UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Componente Gestor de Documentos Esquema de la VUCE.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Autor:

Rafael José Rojas Sánchez.

Tutor:

Ing. Eilys Pacheco Rodríguez.

Co-tutores:

Ing. Juan Javier Dans Moreno.

Ing. Leonardo Antúnez Naranjo.

Habana 2013

DECLARACIÓN DE AUTORÍA

Yo, Rafael José Rojas Sánchez con carnet de identidad 89102238427, declaro por este medio ser el autor del presente trabajo de diploma y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales, con carácter exclusivo.

Para que así conste firmo la presente en la Habana a los ____ días del mes de _____ del año 2013.

Autor:

_____

Rafael José Rojas Sánchez

Tutor:

_____

Ing. Eilys Pacheco Rodrígue**z**

Co-tutores:

_____ _____

Ing. Juan Javier Dans Moreno Ing. Leonardo Antúnez Naranjo.

A mis dos viejitas por su

amor infinito y apoyo incondicional.

Índice de contenidos

Resumen

La presente investigación está encaminada a obtener un mecanismo que permita homogenizar el proceso de gestión de documentos esquemas entre las entidades del país implicadas en el proceso de Intercambio Electrónico de Datos (EDI) para la facilitación comercial y aduanera. Tal mecanismo consiste específicamente en el establecimiento de un editor en línea, a través del cual, cada entidad pueda crear, editar y despachar directamente en la Ventanilla Única de Comercio Exterior (VUCE), los documentos esquema validadores de sus respectivos modelos electrónicos XML. Se incluye además, un navegador de archivos web para la gestión de los ficheros XSD presentes en el repositorio central de la VUCE.

El desarrollo está basado en tecnologías libres, multiplataforma y sobre una arquitectura en capas, utilizando PHP 5 como lenguaje de programación y el patrón de arquitectura Modelo Vista Controlador, legados del marco de trabajo Symfony 2; peticiones AJAX para la comunicación con el servidor así como la librería JQuery 2.0 conjuntamente con el framework[1] CSS Bootstrap 2.2, para obtener una interfaz visual moderna.

Palabras claves: Ventanilla Única de Comercio Exterior, Intercambio Electrónico de Datos, documento esquema.

[1]En el desarrollo de software, un framework o marco de trabajo, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos, que sirve de base para la organización y desarrollo de software.

Introducción

Las negociaciones de facilitación del comercio son las más avanzadas en el contexto del Programa de Doha para el Desarrollo[2] de la Organización Mundial del Comercio (OMC) debido a la importancia que tiene el comercio internacional en el desarrollo económico a escala global. En este ámbito, resulta típico que los países que logran atraer inversión extranjera y que impulsen el comercio exterior, posean mayor crecimiento económico – financiero. Teniendo en cuenta que, para incrementar el intercambio comercial no basta con reducir aranceles o eliminar procedimientos burocráticos; es necesario además, poner en marcha sistemas de modernización aduanera que permitan reducir costos e incrementar la competitividad internacional (1).

Los países con las mejores prácticas en el comercio exterior han adoptado el sistema VUCE, herramienta que permite el envío – recepción de información electrónica, una sola vez, ante una única entidad; posibilitando la simplificación, homologación y automatización de los procesos de gestión enfocados en el cumplimiento de los requerimientos del comercio exterior. En términos prácticos, la VUCE tiene como objetivo agilizar y simplificar los flujos de información entre el comercio y el gobierno, aportando beneficios significativos para todas las partes involucradas en el comercio transfronterizo (2).

En el mundo desarrollado, existen verdaderas potencias en materia de sistemas informáticos para la facilitación del comercio. Países como Corea, cuentan con avanzados sistemas de Intercambio Electrónico de Datos (EDI por sus siglas en inglés), capaces de viabilizar 23 mil trámites a diario y 120 millones de trámites anualmente, con un ahorro anual de 250 millones de dólares en gastos logísticos, conectando por internet a más de 87000 empresas comerciales, instituciones financieras y compañías logísticas (3).

La región de América Latina y el Caribe también se esfuerza notablemente en función de la facilitación del comercio internacional. En el año 2010 se celebró en Bogotá, Colombia, el “I Encuentro Regional Latinoamericano y del Caribe sobre Ventanillas Únicas de Comercio Exterior”, organizado por la Secretaría Permanente del Sistema Económico Latino Americano (SELA), el Ministerio de Comercio, Industria y Turismo de Colombia y la Cámara de Comercio de Bogotá, auspiciado además por la Corporación Financiera Internacional, agencia adscrita al Banco Mundial.

[2] Conjunto de acuerdos y planes de acción que rigen determinadas líneas de trabajo de la OMC.

Según los resultados arrojados hasta la fecha por el mencionado congreso, de los 28 países miembros del SELA: una docena reportaron una VUCE con diversos niveles de desarrollo: Brasil, Chile, Colombia, Costa Rica, El Salvador, Guatemala, Honduras, Nicaragua, Panamá, Paraguay, Perú y República Dominicana. Media decena de países reportaron la existencia de un proyecto para la creación de una VUCE con diversos niveles de avance: Argentina, Bolivia, Ecuador, México y Uruguay; mientras que Cuba, figuraba dentro del grupo de 11 países, junto con: Bahamas, Barbados, Belice, Granada, Guyana, Haití, Jamaica, Suriname, Trinidad y Tobago y Venezuela; de los que no se obtuvo referencia de algún avance en este tipo de proyecto (4).

Actualmente, las acciones de comercio exterior en Cuba demanda un alto número de trámites, papeleo y personas, donde se ven involucrados decenas de actores entre oficinas de gobierno, exportadores, transportistas y auxiliares de la función aduanera (5). A lo anterior se suma el inconveniente de que el declarante[3], se ve obligado a viajar al país al menos para gestionar los procesos de declaración y legalización de la mercancía con cada una de las entidades y ministerios competentes; lo cual constituye una limitación considerable que incide directamente en el factor competitividad internacional, pues no pocas aduanas, incluso en la región, permiten la gestión de estos trámites vía internet, las 24 horas del día, todos los días del año y desde cualquier parte del mundo.

Es por ello que en el año 2011, como parte de la política general del estado cubano para la automatización de los procesos aduaneros orientados a la facilitación del comercio exterior, sesionó en la Habana un encuentro entre los directivos del Departamento de Soluciones para la Aduana del Centro de Informatización de la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) y el Centro de Automatización de la Dirección y la Información (CADI) de la Aduana General de la República de Cuba (AGR), a partir del cual, se decidió iniciar el proyecto de desarrollo “Ventanilla Única de Comercio Exterior”, cuyo objetivo principal es la creación de un sistema que constituya el punto de entrada para las acciones de comercio exterior, donde converjan la totalidad de las entidades y ministerios del país competentes en cada instancia del proceso de comercialización, de modo que se simplifiquen trámites aduaneros como revisión, cuarentena, y declaración; así como la verificación de requisitos para la importación y la notificación de los resultados de la autorización. Se determinó además, utilizar el Lenguaje de Marcas Extensible (XML por sus siglas en inglés) para el marcado de los documentos

[3] Persona que realiza una declaración de mercancía o en cuyo nombre se realiza esta declaración.

electrónicos que llegarán a la VUCE; así como W3C XML Schema[4] para definir la estructura y restricciones de los documentos XML manejados en el sistema EDI.

Una vez desplegado el sistema VUCE, surgen nuevas problemáticas relacionadas con dependencias inevitables entre dicho sistema y las entidades involucradas en cada instancia del proceso de comercialización. Tales dependencias consisten específicamente, en que los XSD que utilizaría el sistema VUCE para el proceso de validación de los documentos XML intercambiados en el sistema EDI, necesariamente tienen que ser definidos y despachados por las entidades gubernamentales, debido a que son estas las expertas en la información relacionada con la estructura y restricciones que deben cumplir tales documentos XML. No obstante, la mayoría de dichas entidades no cuentan con el potencial informático que requiere la eficiente realización de esta tarea, debido a limitaciones tanto en recursos humanos como tecnológicos. Sin embargo, existen otras entidades, como por ejemplo la AGR, que presenta su propio mecanismo de creación de esquemas XML utilizando la herramienta privativa Oxygen[5], lo cual atenta contra la soberanía tecnológica a la cual aspira nuestro país.

En sentido general, entre las entidades competentes en los procesos de importación, exportación y tránsito de mercancías correspondientes a las acciones de comercio exterior del país, existe un desnivel considerable en lo que se refiere a potencialidades necesarias para una gestión eficiente de los esquemas validadores de sus documentos electrónicos XML. El principal riesgo que trae consigo, radica en que la disponibilidad de la información requerida por el sistema VUCE para la validación de los documentos XML asociados al sistema EDI, puede verse afectada de manera potencial; atentando considerablemente contra la fiabilidad, consistencia y precisión de la información electrónica que se manipula e intercambia, así como la agilidad y eficiencia del proceso de comercialización en su conjunto.

Debido a lo anterior, resulta indispensable la creación de un mecanismo que permita a cada entidad definir y despachar los esquema XML correspondientes con sus documentos electrónicos XML de una forma fácil, ágil e interactiva; y que permita homogenizar sus posibilidades de gestión, de manera que no sea un requisito obligatorio poseer amplios conocimientos informáticos ni específicos de lenguajes esquema, sino que la realización de esta tarea esté más cercana al alcance del usuario común.

[4] También se le conoce como XSD, del inglés XML Schema Definition.
[5] Entorno Integrado de Desarrollo con soporte para escribir documentos esquemas.

A partir de la problemática antes descrita, se determina como **Problema de investigación:** ¿cómo garantizar que las entidades puedan gestionar sus XSD directamente en la Ventanilla Única de Comercio Exterior de Cuba, para la validación de la información electrónica XML intercambiada en el sistema EDI?

Objeto de estudio: herramientas para la definición de documentos esquema.

Teniendo como **Campo de acción**: subsistema para la validación XML de la VUCE.

Para dar solución al problema de investigación identificado se propone el siguiente **Objetivo general:** desarrollar un componente que le permita a las entidades gestionar los XSD en la Ventanilla Única de Comercio Exterior de Cuba para la validación de la información electrónica XML intercambiada en el sistema EDI.

Objetivos específicos:

- Elaborar el marco teórico relativo a las soluciones de software que gestionan la creación y presentación de documentos esquema.
- Modelar los procesos de negocio asociados al componente *XML_Schema* de la Ventanilla Única de Comercio Exterior de Cuba.
- Identificar, describir y validar los requerimientos asociados a los procesos de negocio modelados.
- Modelar el diseño del componente *XML_Schema* de la Ventanilla Única de Comercio Exterior de Cuba.
- Implementar el componente *XML_Schema* de la Ventanilla Única de Comercio Exterior de Cuba.
- Realizar pruebas unitarias al componente *XML_Schema* de la Ventanilla Única de Comercio Exterior de Cuba.

Teniendo en cuenta la situación problémica antes planteada, se tiene como **Idea a defender**: el establecimiento de un Editor XSD en línea y un navegador de archivos web en la VUCE, permitirá a las entidades gestionar los XSD para la validación de la información electrónica XML intercambiada en el sistema EDI.

Los **métodos científicos** aplicados para llevar a cabo la investigación fueron los siguientes:

Métodos teóricos:

- *Analítico – Sintético*: para analizar el problema de investigación en el estudio por separado de las reglas de definición los esquemas XML, flujos de trabajo asociados a sistemas de gestión, así como la definición y control de las reglas restrictivas para la creación de documentos esquemas, para luego sintetizarlos en la solución general del problema planteado.
- *Histórico - Lógico*: para analizar la historia y trayectoria de las herramientas de definición de documentos esquema y para expresar teóricamente su esencia.
- *Modelación:* empleado en las disciplinas de análisis y diseño para la elaboración de los artefactos establecidos por el modelo de desarrollo de software.
- *Sistémico*: facilita la integración de cada uno de los elementos estudiados en la conformación de la propuesta de solución.
- *Inductivo-Deductivo*: para extrapolar la teoría estudiada al caso específico del problema a resolver.

Métodos empíricos:

- Medición: como procedimiento para obtener información numérica acerca de algunas características de la solución obtenida, como por ejemplo su rendimiento.
- Experimental: para realizar las pruebas unitarias a la solución, mediante el desarrollo de casos de pruebas experimentales previamente diseñados.

El documento de tesis que se propone, se encuentra estructurado en los 4 capítulos descritos a continuación:

- **Capítulo 1. Fundamentación teórica:** incluye un estudio crítico y valorativo de las tendencias, técnicas, tecnologías, metodologías y software usados en la actualidad y en las que se apoya la presente investigación para dar solución al problema que se enfrenta. Se refiere además, al significado que poseen las categorías y términos fundamentales que se exponen en el planteamiento del problema, los objetivos y la hipótesis.
- **Capítulo 2. Características del sistema:** se describen los procesos del negocio que serán objeto de automatización, documentos específicos que se procesan, así como el funcionamiento general del componente en su conjunto. Se refleja además, un listado con los requisitos funcionales y no funcionales que rigen a la solución propuesta.

- **Capítulo 3. Análisis y diseño:** refleja todo un conjunto de diagramas, modelos y gráficos que constituyen los cimientos de todo producto de software. Engloba elementos de diseño y arquitectura que conforman el núcleo del componente propuesto.
- **Capítulo 4. Implementación y prueba:** refleja elementos de la implementación de la solución expresada en diagramas de componentes, engloba además modelos de casos de pruebas a través de los cuales se verificó y validó la propuesta de solución.

Capítulo I. Fundamentación teórica

Introducción

El presente capítulo tiene como objetivo abordar los diferentes elementos que brindan la base teórico conceptual para el desarrollo de la propuesta de solución, valorándose de forma crítica las tendencias y tecnologías actuales que inciden directamente en el dominio de la investigación. Se abordan cuestiones técnicas como modelo de desarrollo a utilizar, así como tecnologías y herramientas empleadas. Sirva este capítulo como base para realizar una correcta interpretación de la problemática y del problema a resolver, de cara al basamento teórico sobre el que sustenta la presente investigación.

1.1. Marco conceptual

1.1.1. Ventanilla Única de Comercio Exterior (VUCE)

El surgimiento del concepto de Ventanilla Única (VU) no se debe exclusivamente a las operaciones del comercio exterior, su origen está enmarcado en un ámbito mucho más amplio donde ha sido llevado a la práctica con anterioridad. En este sentido, al referenciar el origen de las VUCE, es casi obligatorio abordar las etapas por las que transita la VU como sistema informático, durante el proceso de instauración de un Gobierno Electrónico (GE) en determinado país.

Actualmente, muchos autores definen como Ventanilla Única de Gobierno Electrónico (VUGE) a aquel sitio o portal en internet estructurado y diseñado con el objetivo de crear un único espacio virtual, en donde de manera centralizada, se pone a disposición de los ciudadanos y las empresas una amplia variedad de servicios y trámites, los cuales son ofrecidos por una diversa gama de instituciones del Estado (6).

Sin embargo, la definición anterior hace referencia solamente a un caso particular de VUGE entre los que coexisten durante las etapas de instauración de un GE. Cabe destacar que, no siempre es posible instaurar una VUGE que ofrezca toda una gama de servicios de facilitación de trámites vía internet, sino que, este tipo de sistema conjuntamente con las prestaciones que está en capacidad de brindar, deben estar a tono con la etapa en que se encuentre el proceso de instauración de GE en su campo de acción. La definición anterior se refiere esencialmente a una etapa avanzada en la automatización de los procesos

de gobierno, en que el sistema VUGE ha alcanzado su máxima expresión como herramienta indispensable para la facilitación de trámites y transacciones entre ciudadanos, empresas y Estado.

Se definen al menos tres etapas en el proyecto para el desarrollo de una VU consecuentemente con el proceso de implantación de un GE (6):

- **Ventanillas Informativas**: son la muestra más básica de una VUGE, dado que se reducen a representar la referencia sobre algunos servicios de determinadas instituciones del Estado. De hecho, se sita sólo con carácter documental, pues en sentido general no es tomada en cuenta como una buena práctica. Su implementación puede conllevar a la confusión de objetivos y conceptos, así como al desaprovechamiento de recursos.
- **Ventanillas de Puntos de Enlace**: esta es una opción para la primera etapa de GE. Su fin es crear un punto virtual desde donde el usuario pueda encontrar los diferentes hipervínculos hacia las múltiples instituciones de Estado y por consiguiente, a los servicios digitalizados por cada institución en particular. Su aplicación tiene sentido cuando ha existido un proyecto de GE poco sistematizado y con iniciativas aisladas en cuanto al desarrollo digital de las instituciones, conllevando a la inexistencia de una vinculación transversal entre los sistemas y bases de datos de las instituciones del Estado. Es oportuno utilizar este tipo de ventanilla en cuanto se fortalece una cultura digital en los ciudadanos y empresas, y se delínea una estrategia de GE más estructurada.
- **Ventanillas Transaccionales**: representan un ejemplo de GE aplicado y desarrollado estratégicamente utilizando las Tecnologías de la Información y las Comunicaciones (TIC) en función de la modernización del Estado. Son proyectadas a largo plazo con una articulación tanto tecnológica como política que permite al ciudadano y a las empresas realizar electrónicamente y desde un solo punto virtual sus trámites de manera ágil, eficiente y segura.

De lo anterior se deduce que, la VUCE es un subconjunto de VUGE del tipo Transaccional, típicas de un proyecto de GE ya establecido y con un alto grado de madurez; donde los servicios y trámites que se ofrecen están basados en Estándares de Interoperabilidad Tecnológica[6], de manera que sólo se requiere el envío de información electrónica una sola vez, ante una única entidad, para cumplir con todos los

[6] Propiedad que poseen las bases de datos y sistemas de información de comunicarse entre sí de manera independiente a la plataforma sobre la cual han sido desarrollados.

requerimientos del comercio exterior. Siendo esto posible a partir de la simplificación, homologación y automatización de los procesos de gestión de cada una de las instituciones del Estado.

Las expectativas de los países con la implementación de las VUCE son muchas; el trabajo de su implementación requiere tecnología, metodología y una gobernanza[7] que, jurídicamente respaldada, pueda conciliar los intereses dentro y fuera del gobierno con todos los usuarios. Así mismo, requiere que se convierta en un pilar indispensable políticamente para la facilitación y la competitividad. Algunos beneficios que reportan los países con la implementación de sus VUCE son (7):

- Simplificación de trámites.
- Reducción de tiempos en la obtención de permisos y licencias.
- Transparencia en la interacción de los actores públicos y privados (trazabilidad).
- Reducción en los costos de movilización de sus tramitadores de institución en institución.
- Reducción de los costos en papelería y en las certificaciones.
- Economía de escala en la inversión y mantenimiento tecnológico.
- Controles estadísticos para decisiones comerciales públicas y privadas.
- Transparencia y seguridad financiera con la utilización del pago electrónico.

1.1.2. Intercambio Electrónico de Datos

En los países desarrollados, el sistema informativo contable de cualquier empresa, por pequeña que sea, se encuentra informatizado, de forma que habitualmente se logran elevados niveles de automatización de las tareas administrativo-contables. Por ejemplo, es frecuente que se encuentren integrados los programas de contabilidad con los que gestionan la tesorería o la nómina y que estos datos se procesen muy rápidamente. Pero puede suceder que, dos empresas que mantienen una intensa relación comercial y que disponen de avanzados sistemas informáticos, realicen sus transacciones económicas introduciendo las órdenes de compra, las facturas y el resto de los documentos en sobres, que posteriormente son enviados por correo. La solución que desde hace varios años están adoptando muchas empresas se denomina Intercambio Electrónico de Datos (EDI por sus siglas en Inglés) (8).

