaca la integración Web, que debido al auge de las tecnologías en la red en los últimos años, se trata de una alternativa realmente interesante para conectarse con dominios DDS. Esto posibilita que dispositivos con limitaciones hardware (como teléfonos móviles o dispositivos portables) puedan ser publicadores/consumidores de un dominio DDS mediante un navegador Web.

La capacidad de soporte en la red (Network) también es ofrecida por la mayoría de las implementaciones DDS, mayoritariamente empleando el protocolo no orientado a conexión UDP (Protocolo de Datagrama de Usuario).

Por último, la monitorización de Topics en tiempo real también se torna como una ventaja fundamental para cualquier desarrollador DDS. RTI presenta monitorización mediante hojas de cálculo (Spreadsheet), mientras que OpenSplice DDS y MilSOFT DDS proveen sus propias herramientas independientes denominadas Tuner y DDS SPY, respectivamente.

Tabla IV. Comparativa de las implementaciones DDS: características (II)

Producto	Integración Web	Network	Entorno de desarrollo	Monitorización
RTI Data-Distribution Service	Sí (draft)	Sí, UDP	Sí	Sí (Spreadsheet)
OpenSplice DDS	No	Sí, UDP	Sí	Sí (Tuner)
OpenDDS	No	Sí, TCP y UDP	Sí	No
CoreDX	No	Sí, TCP y UDP	No	No
MilSOFT DDS	No	?	No	Sí (DDS SPY)
InterCOM DDS	No	Sí	?	?
MicroDDS	No	?	?	?

En la Tabla V se listan las diferentes aplicaciones para las implementaciones DDS según la información contenida en las páginas web de los distintos vendedores.

Tabla V: Comparativa de las implementaciones DDS: Aplicaciones

Producto	Aplicaciones
RTI Data-Distribution Service	Aviación y defensa, comunicaciones, sistemas de control, sistemas de energía, sistemas financieros, transporte, vehículos no tripulados
OpenSplice DDS	Aviación y defensa, transporte, sistemas financieros, sistemas SCADA
OpenDDS	Sin especificar
CoreDX	Aviación, defensa, aplicaciones embebidas
MilSOFT DDS	Sin especificar
InterCOM DDS	Control de tráfico aéreo, sistemas de radar, defensa de misiles, sistemas militares
MicroDDS	Microcontroladores

5.3.2 Políticas de calidad de servicio en DDS

De entre los sistemas middleware basados en el paradigma publicador/suscriptor, el modelo DDS de OMG es uno de los pocos que proporcionan un conjunto suficientemente importante de políticas de calidad de servicio que permiten abarcar un gran número de requisitos necesarios en sistemas distribuidos y de tiempo real.

Estas políticas de calidad de servicio son objetos derivados de la clase *QosPolicy*, asociada a una determinada entidad del nivel DCPS, lo que supone que todos los componentes de comunicación (excepto los *listeners* y los relacionados con condiciones) pueden tener una serie de calidades de servicio adjuntas.

En DDS se pueden organizar las políticas de calidad de servicio en grupos, en función de la labor que desempeñan:

- Políticas de calidad de servicio relacionadas con aspectos de configuración:
  - UserDataQoSPolicy
  - TopicDataQoSPolicy
  - GroupDataQoSPolicy
- Políticas de calidad de servicio relacionadas con aspectos temporales y durabilidad:
  - DurabilityQoSPolicy
  - DurabilityServiceQoSPolicy
  - LivelinessQoSPolicy
  - LivelinessQoSPolicy
  - TimeBasedFilterQoSPolicy
  - DeadlineQoSPolicy
- Políticas de calidad de servicio relacionadas con el flujo de datos:
  - PresentationQoSPolicy
  - ReliabilityQoSPolicy
  - DestinationOrderQoSPolicy
  - HistoryQoSPolicy
  - ResourceLimitsQoSPolicy
  - OwnershipQoSPolicy
  - OwnershipStrengthQoSPolicy
- Políticas de calidad de servicio relacionadas con tiempo real:
  - TransportPriorityQoSPolicy
  - LatencyBudgetQoSPolicy
- Políticas de calidad de servicio relacionadas con componentes:
  - PartitionQoSPolicy
  - EntityFactoryQoSPolicy
  - WriterDataLifecycleQoSPolicy
  - ReaderDataLifecycleQoSPolicy

No es objetivo de este trabajo detallarlas todas, pero sí se presenta una tabla en donde se expone una comparativa de cada una de ellas, sus parámetros tal y como se definen en la especificación DDS, así como las que se encuentran implementadas actualmente en OpenSplice (software DDS que se utiliza en este trabajo).