[7] Concepto de reciente difusión para designar la eficiencia, calidad y buena orientación de la intervención del Estado.

La analogía entre el actual proceso EDI y los métodos tradicionales a través de los cuales se llevaba a cabo el intercambio de datos, se reflejan en la siguiente figura:

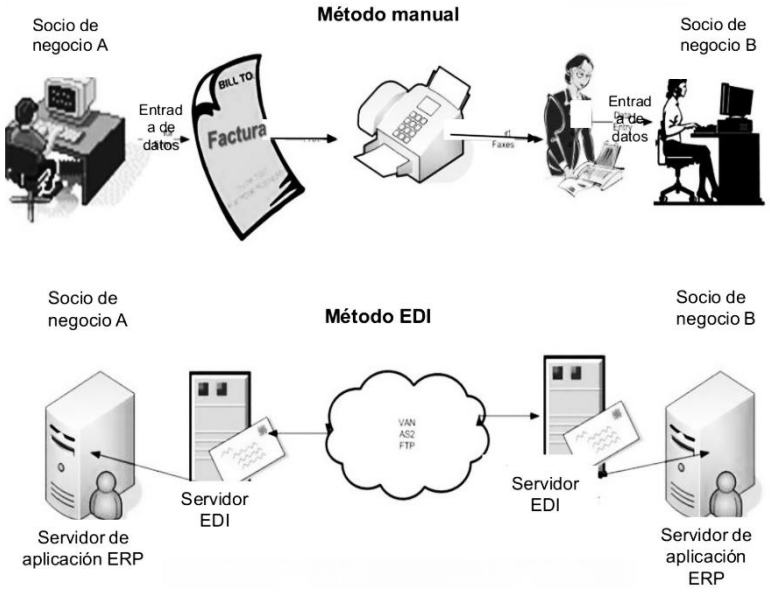


Figura 1.0. Analogía entre EDI y los métodos tradicionales de intercambio de datos.

(Fuente: Extraído de la fuente bibliográfica (9)).

EDI básicamente consiste en transmitir electrónicamente documentos comerciales y administrativos entre aplicaciones informáticas, en un formato normalizado[8]. Los sistemas EDI son un tipo de Sistemas de Información Interorganizativos, que han aumentado considerablemente su presencia en el mundo de los negocios de los últimos años. Tanto es así que han pasado de considerarse fuentes de ventajas competitivas a un imperativo, sobre todo en las grandes empresas, muchas de las cuales se niegan a hacer negocios con otras que no cuenten con esta tecnología (9). El sistema EDI, trata por tanto, de sustituir el soporte físico (papel) de los documentos mercantiles que intercambian las entidades, por transacciones electrónicas entre sus respectivos ordenadores, de manera que el contenido es transmitido de acuerdo a un determinado formato y sintaxis fijado de antemano (8). En definitiva, es posible definir EDI como el Intercambio Electrónico de Documentos Normalizados Universalmente, que se realiza entre aplicaciones informáticas (10).

[8] Se refiere a una serie de especificaciones acerca de la forma, la estructura y el lenguaje que debe llevar la información antes de ser intercambiada.

Las citas anteriores, se complementan de manera tal que en su conjunto reflejan la esencia de un sistema EDI, en las cuales se evidencian tres componentes fundamentales para conseguir una correcta aplicación de este tipo de tecnología, entre ellos: 1) las telecomunicaciones como canal de distribución de la información, 2) la estandarización a través de la normalización y 3) la integración de la información, como vía de automatización de los procesos informáticos.

De los componentes anteriores, resulta de especial importancia para la presente investigación, la *Estandarización a través de la Normalización* de los documentos electrónicos, debido a la estrecha relación que existe entre uno de los estándares internacionales empleados en este propósito –específicamente XML– y el objeto de estudio del presente trabajo de diploma.

Así como en el contexto social el lenguaje internacional para comunicarse con cualquier persona es el inglés; en EDI, existe un "lenguaje" internacional para asegurar que la información que se envíe a cualquier socio comercial (independientemente del país donde se encuentre) pueda ser interpretada correctamente. Análogamente, tal lenguaje está definido en el contexto tecnológico, por alguno de los estándares que se relacionan a continuación: ANSI, ASC, X12, UN/EDIFACT y XML (10).

A modo de síntesis, EDI representa en la actualidad una práctica empresarial indispensable para la obtención del factor competitividad en cualquier entidad, donde convergen procesos de negocio de última generación soportados en las TIC, aplicados al campo de la comunicación entre empresas. Los beneficios que esta tecnología reporta en el ámbito empresarial –según historias de éxito que proliferan en la literatura especializada sobre negocios–, están relacionados con indicadores como los que se enuncian a continuación (10):

- **Calidad de la información:** se logra mayor exactitud de la información, reduciendo los errores al evitar la reentrada de datos.
- **Organización del trabajo y ahorro de tiempo:** la transmisión de información entre organizaciones es notablemente más ágil. Mejora el flujo de efectivo a través de la gestión y transferencia electrónica de fondos. Posibilita trabajar independientemente de los horarios de apertura de los socios comerciales. El ciclo de ventas-facturas-pagos se agiliza de manera potencial.
- **Relaciones comerciales:** el sistema favorece dichas relaciones al disminuir los tiempos de atención al cliente y brindar una mejor información. Algunos autores relacionan EDI con procesos

relacionados con la Gestión Total de la Calidad (TQM por sus siglas en inglés), ya que ambos trabajan el tema de la satisfacción al cliente a partir de requisitos relacionados con la disponibilidad y calidad de la información.

- **Reducción de costos:** aumenta la productividad, lo que se traduce en la eliminación de trabajos relacionados con recoger, enviar y recibir información; se aminora el flujo de papel entre empresas y descienden los costos de fax, correo y teléfono. También hay reducción de inventarios y costos derivados, al disminuir o incluso desaparecer los tiempos necesarios para llevar a cabo las órdenes.

1.1.3. Lenguaje de Marcas Extensible (XML)

Como se observa en el epígrafe anterior, XML es uno de los lenguajes (o estándar de normalización) utilizados para la comunicación entre aplicaciones en los sistemas EDI. En este contexto, el presente epígrafe abordará XML desde un punto de vista estructural y de validez semántica; teniendo en cuenta que, de manera predefinida, XML es el lenguaje que define sintácticamente los documentos electrónicos que son gestionados en el campo de acción del presente trabajo de diploma.

Panorámica general

XML es esencialmente un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del Estándar de Lenguaje de Marcado Generalizado (SGML por sus siglas en inglés) y permite definir la gramática de lenguajes específicos para estructurar documentos complejos. XML brinda soporte a bases de datos, siendo útil cuando varias aplicaciones requieren comunicarse entre sí o integrar información. Su papel en el contexto informático es primordial en términos de compatibilidad entre sistemas automatizados e intercambio de información. XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible (11).

Documentos XML *bien formados* y condiciones de validez

Para que un documento XML pueda considerarse *bien formado* debe adherirse a las reglas de sintaxis descritas en la especificación XML 1.0 del W3C. En su nivel base, un documento XML bien formado requiere (12):

- Incluir una *declaración XML* como etiqueta obligatoria, la cual proporciona hasta tres tipos de informaciones claves sobre el documento que la contiene:
    1. Información de la versión XML (obligatoria): la versión más utilizada es la 1.0 (aunque ya está disponible la versión 1.1 de XML).
    2. Codificación de caracteres utilizada (opcional): hace referencia al modo en que se representan internamente los caracteres, normalmente UTF-8 o UTF-16.
    3. Declaración independiente (opcional): indica al procesador XML si un documento es independiente (standalone=”yes”) o se basa en información de fuentes externas.

       Ejemplo:

       ```
       <? xml version="1.0" encoding="UTF-8" standalone="no" ?>
       ```

- Sólo existe un *único elemento raíz* para cada documento: elemento a partir del cual todos los demás elementos y contenidos se encuentran anidados y correctamente cerrados.
- Todos los elementos deben estar delimitados por una etiqueta inicial y otra final con el mismo nombre, teniendo en cuenta además que XML distingue entre mayúscula y minúscula.
- Los valores de atributos siempre deben estar encerrados entre comillas simples o dobles.
- Los elementos vacíos deben terminar con “/” (autocierre) o añadiendo una etiqueta de fin, teniendo en cuenta que no pueden haber etiquetas aisladas.
- XML es sensible a mayúsculas y minúsculas (case-sensitive) y los nombres de las etiquetas pueden ser alfanuméricos, pero siempre comenzando con una letra.

Cuando un documento XML adquiere la condición de *bien formado,* permite una mejor comprensión del sentido fundamental de la construcción XML y de su naturaleza abstracta. No obstante, el hecho de que un documento XML esté *bien formado,* solamente se refiere a su estructura sintáctica básica, de manera que tal condición representa sólo una parte del proceso de validación que se lleva a cabo en la práctica (13). Cada aplicación de XML o lenguaje definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento. Esta relación entre elementos se refiere a una condición o requisito más complejo con el que debe cumplir un documento XML, el cual determina su condición de validez (14).

Por tanto, el proceso de validación XML además de comprobar que un documento XML está *bien formado*, verifica que éste se ajusta a una estructura definida. Tal definición se lleva a cabo en la práctica a través de: Definición de Tipo de Documentos (DTD) o esquemas XML. Entre los esquemas XML mayormente utilizados en la práctica, destacan: *W3C XML Schema*, *RELAX NG y Schematron,* los cuales serán abordados más adelante.

Documento XML válido

En la práctica, controlar los elementos y atributos que se pueden utilizar y la manera de disponerles en el diseño de un documento XML, es fundamental en los actuales procesos de negocio. Por ejemplo, una empresa puede requerir que los documentos internos para describir el software que utiliza, deben poseer como raíz un elemento llamado *software* (y no otro) y que dicho elemento debe contener los elementos *nombre, fabricante* y *precio.* Estas reglas no se refieren a que el documento esté *bien formado*; en realidad, son reglas más complejas que determinan que el documento sea válido.

El proceso de validación de encarga de verificar (15):

- **La corrección de los datos:** el proceso de validación permite detectar formatos nulos o valores fuera de rango, además de elementos o atributos no permitidos en el documento.
- **La integridad de los datos:** al validar se comprueba que toda la información obligatoria está presente en el documento y la corrección estructural de elementos y atributos.
- **El orden de los elementos:** tiene que ver con la ubicación jerárquica de los elementos, pues la presencia de un tag en una u otra posición del esquema, incide en el sentido semántico del documento en cuestión.
- **La unicidad de valores dentro del documento:** comprueba que determinado elemento no tenga más de una ocurrencia en el esquema, útil para comprobar unicidad de claves y valores únicos como identidad, número de serie, etc.
- **El entendimiento compartido de los datos**: favorece a que el emisor y el receptor perciban el documento de la misma manera en los procesos EDI.

La validación garantiza que los documentos cumplen con reglas más concretas, lo cual posibilita crear en las empresas un protocolo de documentos electrónicos y de comunicación con otras entidades. Controlar el diseño de documentos XML a través de esquemas de validación aumenta su grado de fiabilidad,

consistencia y precisión, facilitando su intercambio entre aplicaciones y usuarios. Un documento XML válido es sinónimo de utilidad y funcionalidad (16).

1.1.4. Estándares para la validación de documentos XML

Uso de las DTD

Una DTD es básicamente una definición de estructura y sintaxis de un documento XML. Utiliza una sintaxis no-XML (basado en etiquetas) para definir la estructura o modelo de contenido de un documento XML válido. Cabe mencionar, que es el mecanismo nativo (y el más antiguo) para validar documentos XML ya que forma parte del propio lenguaje (17). Las DTD tienen a su favor que es el método más sencillo que existe para validar documentos XML, además de poseer compatibilidad con un importante número de aplicaciones y herramientas contemporáneas a su tiempo y que aún conservan su utilidad. Aunque en contraparte, presenta notables limitaciones relacionadas con el soporte a nuevas ampliaciones e incapacidad de describir ciertos aspectos formales de un documento a nivel expresivo; entre otras como las que se relacionan a continuación (15):

- **No posee soporte para espacios de nombre (namespace):** es imposible definir elementos locales que sólo sean válidos dentro de otros elementos. Por ejemplo, si queremos tener un elemento <Manager> que describa al gestor de una compañía o al de una delegación, y la definición de *Manager* es diferente en cada caso, con una DTD habría que crear –por ejemplo- los elementos “CompanyManager” y “DelegationManager” para evitar el conflicto de nombres. La falta de jerarquía en una DTD obliga a crearla manualmente a base de guiones o puntos en el espacio de nombres.
- **Limitada tipología de datos:** no permite definir que un elemento pueda ser de tipo número, fecha, etc. sino que sólo presenta variaciones limitadas sobre cadenas de texto plano (CDATA).
- **Presenta un mecanismo de extensión complejo y frágil:** tal mecanismo está basado en sustituciones sobre cadenas y no hace explícitas las relaciones, esto provoca que dos elementos que tienen definido el mismo modelo de contenido, pueden no presentar ninguna relación.

La necesidad de superar estas limitaciones, propicia la aparición de lenguajes de esquema como *W3C XML Schema,* como alternativas a las ya obsoletas DTD.

Uso de W3C XML Schema

W3C XML Schema, o XSD (del inglés XML Schema Definition, en lo delante se utilizarán indistintamente ambos términos) como también se le denomina, es uno de los tantos lenguajes de esquema XML disponibles actualmente. Su versión 1.0 fue publicada como una recomendación del W3C en mayo de 2001, con posteriores ediciones en el año 2004; hasta evolucionar completamente a su versión 1.1, convertida en recomendación del W3C en el año 2012. Como todos los lenguajes de esquemas, XSD es usado para expresar un conjunto de reglas con las que un documento XML debe cumplir para considerarse válido (18).

Como factor en contra, hay diversos autores que coinciden en que la utilización de XSD a la hora de validar un documento XML, supone un gran consumo en recursos y tiempo debido a su gran especificación y complejidad en la sintaxis (18). Aunque por otro lado, presenta notables potencialidades que sugieren mucho su utilización. Entre las funcionalidades más relevantes que provee el uso de esta tecnología se destaca que, luego del proceso de validación, es posible expresar la estructura y contenido del documento XML validado, en términos del modelo de datos usado por el esquema de validación. Esta funcionalidad, conocida como Post-Schema-Validation Infoset (PSVI), se puede utilizar para transformar el documento en una jerarquía de objetos, a los cuales se puede acceder a través de un lenguaje de Programación Orientada a Objetos (POO). Este modelo de datos incluye: el vocabulario (nombres de elemento y atributo), el contenido modelo (relaciones y estructura) y tipos de datos (19).

Los XSD tienen un enfoque modular que favorece la reutilización de código, los tipos de datos tienen una función análoga al concepto de *clase* de los lenguajes de POO; el usuario puede construir tipos de datos (llamados *Arquetipos*) a partir de los tipos predefinidos, agrupando elementos y atributos de un modo específico y con un mecanismo de extensión análogo a lo que en POO se denomina Herencia (20). Los tipos de datos disponibles en la versión 1.1 se pueden consultar en el **anexo 1**.

XSD incluye además el uso de espacios de nombre (namespace), característica que permite definir elementos con igual nombre dentro del mismo contexto, siempre y cuando se anteponga un prefijo al nombre del elemento. El uso de esta característica favorece la reutilización de código al evitar ambigüedades en el contenido del documento (20).

XSD vs. DTD

XSD ofrece las siguientes ventajas con respecto a las DTD (21).

- **Tipado fuerte:** En una DTD, las posibilidades de controlar el contenido de un elemento o un atributo son limitadas. Por ejemplo, un elemento se puede definir como EMPTY, ANY, un modelo de grupo o contenido mixto (elemento y texto). Pero no hay forma de especificar que el contenido de un elemento tiene que ser un entero o no puede exceder un cierto número de caracteres.
- **Tipos de datos similares a los lenguajes de programación y bases de datos:** XSD incluye una serie de tipos de datos básicos similares a los que se puede encontrar en la programación o en las bases de datos orientadas a objetos o relacionales. Pero además, el usuario puede definir sus propios tipos de datos a partir de los existentes.
- **Permite controlar con precisión el número de repeticiones:** En un DTD, la repetición de un elemento se describe con los siguientes modificadores: *"nada"* para una sola vez, "?" para cero o una vez, "*" para cero o más veces y "+" para una o más veces. En XSD se puede especificar cualquier valor para el número de ocurrencias mínimo y máximo.
- **XSD es un documento XML**: Una DTD sigue una notación que no tiene nada que ver con la sintaxis de un documento XML. Sin embargo, un XSD es un documento XML válido, lo cual permite emplear todas las herramientas desarrolladas para trabajar con tecnologías XML.
- **Verdadera representación de claves primarias y ajenas.** Con DTD, si se quiere representar una base de datos en un documento XML, la única forma de especificar claves primarias y ajenas es con los tipos ID e IDREF. Sin embargo, presenta dos graves inconvenientes: ID tiene que ser único en todo el documento (en una base de datos, una clave primaria se puede repetir en distintas tablas) e IDREF tiene que referenciar un ID dentro del propio documento, pero no se puede especificar el tipo de elemento (puede ser una clave ajena a cualquier tabla). Con XSD, estos conceptos se pueden representar sin problemas.

Uso de RELAX NG

RELAX NG (por sus siglas en inglés **RE**gular **LA**nguage for **X**ML **N**ext **G**eneration) es un lenguaje de esquema para XML basado en la gramática, muy intuitivo y fácil de entender comparado con los principales lenguajes de esquemas existentes actualmente –de ahí su popularidad–, además de poseer un alto poder expresivo. Los esquemas RELAX NG especifican patrones para la estructura y contenido de un

documento XML. Este tipo de esquemas es en sí mismo es un documento XML; aunque, también ofrece un modo compacto con sintaxis no-XML que se ha hecho muy popular. Entre sus principales características, cabe destacar además, la capacidad de usar plugin[9] de definiciones de tipos de datos de XSD, combinando así las ventajas de ambos lenguajes (22). La especificación fue definida por el comité técnico OASIS RELAX NG en los años 2001 y 2002 y estandarizado según ISO/IEC 19757, como segunda parte de los Lenguajes de Definición de Documentos Esquemas (DSDL por sus siglas en inglés) (23).

Aunque la especificación RELAX NG es contemporánea con la especificación XSD, este último fue mejor conocido y más ampliamente implementado tanto en editores como analizadores sintácticos de código abierto y propietario, a partir del momento en que se convirtió en recomendación del W3C en el año 2001. Sin embargo, RELAX NG ha ido ganando paulatinamente en popularidad y aceptación en el software XML, e incluso, actualmente es más adoptado como esquema primario para los populares lenguajes de marcado centrados en los documentos tales como: DocBook[10], Directrices TEI (por sus siglas en inglés, Iniciativa de Texto Codificado), OpenDocument[11] y EPUB[12] (24).

RELAX NG comparte con XSD varias funcionalidades que evolucionan las prestaciones de las tradicionales DTD, entre ellas destacan: fuerte soporte para tipos de datos, expresiones regulares, namespace, y definiciones complejas.

Ventajas sobre otros lenguajes de esquema

La especificación RELAX NG, además de poder basar su sintaxis en lenguaje XML, posee una sintaxis compacta que presenta cierta analogía con las DTD, pero es mucho más potente a nivel expresivo. Las principales herramientas que soportan el lenguaje, conmutan entre ambas formas sin pérdida de funcionalidad. La ventaja de la forma compacta radica en su sencillez, lo cual facilita su interpretación de cara al usuario sin que esto afecte el rendimiento a nivel computacional (25).