Esta comparativa se encuentra contenida en el Anexo II de esta memoria.

5.4 Interfaz

Trata del diseño de una interfaz gráfico dependiente de la Web, con el objetivo de que permita a los profesionales, expertos en el dominio y expertos tecnológicos, un portal configurable donde se muestre información relevante acerca de sus disposiciones. El uso de tecnologías abiertas vinculadas a la web 2.0 permitirá que dicho entorno sea totalmente configurable y dinámico, de tal forma que se pueda dotar además de la capacidad de auto-aprendizaje, es decir, de la facultad de que el interfaz evolucione y se adapta conforme a la experiencia suministrada por lo los usuarios.

De esta forma, la interfaz gráfico debe presentar al usuario una serie de servicios disponibles acordes con sus requerimientos y privilegios de autorización. Esta información se encuentra contenida en el módulo de gestión de almacenamiento, y es necesaria la intermediación del MOM para el acceso a estos datos.

A continuación, se describe brevemente las necesidades tecnológicas que exigen esta tarea.

- Lenguaje del lado del cliente: que se ejecuta en el cliente, es decir, en el usuario, y que debe ser compatible con los navegadores existentes en la actualidad. Es el encargado del tratamiento de la jerarquía de objetos del DOM (Modelo de Objetos del Documento), y por tanto de la generación dinámica de contenido en la página.
  Actualmente JavaScript se presenta como la solución más extendida dentro de los lenguajes del lado del cliente. Es compatible con la mayoría de los navegadores actuales y cuenta con más de 15 años de desarrollo. El único competidor de peso existente es VBScript, propietario de Microsoft, que se trata de un lenguaje propietario y no goza de la misma aceptación que JavaScript en la Web
- Lenguaje del lado del servidor: se trata del contrapunto del lenguaje del lado del cliente y cuyo código es ejecutado por un servidor Web (normalmente Apache). Se encarga de gestionar las peticiones entrantes y acceder a la base de datos en el servidor.
  Entre los lenguajes del lado del servidor más difundidos se encuentran PHP (Procesador de Hipertexto), ASP (Páginas de Servidor Activo) y JSP (Páginas ServerPages). En este caso, la elección de PHP se hace indiscutible por tratarse de un lenguaje abierto, multiplataforma y sobradamente probado en un gran número de páginas en la Web.
  Una de las dificultades presentes a la hora acoplar este lenguaje dentro de la arquitectura propuesta, es la de conectar PHP con el MOM (en este caso DDS). Ninguna de las implementaciones actuales de DDS ofrece soporte para PHP, por lo que es necesario de un sistema intermediario que permita el trasvase de información entre ambas tecnologías, utilizando algún mecanismo similar al comentado en el apartado anterior con Matlab.
- Lenguaje de estructuración de contenido y diseño: donde HTML (Lenguaje de Marcado de Hipertexto) y CSS (Hojas de Estilo en Cascada) representan el paradigma para esta llevar a cabo esta tarea. Actualmente se está trabajando sobre la versión 5 de HTML, denominada HTML5, cuya principal novedad es la incorporación de efectos gráficos por medio del elemento Canvas, por lo que se presenta como un serio competidor de otras tecnologías cerradas como Flash o Silverlight. Una variación más restrictiva de este estándar en desarrollo es XHTML (Lenguaje Extensible de Marcado de Hipertexto), que incluye una serie de reglas sintácticas para hacerla compatible con XML. Por otra parte, CSS es un lenguaje de hoja de estilos que describe el diseño del contenido, y que se encuentra en su versión 3.

Con el objetivo de proveer una mayor interactividad, la tecnología abierta de desarrollo web AJAX (JavaScript Asíncrono y XML) se presenta como una técnica ideal que está alcanzando una gran popularidad en la red gracias a su gran potencial para realizar cambios en una página sin

necesidad de recargarla. Está basada en una comunicación asíncrona que corre en segundo plano y que aumenta la flexibilidad, velocidad y usabilidad de las aplicaciones. El uso de esta tecnología en nuestro caso se considera crucial para la visualización de monitorizaciones de variables medibles de un paciente que son tomadas en tiempo real.