---

[9] Características que se adicionan a entes computacionales con el objetivo de extender funcionalidades base.

[10] Aplicación del estándar SGML/XML utilizado principalmente en el área de la documentación técnica. Es un lenguaje semántico que permite crear documentos en un lenguaje neutro que engloba tanto el contenido como la estructura lógica del documento.

[11] Formato de archivo basado en XML para hojas de cálculo, gráficos, presentaciones y documentos de procesamiento de texto.

[12] Proviene del inglés *Electronic Publication,* es un estándar libre y abierto para libros electrónicos (e-book) perteneciente al IDPF (del inglés International Digital Publishing Forum).

RELAX NG provee un buen soporte para contenido desordenado, lo cual permite especificar secuencias de patrones sin restricciones en cuanto al orden en que estos tengan que ser dispuestos en el documento. Lo cual ofrece muchas libertades al programador o autor del documento esquema. La especificación soporta además, especificar atributos que sean tratados como elementos en el modelo de contenido; lo cual puede ser muy útil, por ejemplo, para especificar elementos bajo condiciones como en el siguiente ejemplo (25):

```
1.  <choice>
2.      <attribute name="has_name">
3.          <value>false</value>
4.      </attribute>
5.      <group>
6.          <attribute name="has_name">
7.              <value>true</value>
8.          </attribute>
9.          <element name="name"><text /></element>
10.     </group>
11. </choice>
```

Mientras que XSD no permite especificar dependencias entre el contenido de un atributo y elementos secundarios.

La especificación RELAX NG sólo lista dos tipos de datos integrados: *string y token*. No obstante, la falta de listas específicas permite al procesador soportar tipos de datos que son muy específicos del dominio del problema que se esté tratando. Además, la mayoría de los esquemas RELAX NG pueden ser algorítmicamente convertidos en esquemas XSD o incluso DTD, pero no a la inversa (23).

Desventajas

En determinadas ocasiones, cuando se mencionan factores en contra o desventajas de una u otra tecnología, estos suelen tener un carácter relativo. Es decir, lo que en determinado contexto puede parecer una limitación, en otro puede constituir un requisito fundamental. Por tal motivo, y debido a que esto se manifiesta especialmente en la especificación RELAX NG, es necesario aclarar que los factores que a continuación se expondrán, constituyen limitaciones importantes en relación al uso del estándar en el campo de acción del presente trabajo de diploma. Intencionalmente, se establece un contraste entre

tales limitaciones y la especificación XSD, teniendo en cuenta que, ambas tecnologías se consideran los principales exponentes en el campo de los mecanismos de validación XML existentes actualmente.

Desventajas de la especificación RELAX NG enmarcadas en el dominio de la presente investigación:

- No soporta PSVI ni posee mecanismos análogos con este.
- Carece de mecanismos formales para la fijación del esquema a un documentos XML (esto fue concebido así de manera intencional debido a razones de seguridad e interoperabilidad).
- Posee un sistema extremadamente simplista para especificar tipos de datos simples, mientras que este es uno de los puntos fuertes en la especificación XSD (también concebido así de manera intencional con el objetivo de lograr especificaciones externas de tipos integrados mayormente asociados al dominio del problema en cuestión).

Uso de Schematron

Schematron forma parte de los lenguajes de marcado XML, como un mecanismo de validación basado en reglas, característica que le permite utilizar expresiones de acceso en lugar de expresiones gramaticales, para definir lo que es válido en un documento XML. Este método de validación aporta gran flexibilidad en la descripción de estructuras relacionales. En cambio, es un lenguaje muy limitado a la hora de especificar la estructura básica del documento, problema que se soluciona combinando Schematron con otros lenguajes de esquema. Junto con RELAX NG, Schematron ha sido estandarizado como parte de la norma ISO/IEC (26).

En su forma típica de implementación, los esquemas Schematron son procesados con código XSLT (por sus siglas en inglés, Extensible Stylesheet Language Transformations). Pudiendo ser utilizados en cualquier situación donde XSLT pueda ser aplicable. Actualmente, XSLT es muy usado en la edición web, generando páginas HTML o XHTML. La unión de XML y XSLT permite separar contenido y presentación, aumentando así la productividad (26).

Según palabras de su propio creador, Rich Jelliffe de la Academia Sinica Computing Centre, de Taiwan: “Schematron es un plumero para llegar a aquellas partes del documento que otros lenguajes no pueden alcanzar”. Expresión que ha sido avalada por su propia obra, a partir de las utilidades que Schematron ha demostrado en la práctica. Muchos editores XML utilizan Schematron para sus reglas condicionales

detectores/resaltadores de errores. También es empleado como motor ligero de reglas empresariales; aunque no es tan exhaustivo como otros motores de reglas (por ejemplo RETE[13]), Schematron puede utilizarse para expresar reglas acerca de complejas estructuras dentro de un documento XML y sobre todo; es muy usado como asistente de validación estructural mediante la comprobación de restricciones de concurrencia, restricciones no-regulares y restricciones inter-documentos, demostrando gran utilidad para extender validaciones expresadas en lenguajes como DTD, RELAX NG o XSD (27).

Fundamentación del estándar de validación a emplear

Luego de analizar potencialidades y desventajas asociadas a los principales mecanismos usados en la práctica para la validación de documentos XML; surge entonces una pregunta fundamental: ¿cuál de ellos emplear? La respuesta a la pregunta anterior puede depender de múltiples factores, los cuales pueden asociarse con: necesidades contextuales del problema en particular que se esté tratando, características de los documentos XML a validar, o puede suceder que, tal decisión esté condicionada por especificidades de los procesos de negocios propios de la organización; como es el caso.

Concretamente, se determinó emplear XSD por varios factores, entre los cuales adquiere especial importancia, funcionalidades como el mencionado PSVI a partir de su relevancia en el contexto del problema a resolver. En este punto, es importante mencionar que un requisito clave en el sistema VUCE es la generación dinámica de formularios a partir del modelo de datos descrito por el esquema de validación de los documentos electrónicos XML; siendo XSD la única especificación entre los lenguajes esquemas disponibles, que soporta tal funcionalidad. Otro factor importante a favor de XSD con respecto a RELAX NG (su principal competidor) es que, a pesar de que ambas especificaciones son contemporáneas; la primera contó con mejor aceptación y fue mayormente documentada y difundida en editores y analizadores sintácticos tanto de código abierto como propietarios. Cuestión que todavía tiene mucha incidencia, independientemente a que RELAX NG se ha ido abriendo camino en el mundo de las tecnologías XML en los últimos tiempos.

1.2. Sistemas automatizados existentes asociados al campo de acción

1.2.1. Oxygen XML Editor

[13] Algoritmo de reconocimiento de patrones eficiente para implementar un sistema de producción de reglas. RETE es hoy en día la base de muchos sistemas expertos, incluyendo: *CLIPS, Jess, Drools, y Soar*, entre otros.

Oxygen es una aplicación multiplataforma disponible en los principales sistemas operativos (Windows, Mac OS, Linux, Solaris), el cual se puede utilizar de forma independiente o como un plugin de Eclipse[14]. Desarrollado por *Syncro Soft Company*, empresa especializada en tecnologías XML fundada en 1998 y miembro del W3C. Desde el punto de vista competitivo, Oxygen es uno de los principales editores XML disponibles actualmente. Su precio oscila alrededor de los 500 dólares americanos. Es una de las pocas herramientas con soporte para todas las tecnologías basadas en XML, incluyendo: lenguajes de esquemas XML, bases de datos XML, tuberías XProc[15], Servicios Web, etc. Posee además, depuradores de gran alcance y perfiladores de rendimiento para Lenguajes de Transformación de Hojas de Estilo Extensibles (XSLT por sus siglas en inglés) y XQuery[16]. Dicha herramienta está subdividida en dos vistas fundamentales para la edición de documentos XML: *XML Developer* y *XML Author*; cada una de las cuales con notables potencialidades (28).

1.2.2. Liquid XML Studio

Liquid XML Studio 2012 (Liquid XML) es un avanzado entorno de desarrollo que integra todas las herramientas necesarias para diseñar y desarrollar proyectos XML, sintetizadas en interfaces intuitivas y funciones completas y extensibles a todo tipo de usuario. Desarrollado por *Liquid Technologies Ltd*, empresa de software propietario con sede en West Yorkshire, Reino Unido, fundada en el año 2000; especializada en la creación de herramientas de productividad avanzada para desarrolladores de software XML, conformes a los estándares del W3C. Su actual producto estrella: Liquid XML, es una herramienta basada en tecnología .NET de Microsoft y proporciona una completa integración con *Microsoft Visual Studio 2005, 2008 y 2010*. En el año 2012 arribó a su versión 10.0 y su precio oscila alrededor de los 800 dólares americanos (29).

1.2.3. Syntext Serna XML Editor

Syntex Serna XML Editor es una herramienta multiplataforma de código abierto para la edición de documentos XML basada en editores WYSIWYG (siglas de la frase inglesa “What You See Is What You Get”), desarrollada por Syntex, Inc. Su última versión estable es la 4.4, liberada en diciembre de 2011.

[14] IDE de código abierto multiplataforma, desarrollado originalmente por IBM y perteneciente actualmente a la Fundación Eclipse. Licenciado bajo la *Eclipse Public License* y basado en Plataforma de Cliente Enriquecido.
[15] Es una recomendación del W3C para definir XML Pipeline (conexión entre procesos de validación y transformación XML).
[16] Lenguaje de consulta diseñado para colecciones de datos XML. Es semánticamente similar a SQL, aunque incluye algunas capacidades de programación.

Dicha herramienta tiene incorporado soporte para estándares como: *W3C XML Schema, DTD, XPath, DITA, DocBook, XHTML, TEI, NITF, MathML, CALS Table, XSL: XSLT, Catálogos XML*, entre otros (30).

Los sistemas automatizados vistos anteriormente constituyen potentes herramientas asociadas al campo de acción de la presente investigación. Estas soluciones garantizan no sólo una eficiente creación/edición de esquemas XML, sino que abarcan también un amplio conjunto de tecnologías XML. Además, poseen novedosos editores tanto textuales como gráficos, provistos de una amplia gama de utilidades que constituyen importantes catalizadores para el desarrollo.

A pesar de que dichas soluciones pueden tomarse como referencia para la implementación de la propuesta de solución que ocupa el presente trabajo de diploma, presentan algunos inconvenientes de cara a especificidades del problema a resolver. El primero de ellos radica en su condición desktop[17], lo cual constituye una importante limitación que afecta el nivel de centralización que se requiere en las acciones de definición y despacho de documentos esquemas por parte de las entidades involucradas en los procesos de comercio exterior. Estas herramientas son incapaces de establecer distinciones entre usuarios (debido a que no fueron concebidas para este propósito), siendo este un requisito clave en el dominio del problema a resolver. Otras limitaciones no menos importantes están relacionadas con el tipo de licencia y con el elevado precio de adquisición, cuestiones que descartarían su utilización, teniendo en cuenta particularidades económicas que presentan no pocos entornos de despliegue en el país.

1.3. Modelo de desarrollo de software

El desarrollo de la propuesta de solución del presente trabajo de diploma, estará guiado metodológicamente por el **Modelo de Desarrollo de Software para el CEIGE** en su versión 1.1, el cual define la secuencia de fases por la que debe transitar todo proyecto bajo su jurisdicción, así como el conjunto de artefactos a generar en cada una de ellas. Tal modelo incorpora los distintos subprocesos dictados por el nivel II de CMMI (del inglés Capability Maturity Model Integration), certificación acreditada al CEIGE en julio de 2011 y reconocida por el SEI (del inglés Software Engineering Institute) como aval de la calidad de su proceso de desarrollo de software (31).

[17] Herramienta instalada en el ordenador del usuario, que es ejecutada directamente por el sistema operativo y cuyo rendimiento depende de diversas configuraciones de hardware como memoria RAM, disco duro, memoria de video, etc.

En cuanto a las características fundamentales asociadas a este modelo de desarrollo cabe destacar que, es ágil y adaptable al cambio, iterativo e incremental; además, propone un desarrollo basado en componentes y centrado en la arquitectura.

1.4. Técnicas de captura y validación de requisitos de software

La obtención de requisitos es el proceso mediante el cual los interesados en un sistema de software descubren, revelan, articulan y entienden sus requisitos. En muchos casos, se requiere tiempo para llegar a especificar claramente lo que el interesado espera de la aplicación de software, por lo que se hace necesario por parte de los analistas el empleo de técnicas que permitan establecer una buena comunicación con los interesados del producto y así lograr la satisfacción del cliente. A continuación se enuncian las principales técnicas utilizadas en la disciplina ingeniería de requisitos para el desarrollo de la solución propuesta (32).

- **Tormenta de ideas:** Reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo consiste en que cada participante exponga su criterio o propuesta. Al finalizar la sesión se debe hacer una recolección de ideas sin duplicidad.
- **Sistemas existentes:** fue utilizada durante el análisis de varias soluciones existentes relacionadas con la que va a ser construida. Enfatizando en características claves que no deben estar ausente en la solución que se propone.

Asegurar la validez de los requisitos es una de las actividades claves que debe llevarse a cabo antes de comenzar el desarrollo de todo producto software, o de lo contrario se corre el riesgo de implementar una inadecuada especificación, que conllevaría a pérdida de tiempo, dinero, o peor; clientes insatisfechos. Debe comprobarse que el modelo obtenido es capaz de responder tanto a los requisitos de negocio como a los requisitos de usuario (33).

Durante la ingeniería de requisitos, resulta de mucha utilidad marcar los objetivos del sistema y chequearlos contra los requisitos identificados, de manera que sea posible visualizar qué objetivos cubre cada requisito con el propósito de detectar inconsistencia y objetivos no cubiertos. Esta técnica se conoce como validación de requisitos a través de matriz de trazabilidad. Otra técnica de validación de requisitos, consiste en construir prototipos con una funcionalidad similar muy reducida, para que el cliente se haga

una idea aproximada del resultado final; lo cual es indispensable para llegar a acuerdos firmados en cuanto a lo que se va a hacer, antes de empezar la construcción del producto (33).

Las Revisiones Técnicas Formales (RTF) sobre los artefactos generados durante la especificación de los requisitos de software, constituye una técnica de validación de requisitos muy utilizada en la práctica. Esta consiste en las revisiones realizadas por el usuario final conjuntamente con especialistas del proyecto, sobre las especificaciones de los requisitos definidos, lo cual permite detectar deficiencias, ambigüedades, omisiones o errores de otra índole (33).

1.5. Patrones de arquitectura y diseño

Los patrones de diseño surgen ante la necesidad reutilizar soluciones probadas en escenarios comunes durante el diseño de un software. Con la reutilización se consigue la reducción de tiempos y la disminución del esfuerzo de mantenimiento, esto trae consigo mayor eficiencia y consistencia en el diseño de la solución (34). A continuación se ilustran los principales patrones de arquitectura y diseños utilizados en la propuesta de solución.

Modelo Vista Controlador (MVC)

La esencia del patrón de diseño MVC, consiste en separar los datos y la lógica de negocio de una aplicación, de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones; mediante la construcción de tres componentes distintos: el Modelo, la Vista y el Controlador. La arquitectura resultante del uso de este patrón, favorece a indicadores como: reutilización de código, mantenibilidad, escalabilidad; y demás factores directamente proporcionales a la calidad de software (34).

De manera genérica, los componentes de MVC se pueden definir como sigue:

- **Modelo:** Es la representación específica de la información con la cual el sistema opera. Por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la *Vista* aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al *Modelo* a través del *Controlador*.

- **Controlador:** Responde a eventos (usualmente acciones del usuario) e invoca peticiones al *Modelo* cuando se hace alguna solicitud sobre la información. También puede enviar comandos a la *Vista* asociada, si se solicita un cambio en la forma en que se presenta de *Modelo*. Por tanto, se podría decir que el *Controlador* juega un papel de intermediario entre la *Vista* y el *Modelo*.
- **Vista:** Presenta al *Modelo* (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho *Modelo* la información que debe representar como salida.

Patrón Observador u Observador/Observado

Este patrón es muy utilizado en la implementación de una arquitectura MVC. La capa del *modelo* es un subtipo de observado y la capa de la *vista* un subtipo de observador. Normalmente se implementan como dos clases que manejan adecuadamente la función de notificación de cambios que necesita MVC, pues proporciona el mecanismo por el cual, las vistas pueden ser notificadas automáticamente de los cambios producidos en el modelo. Permite gestionar la relación entre un componente observado y sus observadores, pudiendo un componente determinado comportarse de ambas formas, es decir, cada componente observado posee una referencia a todos sus observadores, así como los servicios que están siendo consumidos por estos (34).

El patrón de diseño **Observador/Observado** es aplicable cuando:

- Existe una relación fuerte entre datos y vistas, de manera que conviene separar el control de los datos de su representación final.
- Un cambio en el estado de un determinado componente afecta a muchos otros.
- Se necesita notificar a otros componentes sin hacer presunciones sobre la identidad y ubicación física, evitando un fuerte acoplamiento lo cual ayuda a diseñar sistemas con diferentes capas de abstracción claramente separadas.
- Se necesitan sincronizar coherentemente las acciones de varios componentes sobre un estado común.

Patrones GRASP

Bajo acoplamiento: es la idea de tener las clases lo menos ligadas entre sí que se pueda, de forma tal que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases (35).

Alta Cohesión: la alta cohesión al igual que el bajo acoplamiento logran un diseño fácil de reutilizar y adaptar, más legibilidad para los programadores y diseñadores en general; la extensibilidad y flexibilidad del diseño aumentan, ya que las clases usan solo lo que necesitan y en su mayoría son recursos propios. El objetivo es lograr que cada clase haga lo que le corresponde hacer según la parte del problema que está modelando (35).

Experto: es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento) (35).

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento (35).

1.6. Tecnologías y herramientas a utilizar para el desarrollo del componente

El marco tecnológico sobre el cual se sustenta el desarrollo de la propuesta de solución, está condicionado por decisiones arquitectónicas del proyecto VUCE. En lo adelante se describe el mismo de manera general, enfatizando en características individuales de las herramientas y framework utilizados, agrupados por tipo de actividad.

1.6.1. Actividades ingenieriles

Lenguaje Unificado de Modelado (UML)

El lenguaje unificado de modelado (UML por sus siglas en inglés) es un “lenguaje” para especificar, construir, visualizar y documentar los artefactos[18] de un sistema de software orientado a objetos (OO). Captura decisiones y conocimiento sobre los sistemas que se deben construir, generando modelos que puedan ser comprensibles para todas las partes involucradas en un proyecto de software. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información que se maneja en tales proyectos. Pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios, incluyendo conceptos semánticos, notación y principios generales. UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño OO. Contiene construcciones organizativas para agrupar los modelos en paquetes, permitiendo a los equipos de software dividir grandes sistemas en piezas de trabajo, para entender y controlar las dependencias entre paquetes y para gestionar las versiones de las unidades del modelo en un entorno de desarrollo complejo. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios (36).

Notación para el Modelado de Procesos de Negocio (BPMN)

La notación para el modelado de procesos de negocio (BPMN de sus siglas en Inglés Business Process Modeling Notation) brinda grandes ventajas para modelar el negocio fundamentalmente en base a la gestión por procesos. BPMN, define diagramas de procesos de negocios basados en la técnica de diagramas de flujo adaptados para graficar las operaciones de los procesos de la organización. Se compone de un conjunto de elementos gráficos que facilitan un diagrama entendible tanto por audiencias de negocio como por técnicos (37).

Los elementos gráficos se dividen en cuatro categorías:

- Objetos de flujo (eventos, actividades, rombos de control).
- Objetos de conexión (flujo de secuencia, flujo de mensaje, asociación).
- Carriles de piscina.
- Artefactos (objetos de datos, grupos).

[18] Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.

UML se enfoca al diseño de software mientras que BPMN se enfoca a los procesos de negocios. A partir de esto, esta notación es una propuesta a lo que plantea RUP para el modelado de negocio, o sea, cambiar las propuestas tradicionales de modelos de dominio o casos de uso, por una concepción de modelación de procesos a partir de los procesos elementales que la componen (37).

Visual Paradigm 6.1

Visual Paradigm es una de las herramientas UML CASE[19] del mercado, considerada como muy completa y fácil de usar, con soporte multiplataforma y que proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Fue creada para el ciclo vital completo del desarrollo de software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación. Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de las clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento OO, además apoya los estándares más recientes de las notaciones de Java y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros (38).

1.6.2. Actividades de desarrollo

Symfony 2.2

Symfony es un completo marco de trabajo diseñado para optimizar el desarrollo de aplicaciones web, patrocinado por la compañía Web francesa *Sensio Labs*, quién liberó su código en octubre de 2005 bajo políticas de código abierto (39). Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación, lo cual posibilita aumentar la productividad. Está escrito completamente en PHP 5.3 y tiene soporte para los gestores de bases de datos más utilizados como son: MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, pudiéndose ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows (40).

[19] Siglas en ingles de **C**omputer **A**ided **S**oftware **E**ngineering, o traducida al español: Ingeniería de Software Asistida por Computadora.

JQuery 2.0

La capa de la interfaz de usuario de la propuesta de solución está basada en esta fabulosa librería Javascript en su versión 2.0, la cual permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

JQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos (41). Provee una capa de abstracción al Javascript nativo, ofreciendo una serie de funcionalidades que constituyen importantes catalizadores para el desarrollo del lado del cliente. JQuery se destaca por su ligereza y versatilidad para crear aplicaciones web dinámicas, provisto de un amplio conjunto de efectos y animaciones que estimulan la creatividad, lo cual favorece el desarrollo de sistemas web provistos de una fabulosa experiencia de usuario. Además, es compatible con los principales navegadores web: Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+ (42).

Twitter Bootstrap 2.2

Bootstrap es un potente framework CSS desarrollado a partir de una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Creado y auspiciado por Twitter y liberado en agosto de 2011 como una herramienta de código abierto. Contiene plantillas de diseño basadas en HTML y CSS con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript. Bootstrap es una solución flexible que ofrece mucha más potencia que el CSS normal. Tiene una estructura modular y consistente que se basa esencialmente en una serie de hojas de estilo LESS, lo cual permite el uso de variables, funciones y operadores, selectores anidados así como clases mixin; siendo esta la arquitectura que soporta la variedad de componentes de la herramienta. Desde la versión 2.0 soporta características tan innovadoras como diseños sensibles: esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles, etc.) (43).

1.7. Pruebas de software

Hay dos maneras de probar cualquier producto construido (y casi cualquier cosa): 1) si se conoce la función específica para la que se diseñó el producto, se aplican pruebas que demuestran que cada función es plenamente operacional, mientras se buscan los errores de cada función; 2) si se conoce el funcionamiento interno del producto, se aplican pruebas para asegurar que “todas las piezas encajan”; es decir, que las operaciones internas se realizan de acuerdo con las especificaciones y que se ha probado todos los componentes internos de manera adecuada. Al primer enfoque de prueba se le denomina prueba de caja negra; al segundo prueba de caja blanca (33).

1.7.1. Pruebas de caja blanca

La prueba de caja blanca es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Los métodos de pruebas de caja blanca permiten derivar casos de pruebas que: 1) garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez; 2) ejerciten los lados verdadero y falso de todas las decisiones lógicas; 3) ejecuten todos los bucles en sus límites y dentro de sus límites operacionales; y 4) ejerciten estructuras de datos internos para asegurar su validez (33).

Asociada a la prueba de caja blanca existen varias técnicas de prueba; entre las principales: Prueba del Camino Básico, Prueba de Condición, Prueba de Flujo de Datos y Prueba de Bucles (33). La prueba de caja blanca empleada para verificar la solución desarrollada fue la Prueba del Camino Básico, la cual permite obtener una medida de la complejidad lógica de un diseño procedimental como guía para obtener un conjunto básico de rutas de ejecución que garanticen ejercitar cada sentencia del programa por lo menos una vez.

1.7.2. Pruebas de caja negra

Las *pruebas de caja negra,* también denominada *prueba de comportamiento,* se concentran en los requisitos funcionales del software; es decir, permite derivar un conjunto de condiciones de entrada que ejercitarán por completo todos los requisitos funcionales de un programa. Estas pruebas tratan de encontrar errores en las siguientes categorías: 1) funciones incorrectas o faltantes, 2) errores de interfaz, 3) errores en estructuras de datos o en acceso a bases de datos externas, 4) errores de comportamiento o desempeño, y 5) errores de inicialización y término (33).

En la práctica existen varios métodos a través de los cuales se pueden realizar pruebas de caja negra, entre ellos: Métodos gráficos de pruebas, Prueba de tabla ortogonal; así como Partición equivalente y Análisis de valores límite (33), siendo éstos dos últimos, los métodos empleados para el caso que ocupa al presente trabajo de diploma.

La *partición equivalente* es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. El diseño de casos de pruebas para este método se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición booleana (33).

Debido a que el número de objetos de datos del dominio de entrada se torna excesivamente grande para los requisitos funcionales asociados a la solución propuesta, se acudió además al método *Análisis de valores límite.*

El análisis de valores límites (AVL) es una técnica de diseño de casos de prueba que complementa la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, AVL lleva a la selección de casos de prueba en las “aristas” de la clase (33).

Las AVL se definen de acuerdo a las siguientes directrices:

1. Si una condición de entrada especifica un rango limitado por los valores a y b, los casos de pruebas deben diseñarse con esos valores, además de los que se encuentran apenas arriba y debajo de ellos.
2. Si una condición de entrada especifica diversos valores, deben desarrollarse casos de pruebas que ejerciten los valores máximo y mínimo. También se pruebas los valores ubicados apenas abajo y arriba de estos máximos y mínimos.
3. Si la estructura interna de datos del programa tiene límites prescritos (por ejemplo, una matriz que tiene un límite definido de cien entradas) debe diseñarse casos de prueba para ejercitar los límites de la estructura de datos.

Conclusiones parciales

En este capítulo se han expuesto los fundamentos teóricos conceptuales que constituyen la base para la correcta concepción, diseño e implementación de la solución propuesta. Del marco conceptual se puede observar la estrecha relación que existe entre los conceptos de VUCE, EDI, XML y XSD, en función del establecimiento de estrategias de facilitación comercial y aduanera en el país. Se estudiaron las principales soluciones de software disponibles en el mercado asociadas al campo de acción, llegando a la conclusión de que indicadores como: entorno de operatividad, tipo de licencia y precio, descartan su utilización en la mayoría de los entornos de despliegue del país, con lo cual se justifica la construcción de la solución propuesta. Otro criterio importante al que se arribó, es que se utilizará el modelo de desarrollo definido para el CEIGE, como guía metodológica en la construcción del componente.

Capítulo 2. Características del sistema

Introducción

El presente capítulo tiene como objetivo fundamental brindar una panorámica general acerca del problema que se pretende resolver; así como abordar las características fundamentales asociadas a la solución que se propone, con el objetivo de ofrecer una primera aproximación al resultado obtenido.

En función de lo anterior, en un primer momento se establece un contraste entre la vista interna del campo de acción, de cara a los objetivos estratégicos que persigue la VUCE como sistema informático de gobierno electrónico, donde se evidenciaron dos procesos claves: Creación-despacho-recepción XSD y Gestión XSD. Tales procesos fueron modelados y analizados desde un enfoque crítico, donde se obtuvo un catálogo de deficiencias, a partir de las cuales emergieron posibles soluciones. Un segundo momento estuvo dedicado más específicamente a la propuesta de solución, donde se expuso el catálogo de requisitos funcionales y no funcionales con los que debe cumplir la solución informática, de cara a las deficiencias detectadas en el campo de acción. Sirva este capítulo para realizar una correcta interpretación tanto del incentivo como de la necesidad que demanda la realización del presente trabajo.

2.1. Modelo de negocio

2.1.1. Procesos de negocio de nivel 0

Creación-Despacho-Recepción XSD

Cada entidad es experta en la información relacionada con la estructura y restricciones con la que debe cumplir la información electrónica que le compete, dentro del conjunto de información que se manipula en el proceso EDI; motivo por el cual, es su responsabilidad definir y despachar con la VUCE los XSD validadores de sus documentos electrónicos XML. Cada una de estas entidades se gestiona individualmente a la hora de definir tales XSD, de acuerdo a sus posibilidades tecnológicas y nivel de dominio en este tema. No obstante, e independientemente a la diversidad de mecanismos adoptados para este propósito, es posible identificar un conjunto de tareas inherentes a este proceso, que constituyen un denominador común entre cada una de estas entidades.

El elemento de salida del proceso de creación de XSD es un fichero con extensión *xsd*, el cual contiene el código del esquema XML creado. Una vez obtenido el fichero XSD, éste es enviado al repositorio central de fichero XSD ubicado en la VUCE, a la vez que una copia del mismo es archivada en el repositorio local de la propia entidad. Cada entidad posee, por razones de seguridad, trazabilidad y contingencia ante fallas, un repositorio interno de ficheros XSD, los cuales deben estar sincronizados con la información presente en el repositorio central de la VUCE, de manera que las acciones de gestión que se lleven a cabo sobre los esquemas internos, debe reflejarse con carácter inmediato en el mencionado repositorio central.

El mecanismo que permite la sincronización entre los repositorios local y central, está basado en reportes electrónicos. Los reportes electrónicos están conformados por un documento XML que posee una estructura predefinida, en el cual se especifica cómo debe ser procesado en el repositorio central, la "imagen" del fichero XSD manipulado localmente en la entidad. En el caso particular de las acciones de creación de XSD, lo que se hace es crear una imagen del nuevo fichero en el repositorio central.

Una vez que el reporte electrónico llega a la VUCE, el subproceso de recepción tiene asignado entre sus tareas, la autenticación de la entidad emisora; cuyo elemento de salida es la obtención propiamente dicha del reporte electrónico XML. Tal elemento de salida, constituye el elemento de entrada del proceso encargado de ejecutar las acciones de gestión especificadas en el mencionado reporte electrónico. Acciones que, puede consistir tanto en la creación de un nuevo fichero XSD, como en la modificación, reemplazo o supresión de uno ya existente. No obstante, en cualquier caso, el proceso se lleva a cabo de manera automatizada, incluso desde el momento en punto en que se recepciona el reporte electrónico, luego de la correcta autenticación de la entidad emisora.

Una vez ejecutadas las acciones de gestión especificadas en el reporte electrónico, el resultado final debe consistir en la correcta sincronización de los ficheros XSD presentes en los repositorios local y central, generándose las notificaciones correspondientes en cada paso.

Lo anteriormente expuesto se resume en el siguiente diagrama de procesos de negocio:

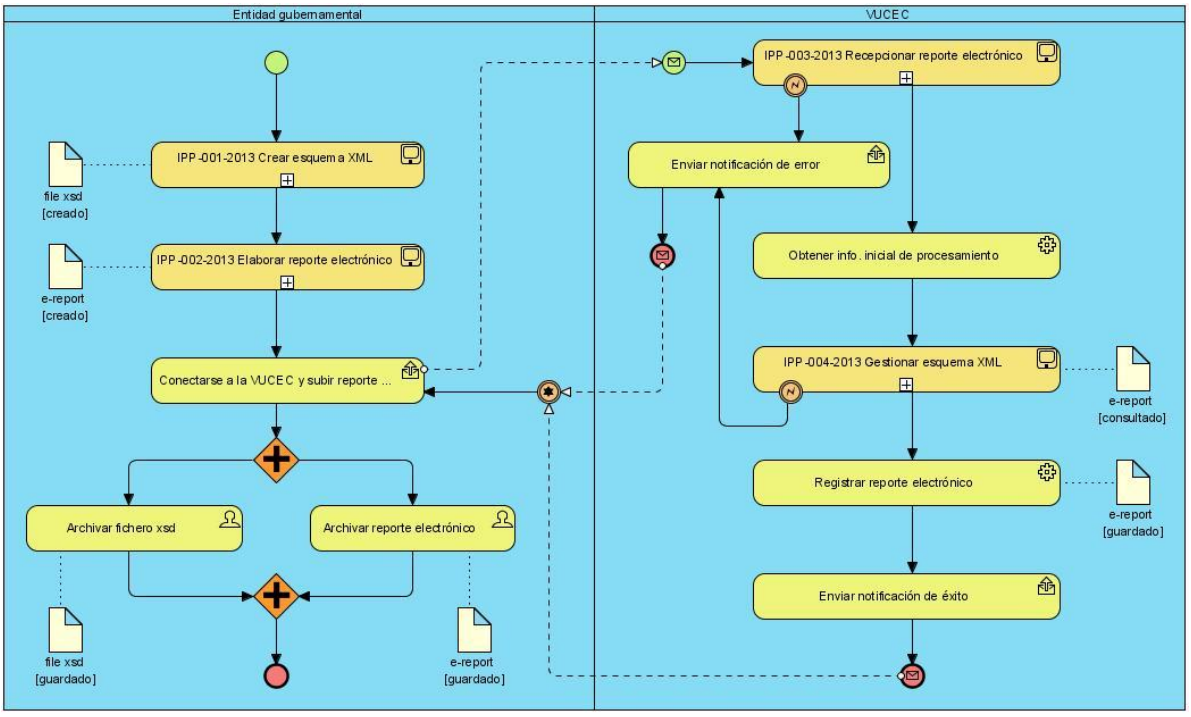


Diagrama 2.0. Modelo de negocio correspondiente al proceso de Creación-Despacho-Recepción de XSD.

Gestión XSD

Los procesos de gestión de esquemas XML, están relacionados con las acciones de actualización, reemplazo o supresión de ficheros XSD que se llevan a cabo en el repositorio interno de cada entidad y posteriormente son reflejadas en el repositorio central de ficheros XSD ubicado en la VUCE a través de reportes electrónicos, como se explicó anteriormente. El modelo de negocio que describe al proceso de Gestión XSD, se refleja en el siguiente diagrama:

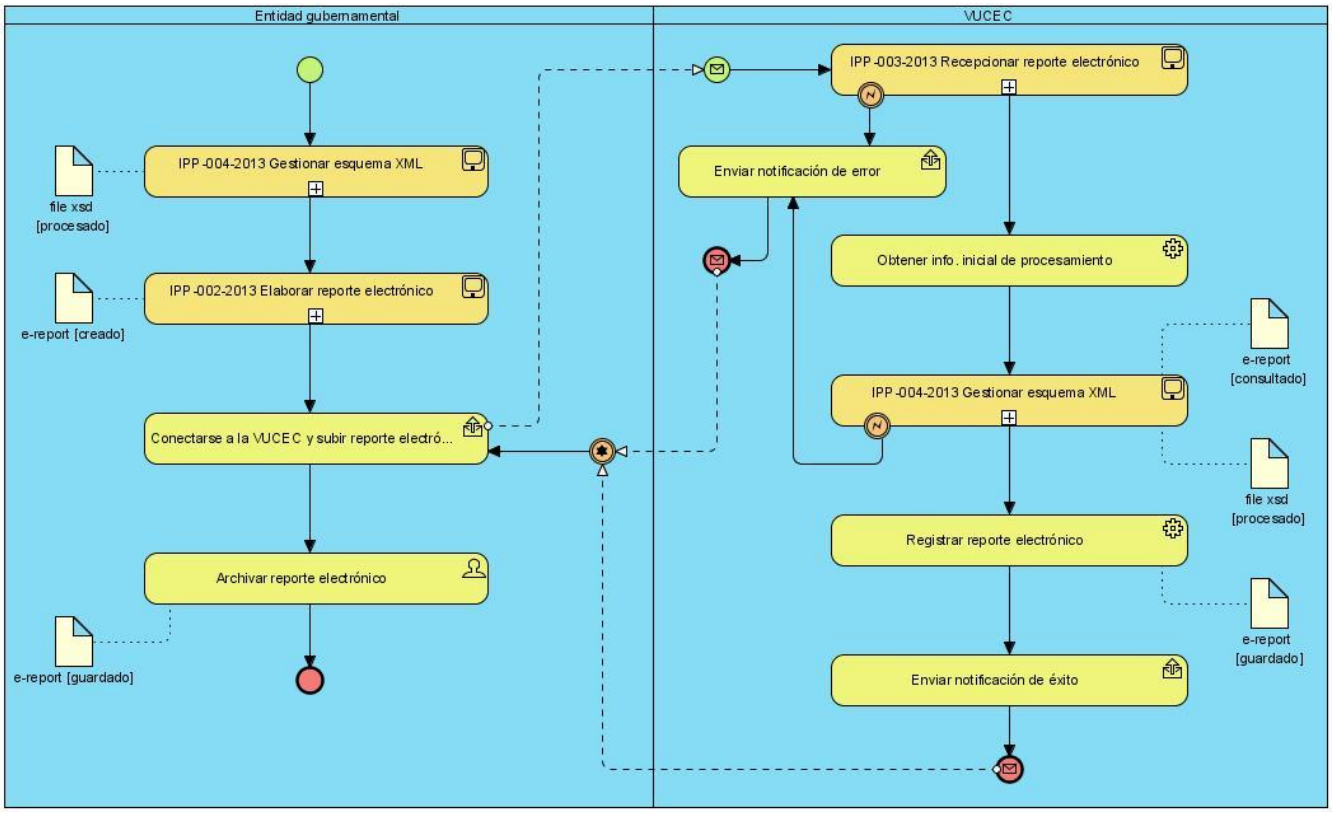


Diagrama 2.1. Modelo de negocio correspondiente al proceso de Gestión XSD.

2.2. Situación problémica, problema y objeto de automatización

2.2.1. Valoración crítica de los procesos de negocio presentes en el campo de acción

Problema:	Heterogeneidad en los procesos de definición y gestión de esquemas XML.
Causas:	

- Desnivel en cuanto a soporte tecnológico y dominio del lenguaje *W3C XML Schema* de cara a los procesos de definición y gestión de XSD, existente entre las entidades implicadas en el proceso de comercialización.
- La obtención de la totalidad de los XSD necesarios para el proceso de validación está sujeto a diversas variables, teniendo en cuenta que, cada XSD proviene de una ubicación diferente.
- Procesamiento redundante en la definición y gestión de esquemas XML debido a la ineficiente

distribución de responsabilidades entre los actores implicados en el negocio.

- Ineficiente sincronización entre los repositorios locales de las entidades gubernamentales y el repositorio central de la VUCE, teniendo en cuenta el bajo nivel de automatización inherente a esta tarea.
- Introducción de nuevos errores en los procesos de definición y gestión de esquemas XML a partir de la existencia de un intermediario *(e-report)* en la comunicación entidad-sistema.

Consecuencias:	

- El proceso de validación de documentos XML en ocasiones no cuenta con la información requerida en el momento preciso, lo que conlleva a la aparición de errores en etapas avanzadas del proceso de comercialización, donde su corrección implica redundancia de transacciones y retrasos innecesarios que provocan pérdida de tiempo y dinero; todo lo cual, atenta considerablemente contra la confiabilidad, agilidad y eficiencia del proceso de comercialización en su conjunto.
- Sobrecarga de procesamiento en la definición y gestión de esquemas XML, a partir de lo cual, puede verse comprometido la capacidad de respuesta del sistema, o la agilidad requerida en determinadas situaciones de elevados flujos de transacciones.
- La desincronización entre el repositorio local de determinada entidad y el repositorio central de la VUCE, produce errores cíclicos en las acciones de gestión sobre el(los) fichero(s) XSD no sincronizado(s), lo cual atenta considerablemente contra la fluidez y el nivel de automatización de los procesos de gestión.

Directrices de solución	

- Factorizar el procesamiento dedicado a las acciones de definición y gestión de XSD.
- Establecer un conjunto de herramientas, disponibles en el mismo sistema VUCE, que permitan el procesamiento en línea de ficheros XSD, como alternativa al mecanismo de reporte electrónico para la comunicación entidad-sistema.
- Buscar alternativas al mecanismo de repositorio imagen como plan de contingencia ante fallas.

2.2.2. Cambios introducidos a partir de la solución propuesta

A partir de los procesos de negocio antes descritos y teniendo en cuenta los inconvenientes que sobre estos se identificaron, se introducen los siguientes cambios:

1. Se sustituyen los procesos de gestión interna (en las entidades) de ficheros XSD, por la gestión directa en el sistema VUCE. Lo cual implica la desaparición de los repositorios internos de las entidades; y por ende, la desaparición del mecanismo de sincronización basado en reportes electrónicos.
2. Se establece en el sistema VUCE un único espacio de trabajo dedicado a los procesos de definición y gestión de ficheros XSD, que se adapta en función del usuario Online. Dicho espacio de trabajo está subdividido en tres subcomponentes:
    - Editor XSD: herramienta para la definición y edición de ficheros XSD.
    - File-upload: herramienta para la recepción de ficheros XSD.
    - File-browser: herramienta para la gestión de los ficheros XSD presentes en el repositorio central de la VUCE.
3. Se introducen opciones de salvas de datos (backup) y puntos de restauración, como plan de seguridad y contingencia ante fallas.
4. Se factorizan los procesos de creación-despacho de ficheros XSD en un único componente: Editor XSD Online; aunque no es la única variante de despacho que se brinda.

De manera que, los procesos de negocio de Creación-Despacho-Recepción XSD y Gestión XSD se llevarían a cabo tal y como se muestra en el siguiente diagrama de procesos de negocio.

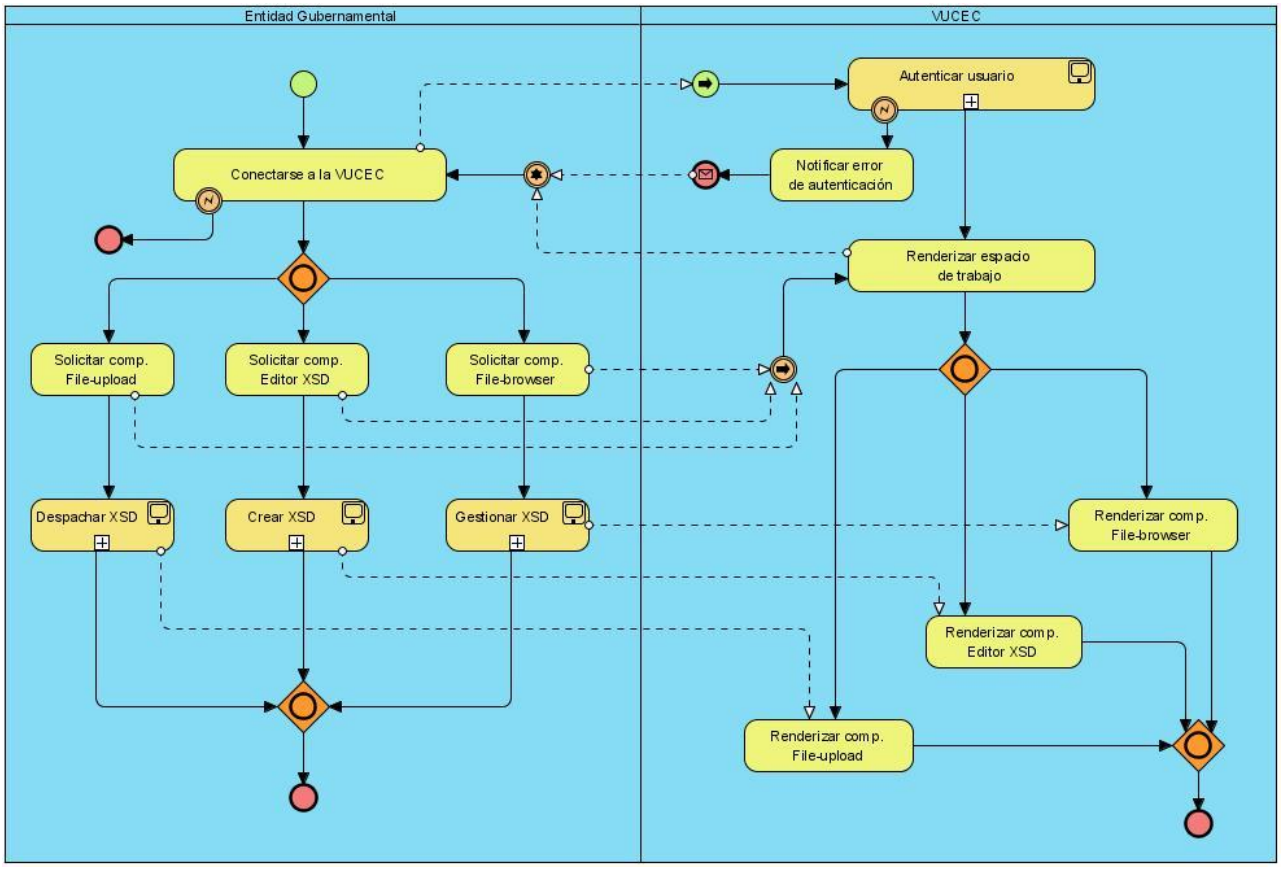


Diagrama 2.2. Modelo de negocio correspondiente a la solución que se propone.

2.3. Propuesta de sistema

2.3.1. Especificación de requisitos funcionales

El IEEE Standard Glossary of Software Engineering Terminology, define los requisitos de software como condiciones o capacidades que deben estar presentes en un sistema o componentes de este, para satisfacer un contrato, estándar, especificación u otro documento formal. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea, debido a que desempeña un papel primordial en el proceso de producción de software. Los requisitos funcionales se enfocan en un área fundamental: la definición de lo que se desea producir, permitiendo describir detalladamente el comportamiento del sistema; lo cual permite minimizar los problemas derivados de su desarrollo.

La totalidad de los requisitos funcionales que posee la solución, se pueden consultar en el **anexo 2**. No obstante, para una mejor organización y comprensión de los mismos, se definieron las siguientes agrupaciones de requisitos funcionales de acuerdo con los intereses u objetivos del negocio, los cuales se relacionan a continuación:

- *RF_1. Crear Esquema XML en línea.*
- *RF_2. Importar fichero XSD.*
- *RF_3. Exportar fichero XSD.*
- *RF_4. Despachar XSD.*
- *RF_5. Gestionar fichero XSD.*

Especificación del requisito *Crear Esquema XML en línea*

RF_1	Crear esquema XML en línea.
Precondiciones	El actor está debidamente autenticado en el sistema.
Flujo de eventos	
Flujo básico Crear Esquema XML en línea	
1. El sistema visualiza el Editor XSD con el tag ***<xs:schema></xs:schema>*** ubicado como raíz del esquema gráfico.	
2. Se selecciona el tag que se desea procesar (en la primera iteración sólo existe la opción del tag raíz) o se selecciona la opción *Exportar fichero XSD.* En caso de seleccionar la opción *Exportar fichero XSD,* se ejecuta el paso 1 del Flujo básico de eventos del requisito *Exportar Fichero XSD*.	
3. El sistema despliega un menú contextual donde se listan las opciones disponibles sobre el tag seleccionado (tag base).	
4. Se selecciona la opción *Adicionar, Configurar, Reemplazar, Eliminar o Eliminar árbol de tag.* - Si se selecciona la opción *Configurar,* ver flujo alternativo 4.a *Configurar tag*. - Si se selecciona la opción *Reemplazar,* ver flujo alternativo 4.b *Reemplazar tag*. - Si se selecciona la opción *Eliminar,* ver flujo alternativo 4.c *Eliminar tag*. - Si se selecciona la opción *Eliminar árbol de tag,* ver flujo alternativo 4.d *Eliminar árbol de tag*.	

5. El sistema lista en una ventana emergente los tags (tags candidatos) que es posible adicionar al esquema gráfico como hijos del tag seleccionado (tag base).
6. Se especifica el tag que se desea incluir al esquema gráfico y se selecciona la opción *Añadir o Añadir y configurar.*

   Si se selecciona la opción *Añadir y configurar,* ver flujo alternativo 6.a *Añadir y configurar tag.*
7. El sistema adiciona el tag especificado al esquema gráfico.
8. Se ejecuta el paso 2 tantas veces se considere necesario.
9. Finaliza el requisito.

Pos-condiciones.

1. El sistema genera el esquema XML correspondiente con el esquema gráfico en proceso.

Flujos alternativos

Flujo alternativo 4.a Configurar tag

1. El sistema lista los atributos compatibles con el tag en proceso, señalando de manera predeterminada la configuración que éste trae por defecto (configuración base).
2. Se seleccionan los atributos que se desean incluir al tag en proceso, especificando en cada caso el valor correspondiente o se desmarca(n) el(los) atributo(s) que se desea(n) eliminar de la configuración base; y se selecciona la opción *Aceptar* o *Cancelar*.

   Si se selecciona la opción *Cancelar*, se ejecuta el paso 2 del flujo básico.
3. El sistema actualiza en el esquema gráfico el tag en proceso con la nueva configuración.
4. Se ejecuta el paso 2 del flujo básico.

Pos-condiciones.

1. Se actualiza en el esquema gráfico atributos y/o valores especificados en el tag base.

Flujo alternativo 4.b Reemplazar tag

1. El sistema lista en una ventana emergente los tags con los que es posible reemplazar el tag especificado (tag base), teniendo en cuenta la posición que ocupa el tag base en la jerarquía del esquema XML.
2. Se especifica el tag (tag candidato) con el que se desea reemplazar el tag base.
3. El sistema lista los atributos compatibles con el tag candidato.
4. Se especifica(n) y/o configura(n) el(los) atributo(s) que se desea(n) incluir al tag candidato, o directamente se selecciona la opción *Reemplazar* o *Cancelar*.

Si se selecciona la opción *Cancelar*, se ejecuta el paso 2 del flujo básico.

Si se selecciona la opción *Reemplazar,* se ejecuta el paso 7 del presente flujo alternativo 4.b Reemplazar tag.

5. El sistema adiciona al tag candidato los atributos y valores especificados.
6. Se selecciona la opción *Reemplazar* o *Cancelar*.

   Si se selecciona la opción *Cancelar*, se ejecuta el *Paso 2 del Flujo básico.*
7. El sistema reemplaza el tag base con el tag candidato.
8. Se ejecuta el *Paso 2 del Flujo básico.*

Pos-condiciones.

1. Se actualiza el esquema gráfico con los cambios realizados.

Flujo alternativo 4.c Eliminar tag

1. El sistema comprueba que el tag especificado es "hoja" *(no tiene tags hijos)* en la jerarquía del esquema XML creado hasta el momento.

   Si el tag seleccionado no es hoja, ver flujo alternativo 10.a *Validar Supresión Individual*.
2. El sistema elimina el tag seleccionado del esquema gráfico.
3. Se ejecuta el paso 2 del flujo básico.

Pos-condiciones.

1. Se actualiza el esquema gráfico con los cambios realizados.

Flujo alternativo 4.d Eliminar árbol de tag

1. El sistema elimina de manera recurrente los tag hijos del tag especificado, incluyendo a este último.
2. Se ejecuta el *paso 2 del flujo básico.*

Pos-condiciones.

1. Se actualiza el esquema gráfico con los cambios realizados.

Flujo alternativo 6.a Añadir y Configurar tag

1. El sistema adiciona al esquema gráfico el tag especificado.
2. Se ejecuta el paso 1 del flujo alternativo 4.a *Configurar tag.*

Pos-condiciones.

1. Se actualiza el esquema gráfico con los cambios realizados.

Flujo alternativo 10.a Validar supresión individual

1. El sistema verifica la validez estructural del esquema resultante.

   Si el esquema resultante no es válido, se ejecuta el paso 1 flujo alternativo 11.a *Acción denegada*.

2. El sistema elimina el tag especificado.
3. Se ejecuta el paso 2 del flujo básico.

Pos-condiciones.

1. El sistema actualiza la estructura del esquema en proceso.

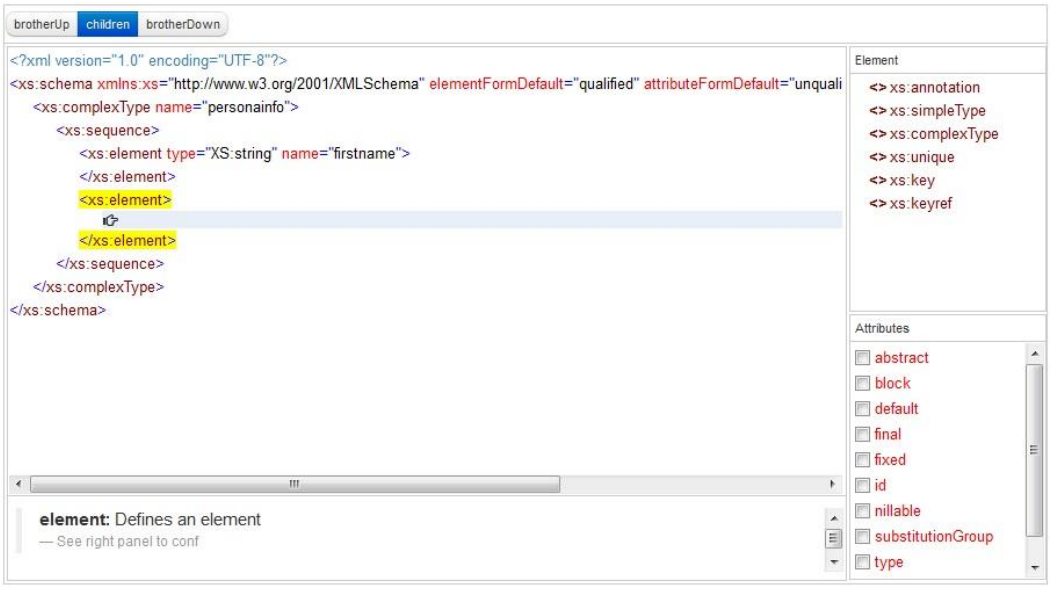
Flujo alternativo 11.a Acción denegada.

1. El notifica al actor que el tag especificado no puede ser eliminado y se brinda la opción de Eliminar árbol de tag a partir del tag base.
2. Se selecciona la opción *Eliminar árbol de tag* o *Cancelar.*

   Si se selecciona la opción *Eliminar árbol de tag,* se ejecuta el paso 1 del flujo alternativo 4.d *Eliminar árbol de tag*.

3. Se ejecuta el paso 2 del Flujo básico.

Prototipo elemental de interfaz gráfica



(Nota: Consular el resto de los artefactos en la plantilla *Especificación de Requisitos*, adjunta en los entregables del presente trabajo de diploma**)**.

2.3.2. Especificación de requisitos no funcionales

Los requisitos no funcionales son aquellos requisitos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este. Surgen a partir de las necesidades del usuario que pueden estar dadas por: restricciones de presupuesto, políticas de la organización, necesidades de interoperabilidad con otros sistemas software o hardware; o a factores externos como regulaciones de seguridad o legislaciones sobre privacidad (44).

A menudo suele pensarse en los requisitos no funcionales como atributos de calidad que hacen al producto atractivo, usable, rápido o confiable. No son parte de la razón fundamental del producto, pero si son necesarios para hacer funcionar el producto de la manera deseada.

A continuación se exponen los requisitos no funcionales (o atributos de calidad) presentes en la propuesta de solución, agrupados por categorías:

Usabilidad

RNF_1: El componente debe proporcionar un ambiente RIA[20] en la interacción con el usuario.

RNF_2: El componente debe poseer interfaces intuitivas que garanticen a usuarios no experimentados una completa maniobrabilidad luego de 10 minutos de entrenamiento.

RNF_3: El componente deberá brindar una descripción funcional en forma de ayuda sobre cada subcomponente individual.

RNF_4: El subcomponente *Editor XSD Online* debe asistir al usuario en todo momento durante el proceso de definición de esquema XML.

RNF_5: El subcomponente *Editor XSD Online* debe garantizar la validez de los esquemas XML que mediante éste se definan.

[20] Del inglés *"Rich Internet Applications",* se refiere a sistemas web que ofrecen una interacción con el usuario, análoga con el ambiente que brindan las aplicaciones de escritorio.

RNF_6: El subcomponente *Editor XSD Online* debe permitir al usuario realizar varios pasos de las acciones deshacer/rehacer.

RNF_7: El subcomponente *File-upload* debe notificar claramente el resultado de las operaciones realizada sobre cada fichero XSD en proceso.

Rendimiento

RNF_8: El tamaño máximo en memoria de los recursos que carga el navegador para el funcionamiento del componente, no debe sobrepasar de 1MB.

RNF_9: El tiempo total a consumir por el navegador para la carga total del componente, no debe sobrepasar de 1.7 segundos (estando el servidor web localmente).

RNF_10: El tiempo total a consumir al importar/exportar un fichero XSD, no debe sobrepasar de 500ms.

Restricciones de diseño e implementación

RNF_11: El componente se implementará sobre un *bundle* del framework PHP Symfony en su versión 2.2

RNF_12: El formateo visual de los elementos HTML estará basado en el framework CSS Bootstrap en su versión 2.2

RNF_13: La codificación JavaScript estará basada en la librería jQuery en su versión 2.0

RNF_14: El desarrollo de la solución estará guiado metodológicamente por el Modelo de Desarrollo para el CEIGE.

Seguridad

RNF_16: El subcomponente *File-browser* sólo debe mostrar al usuario en línea, los fichero XSD correspondientes con la entidad a la cual representa.

RNF_17: La interfaz de configuración del núcleo del subcomponente *Editor XSD Online*, sólo debe estar disponible para el administrador del componente.

Fiabilidad

RNF_18: El componente debe permitir al usuario crear puntos de restauración de la información concerniente a la entidad a la cual representa.

RNF_19: El componente debe permitir al administrador crear salvas totales de la información presente en el repositorio centra de archivos XSD, como plan de contingencia ante fallas.

Software

RNF_20: El componente debe garantizar compatibilidad con los siguientes navegadores web: Mozilla Firefox 2.0+, Internet Explorer 6+, Safari 3+, Opera 10.6+ y Google Chrome 8+.

Interfaz

RNF_21: El aspecto visual del componente debe estar basado en colores legados del sistema VUCE.

2.4. Validación de requisitos

Los requisitos identificados en la presente investigación, fueron validados en un primer momento a través de la técnica matriz de trazabilidad (ver **anexo 3**), donde no se obtuvieron inconsistencias ni objetivos no cubiertos. Una segunda prueba de validación se realizó mediante Revisión Técnica Formal (RTF) sobre los artefactos por parte de especialistas del proyecto, donde se detectaron un total de 5 no conformidades que consistieron en ambigüedades y redundancias funcionales entre requisitos, las cuales fueron solucionadas para la segunda revisión que resultó ser satisfactoria. Otro tipo de prueba que se realizó sobre los requisitos definidos, consistió en la realización de prototipos básicos funcionales. En este último, se presentaron los prototipos elaborados durante la especificación a especialistas del proyecto, para corroborar su correspondencia con las necesidades y aspiraciones del cliente[21]. Para ello, se desarrollaron varios escenarios posibles con el auxilio de juegos de datos, de forma tal que se visualizaran las diferentes funcionalidades que debería ofrecer el sistema.

Conclusiones parciales

[21] Para el caso del presente trabajo de diploma, el cliente de la propuesta de solución es el mismo proyecto productivo.

En el presente capítulo, se realizó un estudio acerca del funcionamiento interno de los procesos de negocio presentes en el campo de acción, donde se detectaron deficiencias relacionadas con responsabilidades de procesamiento en la interacción entidad-sistema para el despacho-recepción de documentos esquema. Teniendo en cuenta las deficiencias detectadas, se proponen soluciones cuya esencia radica en factorizar el procesamiento dedicado a las acciones de definición y gestión de documentos esquema, así como establecer un conjunto de herramientas, disponibles en el mismo sistema VUCE, que permitan el procesamiento en línea de ficheros XSD, como alternativa al mecanismo de reporte electrónico en la comunicación entidad-sistema. Asimismo, se definieron los requisitos funcionales y no funcionales asociados al componente donde se concretará la solución que se propone.

Capítulo 3. Análisis y diseño del sistema

Introducción

El presente capítulo está dedicado exclusivamente a dos actividades claves en la concepción de la propuesta de solución. En un primer momento, se toman los requisitos funcionales identificados en el capítulo anterior, y se modelan en término de clases del análisis. Los artefactos obtenidos como resultado de esta actividad, brindan una traducción de tales requisitos en un lenguaje más cercano al desarrollador, que sirve como elemento de entrada a las actividades de diseño. Un segundo momento está dedicado al diseño de la solución, donde se logra expresar las clases del análisis en términos de clases del diseño con atributos y métodos definidos. Se aborda además, las métricas con las que se evaluó el diseño obtenido, así como los resultados (o puntuación) emitidos en cada indicador de calidad. Sirva el presente capítulo como una aproximación a la línea base de la arquitectura del sistema propuesto.

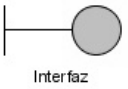
3.1. Modelo de Análisis.

Durante el análisis, se estudian los requisitos que fueron descritos en la actividad de captura de requisitos, con el objetivo de refinarlos y estructurarlos, sin llegar a precisar cómo se implementará la solución. El modelo de análisis es un artefacto descrito en el lenguaje del programador, estructurado por clases y paquetes estereotipados, cuyo objetivo fundamental es ofrecer una vista interna del sistema que sirva de guía para su posterior diseño. Ya en esta fase del proceso, se eliminan redundancias e inconsistencias entre requisitos y se esboza cómo llevar a cabo las funcionalidades significativas para la arquitectura, como primera aproximación al diseño (33).

3.1.1. Diagrama de clases del análisis.

El modelo de análisis está compuesto por clases del análisis y sus objetos organizados en paquetes que colaboran. Las clases del análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema, ya que representan conceptos y relaciones del dominio. Dichas clases se clasifican tal y como se muestra en la siguiente tabla (33).

Tabla 3.0. Clasificación de las clases del análisis para UML 1.x

Entidad	Modelan información que posee larga vida y que es a menudo persistente.	
Interfaz	Modelan la interacción entre el sistema y sus actores.	
Control	Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.	

En una aplicación cliente/servidor con arquitectura MVC, en la capa de la *vista* aparecen fundamentalmente las clases interfaz, debido a que allí es donde se ejecutan las aplicaciones del cliente. En la capa del *modelo* estarían las clases entidad, que modelan la información persistente que maneja el sistema; y la capa controlador, se corresponde con las clases de control, la cuales juegan un papel de intermediarias entre la *vista* y el *modelo* (33).

A continuación se expone el modelo de clases del análisis asociado al requisito funcional *Exportar fichero XSD*, así como el diagrama de comunicación inherente al escenario Despachar fichero XSD desde editor Gráfico, asociado al propio requisito funcional. Análogamente se procedió en el análisis del resto de los requisitos funcionales definidos en el capítulo anterior.

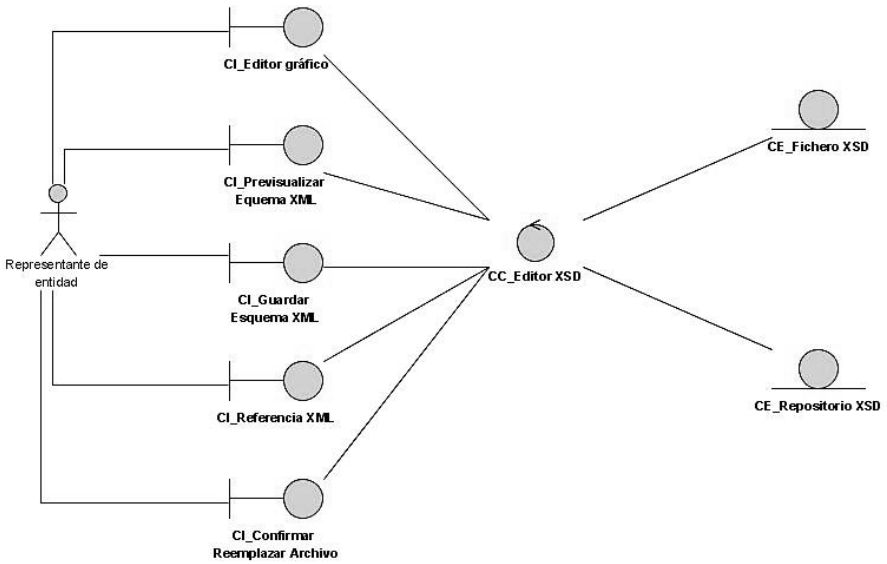


Diagrama 3.0. Diagrama de clases del análisis del requisito *Exportar fichero XSD*.

3.1.2. Diagramas de comunicación.

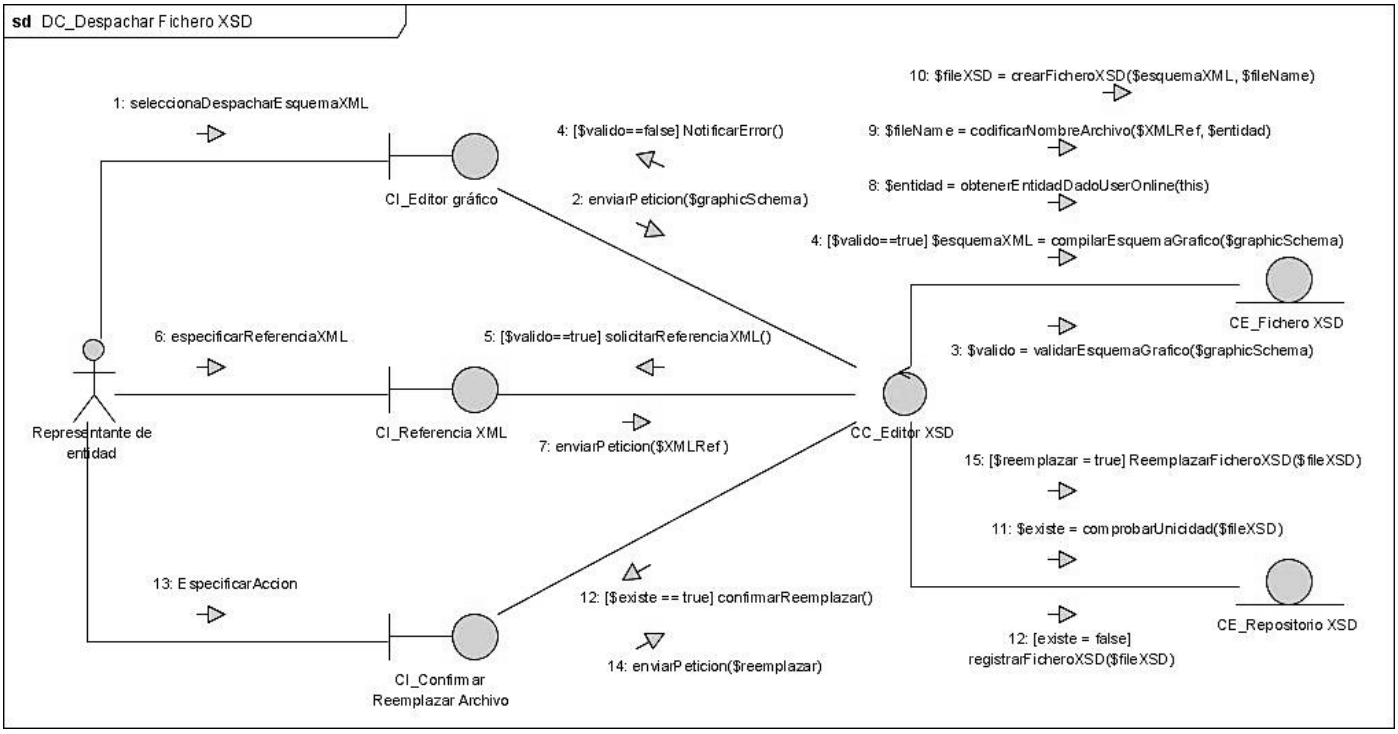


Diagrama 3.1. Diagrama de colaboración del escenario *Despachar fichero XSD*.

3.2. Modelo de Diseño

El diseño es el sitio donde los requisitos del cliente, las necesidades de negocio y las consideraciones técnicas se unen en la formulación de un producto o sistema. Crea una representación o modelo del software, pero a diferencia del modelo de análisis (que se enfoca en la descripción de los datos, las funciones y el comportamiento requerido), el modelo de diseño proporciona detalles acerca de la estructura de datos, la arquitectura, las interfaces y los componentes del software que son necesarios para implementar el sistema (33).

3.2.1. Diagrama de clases del diseño

Los diagramas de clases del diseño (DCD) son ampliamente utilizados en el modelado de sistemas orientados a objetos, empleándose para representar las relaciones que se establecen entre las clases…

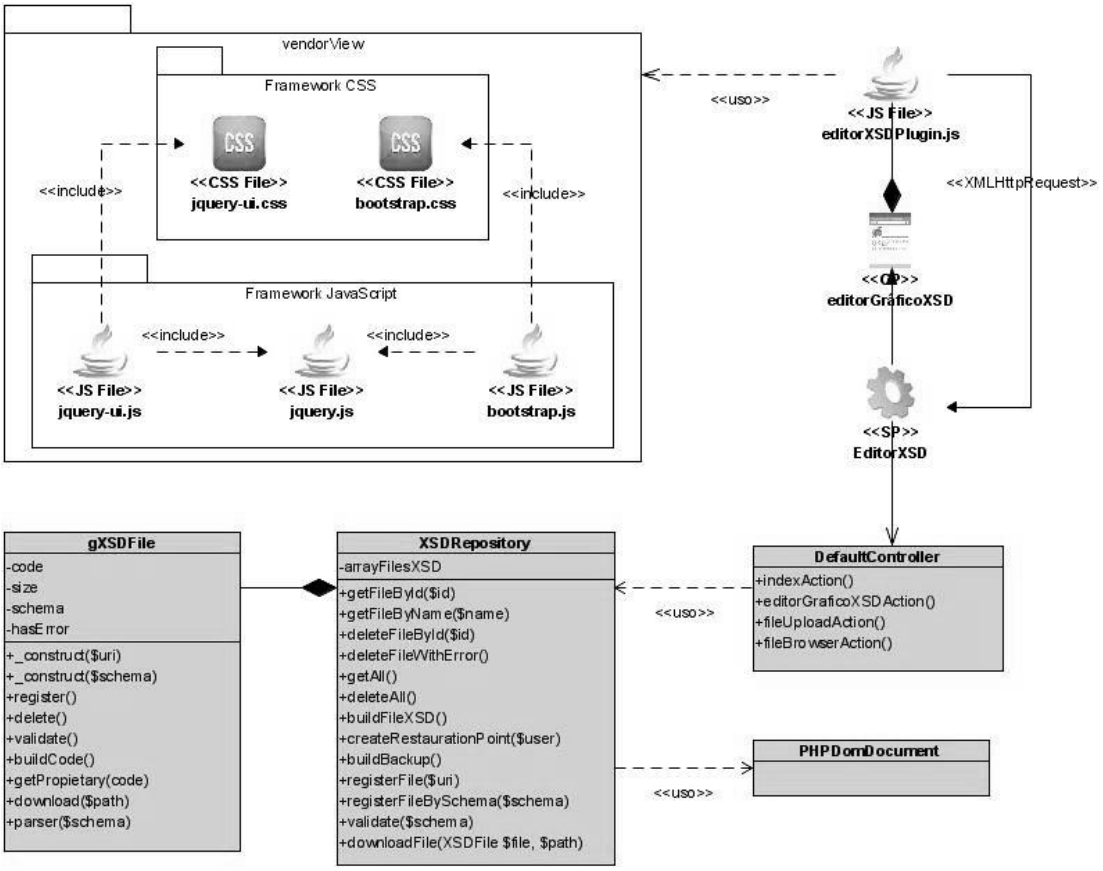


Diagrama 3.2. Diagrama de clases del diseño con estereotipos web asociado al requisito *Exportar fichero XSD*.

… con el objetivo de dar soporte a una funcionalidad determinada. Un DCD describe gráficamente las especificaciones de las clases de software, modeladas con atributos y métodos definidos. El diagrama 3.2 expone la línea base de la arquitectura definida para el requisito Exportar fichero XSD, en la siguiente tabla se expone la descripción de cada clase modelada.

Tabla 3.1. Descripción de las clases del diseño del requisito Exportar fichero XSD.

Clases	Descripción
Paquete vendorView	Encapsula componentes, librerías y framework de terceros, entre ellos: núcleo de la librería jQuery, framework de interfaces de usuario jquey-ui,

	núcleo del framework CSS Bootstrap, entre otros recursos utilizados directamente por el Editor XSD en línea.
editorXSDPlugin.js	Plugin de jQuery desarrollado como parte de la solución, que encapsula el conjunto de componentes Javascript que conforman el núcleo del Editor XSD en línea.
editorXSD.html	Página cliente donde se instancia el Editor XSD en línea.
DefaultController.php	Clase controladora generada por el framework arquitectónico Symfony 2.
gXSDRepository.php	Clase del modelo que encapsula las funcionalidades necesarias para interactuar con los ficheros XSD presentes en un directorio determinado.
gXSDFile.php	Clase del modelo que encapsula funcionalidades cuyo alcance está limitado a un fichero XSD en particular.
DomDocument.php	Clase propia del lenguaje PHP destinada a la manipulación XML.

3.3. Patrones de arquitectura y diseño empleados en la solución

Modelo Vista Controlador (MVC)

El patrón arquitectónico MVC está implícito en el núcleo de la línea base arquitectónica que rige la solución que se propone. A partir del ejemplo expuesto en el diagrama 3.2, nótese que tanto la página cliente editorXSD.html, el plugin de jQuery editorXSDPlugin.js, como el paquete vendorView; conforman la capa de la vista. La clase de symfony DefaultController.php constituye el controlador; y las clases PHP gXSDRepository, gXSDFile y DomDocument, conforman la capa del modelo. El flujo de trabajo se origina en la vista cuando una petición AJAX llega al controlador solicitando datos de las clases del modelo, el controlador invoca al modelo para obtener la información solicitada, para luego enviar a la vista los resultados obtenidos.

Controlador

Asimismo, en la clase DefaultController.php se evidencia el patrón GRASP **Controlador,** ya que esta clase juega el rol de intermediaria entre la vista y la funcionalidad del modelo que la implementa, de manera que es la que recibe los datos del usuario y la que los envía a las distintas clases, consecuentemente con la funcionalidad invocada. Este patrón está estrechamente relacionado con la

filosofía que defiende el patrón arquitectónico MVC, ya que ambos trabajan en función de separar la lógica de negocio de la capa de presentación, en pos de aumentar la reutilización de código.

Patrón Observador u Observador/Observado

Este patrón fue utilizado en la implementación del componente para el despacho/recepción de ficheros XSD, debido a que se requería un proceso de notificación entre el navegador de archivos (componente *File-browser*) y el repositorio central, cada vez que en este último se registrara un nuevo archivo XSD. Se empleó además, en la implementación del mecanismo de sincronización que se ejecuta sobre cada subcomponente que integra la solución *(Editor Gráfico XSD, File-Browser)*, debido a que un mismo fichero XSD puede estarse procesando en ambos subcomponentes al mismo tiempo.

Alta cohesión y bajo acoplamiento:

Las clases que implementan la lógica del negocio se encuentran en el modelo, las cuales no tienen asociaciones con las clases de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja. Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es el mismo controlador, el cual está formado por varias funcionalidades que están estrechamente relacionadas, siendo éste el responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

3.4. Validación del diseño

Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto (33). Los criterios a medir pueden variar en la aplicación de una u otra métrica; no obstante, indicadores como los que se exponen en la tabla 3.1, han devenido en los últimos tiempos como estándares de calidad software (33).

Tabla 3.2. Atributos de calidad de software.

Indicador	Descripción
Responsabilidad	Cuantifica la “carga” asignada a una clase en un marco de modelado de un

	dominio o concepto de la problemática en cuestión.
Complejidad de implementación	Cuantifica el grado de dificultad asociado a la implementación de un diseño de clases determinado.
Reutilización	Cuantifica el grado de reusabilidad inherente a una clase o estructura de clase dentro de un diseño de software.
Acoplamiento	Cuantifica el grado de dependencia o interconexión entre clases o estructura de clases, es inversamente proporcional a la característica de Reutilización.
Mantenibilidad	Cuantifica el grado de esfuerzo que requiere llevar a cabo acciones de modificación o extensión de un diseño de software.

A continuación se exponen las métricas concebidas como instrumento para evaluar la calidad de diseño propuesto, a partir de los indicadores de calidad concebidos anteriormente.

Aplicación de la métrica *Tamaño Operacional de Clase (TOC)*

La métrica TOC se basa en el número de métodos pertenecientes a una clase. Cuantifica atributos como: Responsabilidad, Complejidad de implementación y Reutilización; incidiendo directamente en los dos primeros e inversamente en el último atributo antes mencionado.

Tabla 3.3. Tamaño Operacional de Clase.

Atributos	Proporcionalidad
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 3.4. Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Atributos	Categoría	Criterio
Responsabilidad	Baja	< = Prom.(7.6)
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2 * Prom.
Complejidad de implementación	Baja	< = Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2 * Prom.
Reutilización	Baja	> 2 * Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	<= Prom.

Tabla 3.5. Umbrales TOC.

TOC	Criterio
Pequeña	<= Prom
Media	Entre Prom. y 2 * Prom.
Grande	> 2 * Prom.

Resultados del instrumento de evaluación de la métrica TOC.

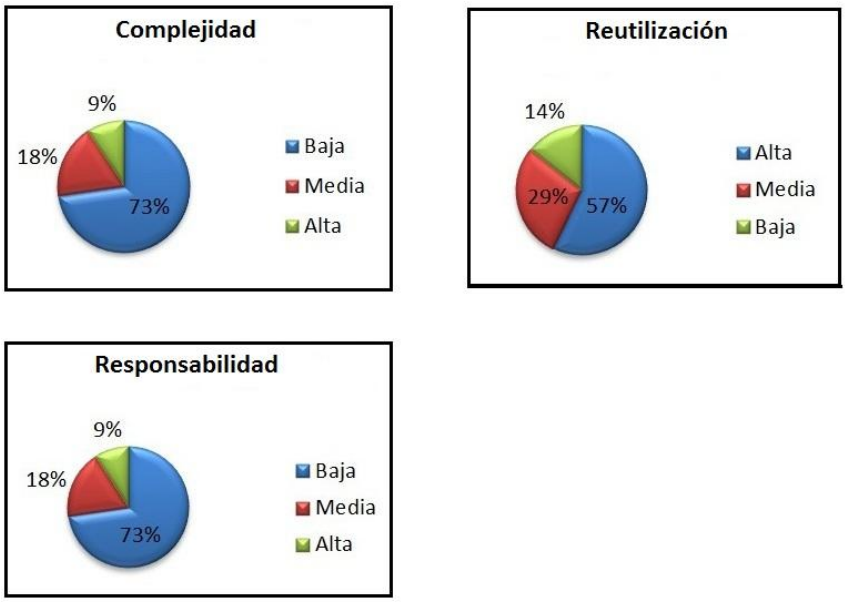


Figura 3.0. Resultado obtenidos en la aplicación de la métrica TOC sobre el diseño propuesto.

Aplicación de la métrica *Relaciones entre Clases (RC)*

La métrica RC cuantifica dado el número de relaciones de uso de una clase, atributos como: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización, existiendo una relación directa en los tres primeros e inversa con el último antes mencionado.

Tabla 3.6. Relaciones entre clases.

Atributo	Proporcionalidad
Acoplamiento	Un aumento de RC implica un aumento del acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento de RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento de RC implica una disminución del grado de reutilización de una clase.
Cantidad de pruebas	Un aumento de RC implica un aumento de la cantidad de pruebas de unicidad necesarias para probar una clase.

Tabla 3.7. Rango de valores para la evaluación de los atributos de calidad relacionados con la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	< =Prom.(0,8)
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom
	Media	Entre Prom. y 2* Prom
	Alta	<= Prom
Cantidad de Pruebas	Baja	< =Prom
	Media	Entre Prom. y 2* Prom
	Alta	> 2* Prom

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC).

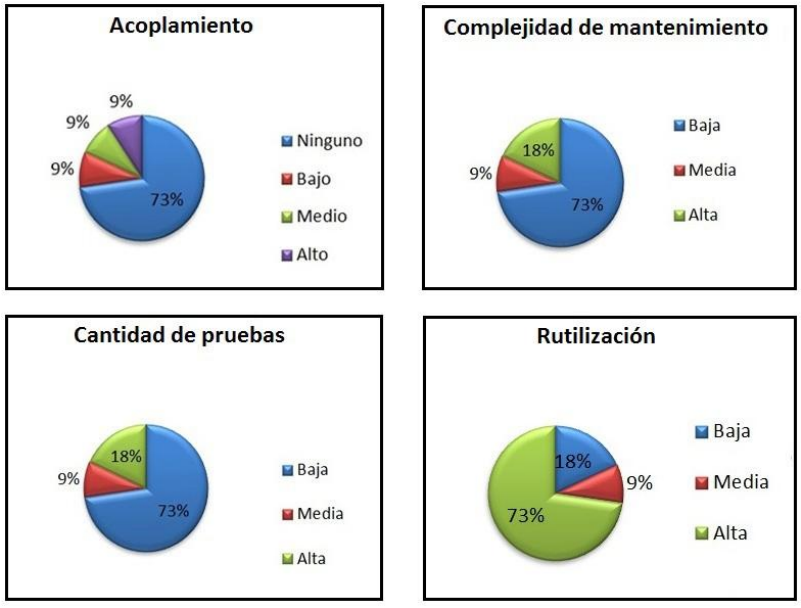


Figura 3.1. Resultado obtenidos en la aplicación de la métrica RC sobre el diseño propuesto.

Resumen de los resultados obtenidos

Al analizar los resultados obtenidos luego de aplicar ambos instrumentos de medición, se puede concluir que se obtuvo un diseño simple y con una calidad aceptablemente buena. Los indicadores de calidad se encuentran en un nivel satisfactorio, en el 73% de las clases el grado de acoplamiento es mínimo, la Complejidad de mantenimiento y la Cantidad de pruebas es baja en el 73% de las clases. Todo lo cual se traduce en un diseño cohesionado, desacoplado y reutilizable.

Conclusiones parciales

En el presente capítulo se logró expresar los requisitos funcionales del componente en término de clases del análisis. Como resultado de la actividad de análisis, se consiguió refinar, estructurar y comprender de una forma más precisa los requisitos definidos, obteniendo una visión general del sistema a construir, que permitió modelar tales requisitos en un lenguaje más cercano al desarrollador de software. Los resultados de esta disciplina, constituyeron un elemento de entrada indispensable en el diseño de la línea base de la arquitectura asociada a la propuesta de solución, donde se logró modelas las clases del análisis en términos de clases del diseño con atributos y métodos definidos, así como la interacción que se establece entre cada una de ellas, en función de soportar las funcionalidades requeridas por el sistema. El diseño obtenido fue evaluado a través de métricas OO que emiten un criterio cuantitativo de calidad del diseño, basándose en indicadores definidos o atributos de calidad. La aplicación de las métricas TOC y RC, arrojaron resultados satisfactorios inherentes al diseño propuesto.

Capítulo 4. Implementación y prueba

Introducción

El presente capítulo aborda detalles técnicos relacionados con la implementación de la propuesta de solución, así como las pruebas de software a las que fue sometida la herramienta, que corroboran indicadores de eficacia en un ambiente controlado. Se aborda además, cómo influyen los resultados obtenidos en cada uno de los parámetros operacionales asociados a la variable definida como eje del problema de investigación planteado, en función de la cual gira el desarrollo del presente trabajo de diploma. Sirva este capítulo como un acercamiento tanto a la implementación del sistema, como a su Verificación y Validación (VyV) en su máxima expresión.

4.1. Modelo de Implementación

4.1.1. Diagrama de componentes de implementación

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes (33).

Un diagrama de componentes ilustra los fragmentos de software y los controladores embebidos que conforman un sistema, permiten editar de forma segura todo el contenido que llevarán los archivos de datos de la aplicación. Tienen un nivel de abstracción más elevado que un diagrama de clases. Usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos diagramas se utilizan para describir la vista de implementación estática de un sistema (33).

A continuación se expone el diagrama de componentes asociado al requisito Exportar fichero XSD.

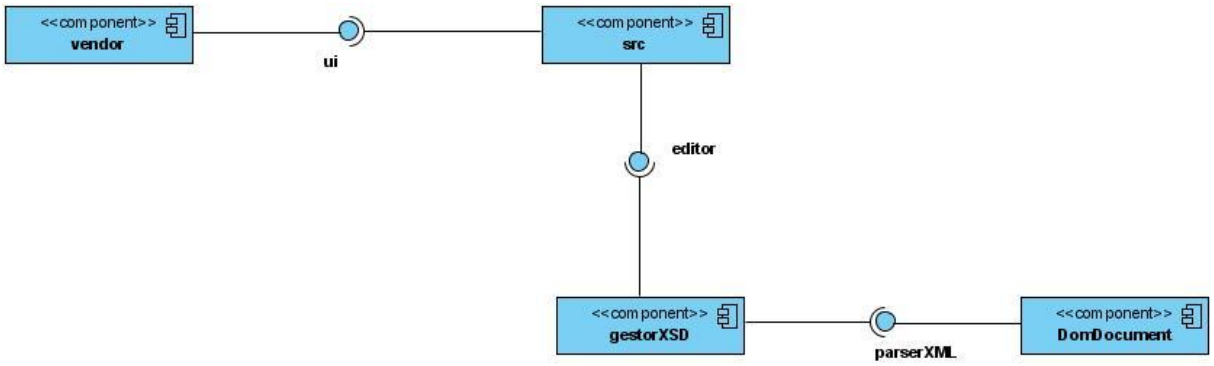


Diagrama 4.0. Diagrama de componentes de implementación asociado al requisito Exportar fichero XSD.

Tabla 4.0. Descripción de los componentes de implementación.

Componente	Descripción
vendor	Encapsula código fuente de terceros reutilizado en la propuesta de solución.
src	Encapsula la codificación JavaScript del editor XSD en línea.
gestorXSD	Encapsula la codificación PHP encargada de interactuar con el sistema operativo para la manipulación de los ficheros XSD.
DomDocument	Componente propio de PHP brindado como utilidad del lenguaje para la manipulación XML.

4.2. Estándar de codificación

Las convenciones o estándares de codificación son un conjunto de directrices que especifican cómo debe escribirse el código fuente. Un estándar de código se basa en la estructura y apariencia física de un programa, con el fin de facilitar la lectura, comprensión, mantenimiento del código y reutilización a lo largo del proceso de desarrollo de un software; y no en la lógica del programa. Este no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y la legibilidad del código, aspecto crucial a la hora de darle mantenimiento y mejorar las funcionalidades de un software (45).

Las normas, patrones y estilo de codificación empleado en la implementación de la solución propuesta, se corresponde exactamente con el estándar de codificación definido para las aplicaciones desarrolladas

bajo la jurisdicción del Departamento de Soluciones para la Aduana del CEIGE; el cual comprende todo el código generado bajo la tecnología y el lenguaje PHP, que utilicen la arquitectura regida por la utilización del Framework Arquitectónico Symfony 2.

4.3. Modelo de prueba de software

4.3.1. Pruebas de caja blanca

A continuación se ilustra el procedimiento de prueba realizado sobre la funcionalidad que manipula el estado de los botones de navegación en el esquema gráfico, análogamente se procedió para el resto de las principales funcionalidades de la solución.

Conceptos claves en el procedimiento realizado

Componentes de un diagrama de flujo: 1) **Nodo,** cada círculo representado se denomina nodo del Grafo de Flujo, simboliza una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una secuencia de decisión; 2) **aristas,** Flechas que unen los nodos del grafo, representan el flujo de control; 3) **regiones,** áreas delimitadas por las aristas y los nodos (también se incluye el área exterior del grafo como una región más); la cantidad de regiones es equivalente a la cantidad de caminos independientes del flujo básico de una secuencia procedimental (33).

Complejidad ciclomática V(G): Define el número de *caminos independientes* del conjunto básico de un programa. Devuelve el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute al menos una vez cada sentencia del programa (33).

Grafo de flujo asociado al fragmento de código anterior

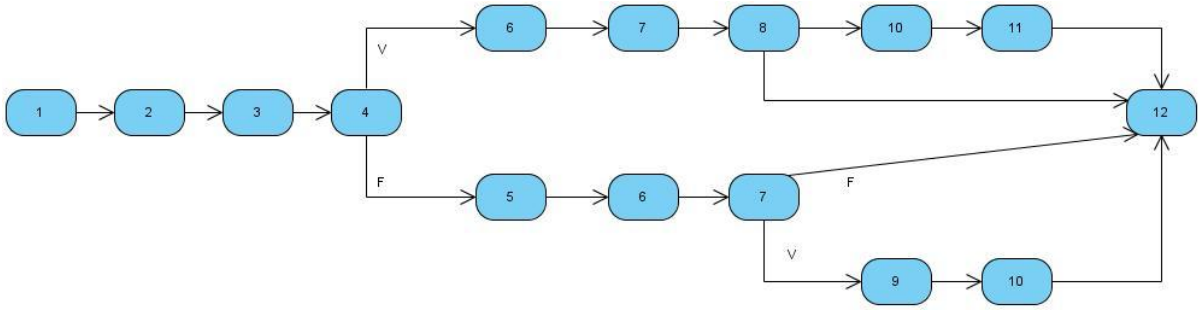


Figura 4.0. Grafo de flujo asociado a la funcionalidad *$editor.btnSchemaEvtListener().*

(Nota: Ver fragmento de código analizado en el anexo 4).

Fórmulas para calcular la complejidad ciclomática:

1. **V(G) = (A - N) + 2**
   Donde "**A**" es la cantidad de aristas y "**N**" la cantidad de nodos.
   V (G) = (17 - 15) + 2
   V (G) = 4
2. **V (G) = P + 1**
   Siendo "**P**" la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).
   V (G) = 3 + 1
   V (G) = 4
3. **V (G) = R**
   Donde "**R**" representa la cantidad de regiones en el grafo **G**.
   V (G) = 4

El cálculo efectuado mediante las fórmulas anteriores ha arrojado el mismo valor, con lo que se puede concluir que la complejidad ciclomática del fragmento de código analizado es numéricamente igual a 4. A continuación se detallan los flujos procedimentales posibles:

Tabla 4.0. Caminos básicos del flujo.

Número	Caminos básicos
1	1-2-3-4-5-6-7-9-10-14
2	1-2-3-4-5-6-7-14
3	1-2-3-4-6-7-8-10-11-14
4	1-2-3-4-6-7-8-14

Una vez determinados los caminos básicos, se procede a ejecutar los casos de prueba para cada uno de ellos. Para definir los casos de prueba es necesario tener en cuenta:

- **Descripción:** describe el caso de prueba de manera general, tratando los aspectos fundamentales de los datos de entrada.

- **Condición de ejecución:** especifica las premisas que deben existir para que pueda ejecutarse el caso de prueba.
- **Resultado esperado:** expone el resultado que se espera devuelva el procedimiento después de efectuado el caso de prueba.
- **Resultado obtenido:** se expone el resultado que finalmente se obtuvo al ejecutar el procedimiento o simplemente, si el resultado obtenido se corresponde satisfactoriamente con el resultado esperado.

Caso de prueba para el camino básico #1:

Descripción: Se selecciona en el esquema gráfico el tag raíz (<xs:schema>) cuando este tiene asociado al menos un tag hijo.

Condición de ejecución: El tag raíz no es hoja del árbol definido por el esquema gráfico.

Resultados esperados:

1. El cursor debe ubicarse como último hijo del tag raíz.
2. Deben habilitarse los botones de navegación *"primerHijo"* y *"ultimoHijo"*.
3. Deben deshabilitarse los botones de navegación *"hermanoArriba"* y *"hermanoAbajo"*.
4. Debe ubicarse el foco sobre el botón de navegación *"ultimoHijo"*.

Resultado obtenido: satisfactorio

Caso de prueba para el camino básico #2:

Descripción: Se selecciona en el esquema gráfico un tag distinto de <*xs:schema*>, cuando este tiene asociado al menos un tag hijo.

Condición de ejecución: Existe al menos una ramificación de nivel 2 en el árbol definido por el esquema gráfico.

Resultados esperados:

1. El cursor debe ubicarse como último hijo del tag seleccionado.

2. Deben habilitarse los botones de navegación *“primerHijo”, “ultimoHijo”, “hermanoArriba” y “hermanoAbajo”*.
3. Debe ubicarse el foco sobre el botón de navegación *“ultimoHijo”*.

Resultado obtenido: satisfactorio

Caso de prueba para el camino básico #3:

Descripción: Se selecciona en el esquema gráfico el tag raíz cuando este no tiene tags hijos asociados.

Condición de ejecución: El tag raíz es hoja del árbol definido por el esquema gráfico.

Resultados esperados:

1. El cursor debe ubicarse como hijo del tag raíz.
2. Deben deshabilitarse los botones de navegación *“primerHijo”* y *“ultimoHijo”*.
3. Deben deshabilitarse los botones de navegación *“hermanoArriba”* y *“hermanoAbajo”*.
4. Debe habilitarse el botón de navegación *“hijo”*.
5. Debe ubicarse el foco sobre el botón de navegación *“hijo”*.

Resultado obtenido: satisfactorio.

Caso de prueba para el camino básico #4:

Descripción: Se selecciona en el esquema gráfico un tag distinto de <*xs:schema*>, cuando éste no tiene tags hijos asociados.

Condición de ejecución: Existe al menos una ramificación de nivel 1 en el árbol definido por el esquema gráfico.

Resultados esperados:

1. El cursor debe ubicarse como hijo del tag seleccionado.
2. Deben habilitarse los botones de navegación *“hermanoArriba”* y *“hermanoAbajo”*.
3. Deben deshabilitarse los botones de navegación *“primerHijo”* y *“ultimoHijo”*.
4. Deben habilitarse el botón de navegación *“hijo”*.

5. Deben ubicarse el foco sobre el botón de navegación *“hijo”*.

Resultado obtenido: satisfactorio

4.3.2. Pruebas de caja negra

A continuación se ilustra el procedimiento realizado para el caso del requisito funcional *Crear Esquema XML*, análogamente se procedió para el resto de los requisitos funcionales de la solución propuesta.

Tabla 4.1. Diseño de casos de prueba asociada al requisito Crear esquema XML.

Requisito	Descripción	Escenario de prueba	Flujo del escenario
RF_1.1 Obtener tag hijos dado tag padre.	Obtiene los tag XSD que resulta válido semánticamente ubicarlos como hijos del tag especificado dentro del esquema XML.	EP_1.1.1. Se especifica como padre al tag <xs:schema> sin que éste tenga tag hijos.	- Se selecciona en el esquema gráfico el tag <xs:schema> bajo las condiciones antes descritas. - Se ubica el cursor como hijo del tag <xs:schema> - Se muestra en el panel de elementos los siguientes tags: 1. xs:include 2. xs:import 3. xs:redefine 4. xs:annotation 5. xs:simpleType 6. xs:complexType 7. xs:group 8. xs:attributeGroup 9. xs:element 10. xs:attribute 11. xs:notation
		EP_1.1.2. Se especifica como padre al tag <xs:schema> cuando éste tiene como hijos los siguientes tags:	- Se selecciona en el esquema gráfico el tag <xs:schema> cuando éste tiene como hijos los tags antes especificados. - Se ubica el cursor como último hijo

		1. xs:include 2. xs:import 3. xs:redefine 4. xs:annotation 5. xs:simpleType 6. xs:complexType 7. xs:group 8. xs:attributeGroup 9. xs:element 10. xs:attribute 11. xs:notation	del tag <xs:schema> - Se muestra en el panel de elementos los siguientes tags: 1. xs:annotation 2. xs:simpleType 3. xs:complexType 4. xs:group 5. xs:attributeGroup 6. xs:element 7. xs:attribute 8. xs:notation
		EP_1.1.3 Se especifica como padre al tag <xs:schema> cuando éste tiene como hijos los siguientes tags: 1. xs:include 2. xs:import 3. xs:redefine 4. xs:annotation	- Se selecciona en el esquema gráfico el tag <xs:schema> cuando éste tiene como hijos los tags antes especificados. - Se ubica el cursor como último hijo del tag <xs:schema>. - Se muestra en el panel de elementos los siguientes tags: 1. xs:include 2. xs:import 3. xs:redefine 4. xs:annotation 5. xs:simpleType 6. xs:complexType 7. xs:group 8. xs:attributeGroup 9. xs:element 10. xs:attribute 11. xs:notation
		EP_1.1.4 Se especifica como padre al tag <xs:schema> cuando éste	- Se selecciona en el esquema gráfico el tag <xs:schema> cuando éste tiene como hijos los tags antes

		tiene como hijos los siguientes tags: 1. xs:simpleType 2. xs:complexType 3. xs:group 4. xs:attributeGroup 5. xs:element 6. xs:attribute 7. xs:notation	especificados. - Se ubica el cursor como último hijo del tag <xs:schema>. - Se muestra en el panel de elementos los siguientes tags: 1. xs:annotation 2. xs:simpleType 3. xs:complexType 4. xs:group 5. xs:attributeGroup 6. xs:element 7. xs:attribute 8. xs:notation
		EP_1.1.5 Se especifica como padre al tag <xs:simpleType> cuando éste no tiene tag hijos:	- Se selecciona en el esquema gráfico el tag <xs:simpleType> cuando éste no tiene tag hijos. - Se ubica el cursor como hijo del tag <xs: simpleType > seleccionado. - Se muestra en el panel de elementos los siguientes tags: 1. xs:annotation 2. xs:restriction 3. xs:list 4. xs:union
		EP_1.1.6 Se especifica como padre al tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:annotation>	- Se selecciona en el esquema gráfico el tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:annotation>. - Se ubica el cursor como último hijo del tag <xs: simpleType > seleccionado. - Se muestra en el panel de elementos los siguientes tags:

			1. xs:restriction 2. xs:list 3. xs:union
		EP_1.1.6 Se especifica como padre al tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:restriction>	- Se selecciona en el esquema gráfico el tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:restriction>. - Se ubica el cursor como último hijo del tag <xs: simpleType > seleccionado. - Se muestra el panel de elementos vacío:
		EP_1.1.7 Se especifica como padre al tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:list>	- Se selecciona en el esquema gráfico el tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:list>. - Se ubica el cursor como último hijo del tag <xs: simpleType > seleccionado. - Se muestra el panel de elementos vacío:
		EP_1.1.8 Se especifica como padre al tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:union>	- Se selecciona en el esquema gráfico el tag <xs:simpleType> cuando éste tiene como único hijo al tag <xs:union>. - Se ubica el cursor como último hijo del tag <xs: simpleType > seleccionado. - Se muestra el panel de elementos vacío:

Resumen de los resultados obtenidos.

El componente fue sometido a tres iteraciones de pruebas de software. Se detectaron un total de 25 no conformidades repartidas en indicadores como: 1) mensajes del sistema, 2) recomendación[22], 3) error de funcionalidad, 4) error de interfaz, 5) redacción y 6) ortografía.

En el siguiente gráfico se resume el total de no conformidades por iteraciones:

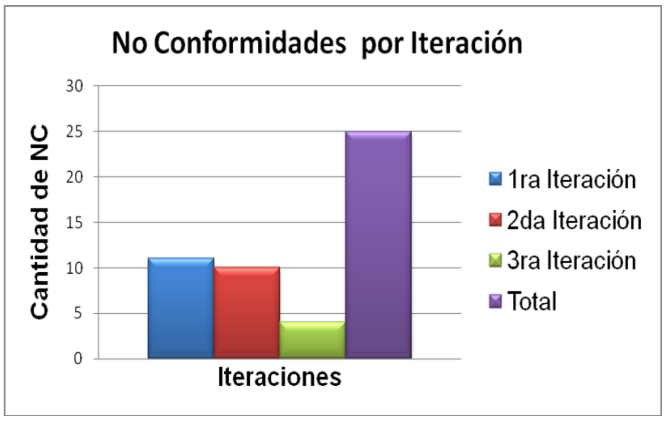


Figura 4.1. Resumen de no conformidades por iteraciones.

Asimismo, se resumen las no conformidades agrupadas en los indicadores de errores definidos.

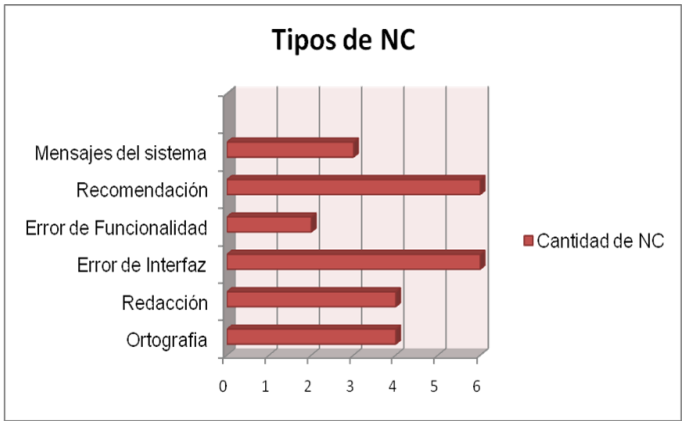


Figura 4.2. No conformidades agrupadas en indicadores de error.

[22] Recomendaciones en forma de ayuda que brinda la herramienta como guía en el proceso de creación de esquema XML.

Conclusiones Generales

El desarrollo de la presente investigación permitió arribar a las siguientes conclusiones:

- El proceso de centralización es un hecho inevitable en ambientes de negocio donde determinado resultado depende del concurso de diversas fuentes participantes, elemento que debe ser considerado desde el proceso de conceptualización de todo producto software desarrollado para este propósito.
- El estudio del estado del arte permitió identificar componentes y elementos funcionales presentes en los principales sistemas de software para la definición/edición de documentos esquema; a partir de los cuales, se definió e incluyó un criterio propio como atributo de calidad indispensable para asistir esta actividad desde una propuesta basada en la web.
- Se obtuvo un componente basado en la web para la definición/edición de documentos esquema, incluyendo el procesamiento de ficheros XSD, que permite una gestión homogénea de documentos esquema a partir de criterios de centralización y facilidad de uso.
- Se obtuvo una solución con un alto nivel de escalabilidad, reutilización de componentes y generalización.
- La solución obtenida es pionera en el campo de los editores XSD basados en la web.

Bibliografía

1. Organización Mundial del Comercio. [En línea] OMC. [Citado el: 10 de diciembre de 2012.] http://www.wto.org/spanish/info_s/cont_s.htm.

2. Ventanilla Digital Mexicana de Comercio Exterior. [En línea] [Citado el: 12 de diciembre de 2012.] http://www.ventanillaunica.gob.mx/vucem/index.htm.

3. **Service, Korea Customs.** *Global Top Customs.* UNI-PASS, 2010.

4. *Ventanillas Únicas de comercio exterior en América Latina y el Caribe: avances y retos pendientes.* **SELA, Secretaría Permanente.** Caracas, Venezuela : Oficina de Prensa y Difusión de la Secretaría Permanente del SELA, 2011. SP/CL/XXXVII.O/Di No. 17-11.

5. **Arnautó, Mercedes Suárez.** *Modelo para la facilitación del despacho mercantil en Cuba.* Universidad de las Ciecias Informáticas. Ciudad de la Habana : s.n., 2009.

6. **Solis, Luis Adrián Salazar.** *Ventanilla Única de Gobierno Electrónico.* Costa Rica : s.n., 2006.

7. **Libby, Margarita H.** *Simplificación de trámites para promover la competitividad.* Santo Domingo, Rep. Dominicana : Compete Caribbean, 2011.

8. **Michael Kantor, James H. Burrows.** *Electronic Data Interchange.* s.l. : National Institute of Standards and Technology, 2008.

9. **Dpto. de Informática. Universidad de Vigo.** *EDI y VAN en la empresa globalizada.* Vigo, España : s.n., 2010.

10. **AMECE.** *Manual Intercambio Electrónico de Datos (EDI).* Mexico : s.n., 2010.

11. **Sturn, Jack.** *Desarrollo de Soluciones XML.* s.l. : Serie de programación Microsoft, 2000.

12. **W3C.** *Extensible Markup Language (XML) 1.0 (Fifth Edition).* [En línea] 26 de Noviembre de 2008. [Citado el: 5 de diciembre de 2013.] http://www.w3.org/TR/2008/REC-xml-20081126/#sec-origin-goals.

13. **Romero, Alfredo Reino.** *Introducción a XML en Castellano.* Mexico : Universidad de Colima, 2000.

14. **RI5 Informática Argentina.** *Qué es XML y para que sirve.* [En línea] [Citado el: 8 de enero de 2013.] http://www.ri5.com.ar/ayuda07.php.

15. **W3C.** XML Schema Part 1: Structures Second Edition. *Constraints and Validation Rules.* [En línea] 2004. [Citado el: 27 de feb de 2013.] http://www.w3.org/TR/xmlschema-1/#concepts-schemaConstraints.

16. —. Extensible Markup Language (XML) 1.1. *Well-Formed XML Documents.* [En línea] 2004. http://www.w3.org/TR/2004/REC-xml11-20040204/#sec-well-formed.

17. **Jack Sturm, Microsoft.** *Desarrollo de soluciones XML.* s.l. : McGraw-Hill, 2001.

18. **Neil Bradley, Addison-Wesley.** *The XML Schema Companion.* 2003.

19. **W3C.** W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. [En línea] 5 de April de 2012. http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/.

20. —. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. [En línea] 5 de April de 2012. http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/.

21. **Mora, Sergio Luján.** *XML Schema.* s.l. : Universidad de Alicante.

22. **Clark, James.** *TREX - Tree Regular Expressions for XML - "TREX has been merged with RELAX to create RELAX NG.*

23. **OASIS.** RELAX NG Specification. [En línea] 3 de 12 de 2001. http://www.oasis-open.org/committees/relax-ng/spec-20011203.html.

24. **Makoto, Murata.** *RELAX (Regular Language description for XML) -- "RELAX NG of OASIS. It is a schema language created by unifying RELAX Core and TREX.* 2002.

25. **OASIS.** RELAX NG Compact Syntax. [En línea] 21 de 11 de 2002. http://www.oasis-open.org/committees/relax-ng/compact-20021121.html.

26. **Academia Sinica Computing Centre.** Schematron. *Academia Sinica Computing Centre's Schematron Home Page.* [En línea] http://www.ascc.net/xml/resource/schematron.

27. **ISO.** Schematron Home Page. [En línea] http://www.schematron.com.

28. **Team, Oxygen.** Oxygen XML Editor. [En línea] SynRO Soft SRL, 2002. http://www.oxygenxml.com/.

29. **Liquid Technologies Ltd.** Liquid XML Studio 2013. [En línea] Liquid Technologies Ltd. http://www.liquid-technologies.com/xml-studio.aspx.

30. **Syntext, Inc.** bitbucket serna free. [En línea] https://bitbucket.org/syntext/serna-free/.

31. **Centro de Informatización de Gestión de Entidades.** *Modelo de desarrollo de software.* Habana : s.n., 2012.

32. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El poceso unificado de desarrollo de software.* Madrid : Adison Wesley, 2000. ISBN 84-7829-036-2.

33. **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico.* 2005.

34. **Kicillof, Nicolás.** Estilos y patrones en la estrategia de arquitectura de Microsoft. [En línea] 2004. http://www.willydev.net/descargas/prev/Estiloypatron.pdf.

35. **Lago, Ramiro.** Patrones de diseño. 2007.

36. **Larman, Craig.** *UML y patrones, introducción al análisis y diseño orientado a objetos.* s.l. : Prentice-Hall, 2002.

37. **Object Management Group.** Bisiness Process Modeling Notation (BPMN), versión 2.1. [En línea] 2009. http//www.omg.org/spec/BPMN/1.2.

38. Visual Paradign. [En línea] http://www.visual-paradigm.com/.

39. **Team, Symfony.** Open-Source PHP Web Framework. [En línea] http://www.symfony-project.org.

40. **Fabien Potencier, François Zaninotto.** *Symfony la guía definitiva.* Paris : s.n., 2008.

41. **The jQuery Foundation.** jQuery Licence. [En línea] https://jquery.org/license/.

42. —. jQuery. [En línea] https://jquery.org/.

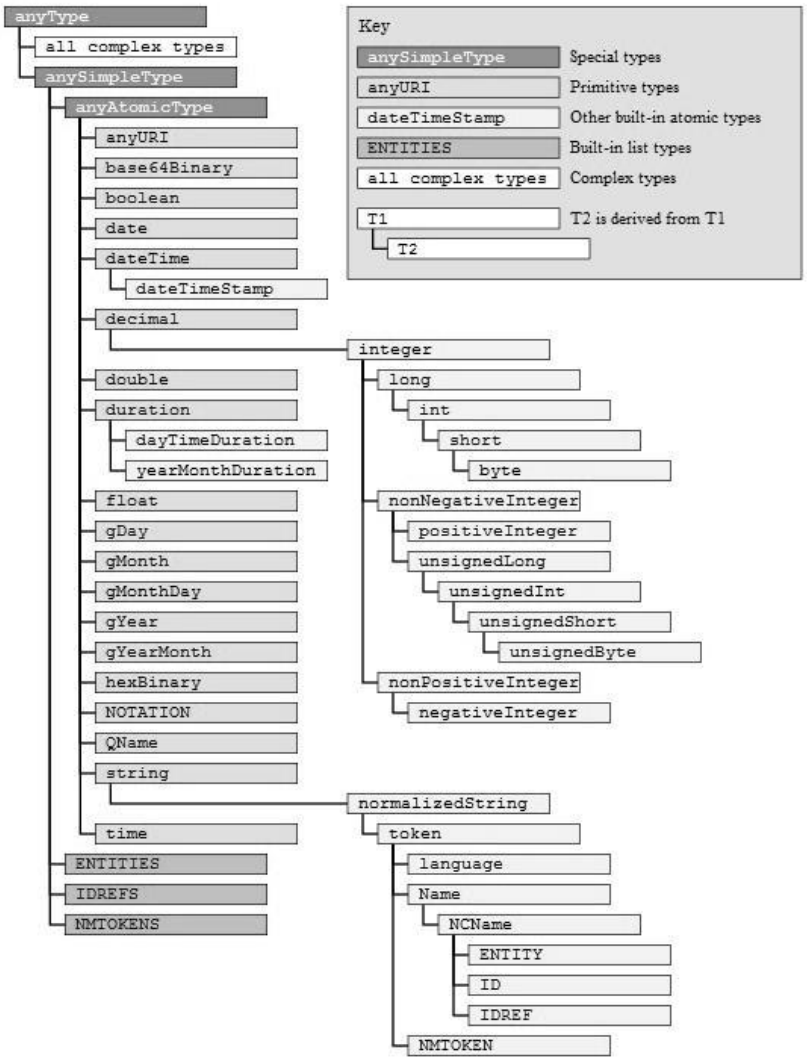
43. **Cochran, David.** *Twitter Bootstrap Web Development.* s.l. : Packt Publishing, 12 de noviembre de 2012.

44. **Sommerville, Ian.** *Ingeniería del Software.* Madrid : ADDISON-WESLEY, 2005.

45. **Herranz, Raúl.** Utópica Informática. [En línea] diciembre de 2010. http://utopicainformatica.blogspot.com/2010/12/convencionesestandares-de-codificacion.html.

Anexos

Anexo 1. Tipos de datos soportados en la versión 1.1 de W3C XML Schema



Anexo 2. Listado de requisitos funcionales organizados por paquetes

REQUISITOS FUNCIONALES	
Paquete << Repositorio XSD >>	*Paquete <<Editor XSD Online>>*

RF_1. Cargar fichero xsd. RF_2. Verificar xsd. RF_3. Registrar fichero xsd. RF_4. Listar ficheros xsd. RF_5. Buscar fichero xsd. RF_6. Ordenar fichero xsd. RF_7. Eliminar fichero xsd. RF_8. Reemplazar fichero xsd. RF_9. Visualizar fichero xsd en modo solo lectura. RF_10. Visualizar fichero xsd en modo editable. RF_11. Descargar fichero xsd. RF_12. Archivar fichero xsd. RF_13. Generar backup. RF_14. Crear punto de restauración.	RF_15. Agregar tag. RF_16. Configurar tag. RF_17. Estructurar tags. RF_18. Eliminar tag. RF_19. Validar estructura de tags. RF_20. Eliminar árbol de tag. RF_21. Agrupar tags. RF_22. Generar esquema XML. RF_23. Visualizar esquema XML. RF_24. Guardar esquema XML como fichero xsd. RF_25. Registrar esquema XML.
Paquete <<Administración del editor XSD>> RF_26. Listar tags xsd base. RF_27. Buscar tag xsd base. RF_28. Editar tag xsd base. RF_29. Eliminar tag xsd base. RF_30. Adicionar tag xsd base.	

Anexo 3. Matriz de trazabilidad

Objetivos / Agrupaciones de requisitos	RF_1 Crear esquema XML Online.	RF_2 Exportar Fichero XSD.	RF_3 Importar Fichero XSD.	RF_4 Despachar Fichero XSD vía *File-upload*.	RF_5 Gestionar Fichero XSD.
Obtener tags XSD válidos dado posición en la jerarquía XSD.	X				
Obtener atributos válidos dado tag XSD.	X				
Obtener valores de atributos válidos dado atributo de tag XSD.	X				
Agregar tag XSD al esquema gráfico.	X				
Eliminar tag XSD del esquema gráfico.	X				
Eliminar árbol de tags XSD del esquema de gráfico.	X				
Configurar tag XSD en el esquema gráfico.	X				
Validar estructura del esquema gráfico.	X				
Obtener esquema XML a partir de esquema gráfico.		X			
Crear fichero XSD a partir de esquema XML.		X			
Pre-visualizar esquema XML.		X			
Descargar fichero XSD.		X			
Codificar nombre de fichero XSD.		X		X	
Registrar fichero XSD en repositorio central.		X		X	
Subir fichero XSD al sistema.			X	X	
Análisis sintáctico de fichero XSD.			X		
Crear esquema gráfico a partir del resultado del análisis sintáctico.			X		
Validar extensión de fichero XSD.			X	X	
Validar acción upload de fichero XSD.			X	X	
Comprobar validez sintáctica de fichero XSD.		X		X	
Cancelar despacho de fichero XSD.				X	
Reemplazar fichero XSD.		X		X	X
Editar Fichero XSD.	X			X	X
Eliminar Fichero XSD.					X

Anexo 4. Fragmento de código asociado a la funcionalidad *$editor.btnSchemaEvtListener().*

```
59          $editor.btnSchemaEvtListener = function() { //1
60              //2
61              $('#gXSD_schemaBtn_btherUp').removeClass('btn-primary').removeClass('disabled');
62              //3
63              $('#gXSD_schemaBtn_btherDown').removeClass('btn-primary').removeClass('disabled');
64              //4
65              if ( ($('#' + $editor.currentTag.attr('partialId') +'children').children().length <= 1) ||
66                  ($('#' + $editor.currentTag.attr('partialId') +'children').children().length == 1 &&
67                   $('#' + $editor.currentTag.attr('partialId') +'children').find('#gXSD_cursor').length == 1) )
68              {
69                  //6
70                  $('#gXSD_schemaBtn_childUp').removeClass('btn-primary').addClass('disabled').hide();
71                  //7
72                  $('#gXSD_schemaBtn_childDown').addClass('btn-primary').removeClass('disabled').text('children');
73                  //8
74                  if ($editor.currentTag.attr('tag') == 'schema') {
75                      //10
76                      $('#gXSD_schemaBtn_btherUp').removeClass('btn-primary').addClass('disabled');
77                      //11
78                      $('#gXSD_schemaBtn_btherDown').removeClass('btn-primary').addClass('disabled');
79                  }
80              }
81              else {
82                  //5
83                  $('#gXSD_schemaBtn_childUp').removeClass('btn-primary').removeClass('disabled').show();
84                  //6
85                  $('#gXSD_schemaBtn_childDown').addClass('btn-primary').removeClass('disabled').text('childDown');
86                  //7
87                  if ($editor.currentTag.attr('tag') == 'schema') {
88                      //9
89                      $('#gXSD_schemaBtn_btherUp').removeClass('btn-primary').addClass('disabled');
90                      //10
91                      $('#gXSD_schemaBtn_btherDown').removeClass('btn-primary').addClass('disabled');
92                  }
93              }
94          } //12
```